Deep Learning with Multi-Dimensional Medical Image Data

Vom Promotionsausschuss der
Technischen Universität Hamburg

zur Erlangung des akademischen Grades

Doktor-Ingenieur (Dr.-Ing.)

genehmigte Dissertation

von
Nils Thorben Gessert

aus
Henstedt-Ulzburg

2020

Gutachter:

- Prof. Dr.-Ing. Alexander Schlaefer
- Prof. Dr.-Ing. Rolf-Rainer Grigat

Tag der mündlichen Prüfung: 17.12.2020

Acknowledgements

The work presented in this dissertation was supported by several other people. First, I would like to thank my advisor Alexander Schlaefer for the opportunity to conduct research under his guidance. I was always given the freedom to go into any research direction and explore my ideas without limitations. Our numerous and long discussions have always been very fruitful and allowed me to maintain a critical view of the new research field that I explored in this dissertation. I am very grateful for his mentoring and guidance which shaped me to be the researcher I am today.

I would also like to thank my co-advisor Rolf-Rainer Grigat for his support, starting with my master thesis and leading up to this work.

Throughout my time as a doctoral candidate, I was fortunate to work with many great colleagues and researchers who helped me in the pursuit of my research goals. My thanks go to my colleague and friend Marcel Bengs with whom I have pursued many exciting and fruitful research projects. Nights of working together and trips to insightful conferences have been crucial for the making of this dissertation. I am also grateful for the support of my colleagues Matthias Schlüter, Sarah Latus, Christoph Otte, Sven-Thomas Antoni, Omer Rajput, Maximilian Neidhardt, Martin Gromniak, Stefan Gerlach, and Johanna Sprenger. My thanks also go to Thore Saathoff, Michael Freude, and Katrin Rausch for their support during the everyday work at the institute. In addition, I would like to thank my research collaborators Matthias Lutz, Roland Opfer, Julia Krüger, and David Ellebrecht.

Furthermore, I would like to thank my girlfriend for her continuous support and encouragement throughout the making of this work. Also, my thanks go to my brother who initially convinced me to become a researcher in the field of machine learning. Last but not least, my thanks go to my family and friends.

Abstract

Over the last few years, deep learning methods have shown tremendous success for a variety of image analysis problems such as classification, object detection, and semantic segmentation. In the natural image domain, numerous applications have been studied, including face recognition, vision for autonomous vehicles, and action recognition. Often, these applications make use of the same image types, which are usually single or multiple 2D color or depth images. In the medical image domain, deep learning methods have been explored for a lot of applications such as image registration, disease detection, and tissue segmentation. However, for many of these applications, the underlying image data is very different. Various imaging systems are able to provide not only 2D images but also 1D scans, 3D volumes, or even 4D spatio-temporal data. This raises two major questions for deep learning applications with medical images. The first question is which data representation to use for processing. Deep learning algorithms can process an image volume slice-by-slice or the entire volume at once, where the latter might be able to capture inter-slice relationships. The second question is what type of deep learning method should be used for processing the different data dimensions. While using higher-dimensional data promises more context, the curse of dimensionality makes model design very challenging due to high computational requirements and exponentially increasing model parameters that aggravate the risk of overfitting.

In this thesis, we propose new deep learning methods and provide an in-depth analysis of deep learning applications from a multi-dimensional perspective. First, we adapt deep learning architectures and propose new models for processing data with different numbers of dimensions. For spatial data processing, we design a variety of convolutional neural network (CNN) architectures for 1D up to 4D data. We explore multi-dimensional transfer learning strategies and neural architecture search for automatically designing higher-dimensional models efficiently. For problems that require combination of two images, we design Siamese two-path architectures, including a novel attention-based interaction method. For spatio-temporal data processing, we consider joint, convolution-based, as well as decomposed methods using both convolutional and recurrent architectures. In this context, we also propose new convolutional-recurrent CNN models. Second, we apply these techniques to a variety of applications with the imaging modalities optical coherence tomography (OCT) and magnetic resonance imaging (MRI). In the field of computer-assisted interventions, we study pose estimation, motion estimation, and vision-based force estimation with 1D up to 4D OCT data. For diagnostic image analysis, we investigate tissue classification, lesion segmentation, and organ quantification with 1D and 2D OCT data and 2D up to 4D MRI data.

Across multiple applications, we find that CNNs and our new convolutional-recurrent models consistently outperform previous methods for spatio-temporal data processing, including 2D, 3D, and 4D data. Furthermore, our Siamese CNNs can be applied effectively to a broad spectrum of problems requiring the processing of two states. When choosing a suitable data representation, we provide evidence that higher-dimensional data appears to be beneficial in most cases, as the additional information and context compared to lower-dimensional data can be exploited effectively by well-designed deep learning models. Overall, our results indicate that future work on deep learning-based medical image analysis should strive towards using higher-dimensional data.

Contents

1 Introduction

1.1 Medical Imaging

Medical imaging has become a fundamental component in patient care, as it assists medical practitioners in early diagnosis, treatment planning, and during surgical interventions [67]. Imaging the inside of the human body is a challenging task, as light in the visible spectrum is largely reflected at surfaces. For overcoming the problem of a missing line-of-sight, multiple imaging modalities have been introduced for visualizing the inner structure of objects and organs. A successful and widely adopted approach is the use of ionizing radiation, which has been applied for radiography and computed tomography (CT) [203]. Here, the idea is to send ionizing radiation through the body and detect the residual signal for reconstructing a superimposed image of the body. The resulting 2D projections and 3D image volumes are used for diagnostic tasks such as lung screenings [485], pre-operative treatment planning [90], and interventional imaging, for example, for guiding percutaneous interventions [33]. CT imaging has also been augmented by positron emission tomography (PET), where a radioactive tracer is introduced to the body for highlighting processes within the body [229]. The imaging concept can provide fast, high-quality images, however, the ionizing radiation damages tissue. Therefore, other imaging techniques such as ultrasound (US) [230] using sound wave scattering and magnetic resonance imaging (MRI) [297] using magnetic excitation and resonance have been introduced. MRI can be employed for similar tasks as CT, including diagnostic tasks such as brain lesion detection [60], tumor visualization, and tracking [239], and cardiac imaging [53]. While CT is more advantageous for bone imaging, MRI is a promising approach for non-ionizing imaging in a lot of applications [245, 332, 569].

Typically, MRI comes with a spatial resolution of a few millimeters and a field of view (FOV) that covers several organs, depending on the device's acquisition parameters. For imaging smaller structures with a micrometer-level resolution, optical methods are popular, for example, microscopy [494]. When using infrared light instead of light in the visible spectrum, methods such as optical coherence tomography (OCT) can also image subsurface structures with a few millimeters penetration depth and without the use of damaging radiation [207]. OCT is frequently employed for diagnostic tasks in intravascular interventions [221] and ophthalmology [114], as well as, during surgical interventions in ophthalmic surgery [121]. As OCT scanning devices can be built in a space-efficient way, the imaging technique is a promising approach for a lot of diagnostic tasks and interventions, including lung tissue biopsy [337], skin lesion analysis [145], and visceral surgery [127]. Different imaging modalities, their penetration depth within tissue, and resolution are illustrated in Figure 1.1.

In general, medical images are analyzed by experts in their respective fields. This

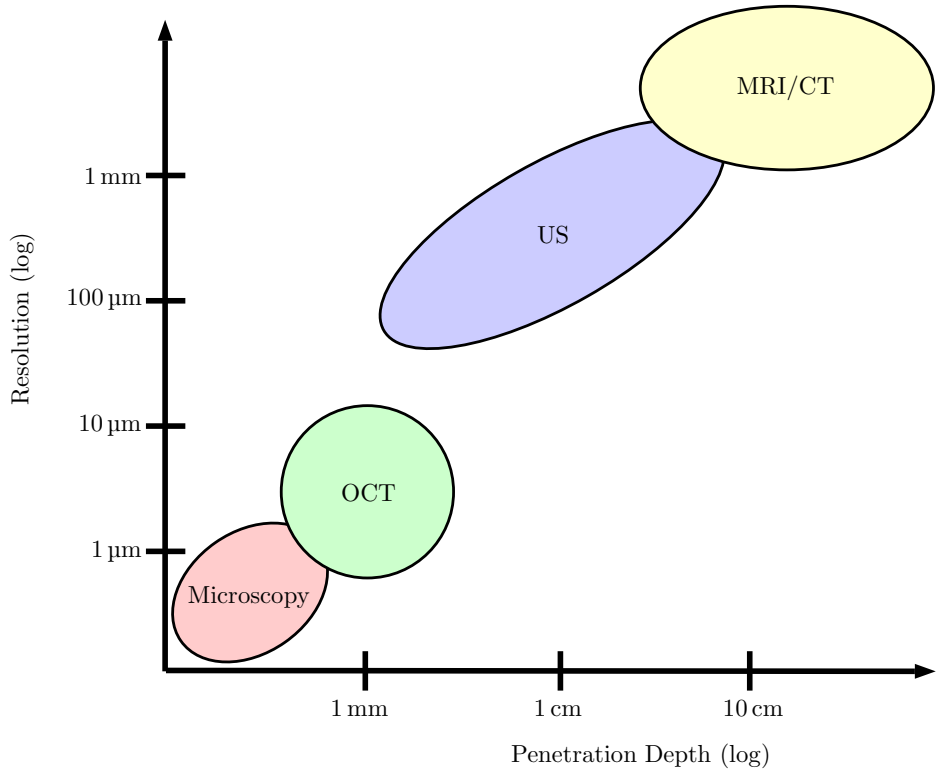


Fig. 1.1: An illustration of different imaging techniques, associated with their spatial resolution and typical imaging depth.

process is time-consuming, often associated with high inter- and intraobserver variability [128]. During interventions, intraoperative imaging can also be distracting and impose an additional burden on the surgeon [127]. Therefore, quantitative and automated medical image analysis has been pursued with the goal of extracting crucial, task-specific information from images for providing decision support to practitioners. For example, the field of radiomics is concerned with the extraction of quantitative parameters from medical images [267]. Also, for more standardized assessment of skin lesions, features such as border regularity, color, or streaks are often quantified based on medical images [196].

Traditional, fully automated, medical image analysis pipelines typically consist of an image pre-processing step, feature extraction, and machine learning-based feature processing for disease classification, structure segmentation, or regression of a quantitative measure [187]. While continuous progress in research on medical image analysis has lead to promising approaches, extensive clinical use has been lacking due to limited accuracy and consistency [78]. Recently, deep learning, an end-to-end machine learning approach, has lead to major breakthroughs and promises to reshape medical image analysis, medicine, and patient care [301].

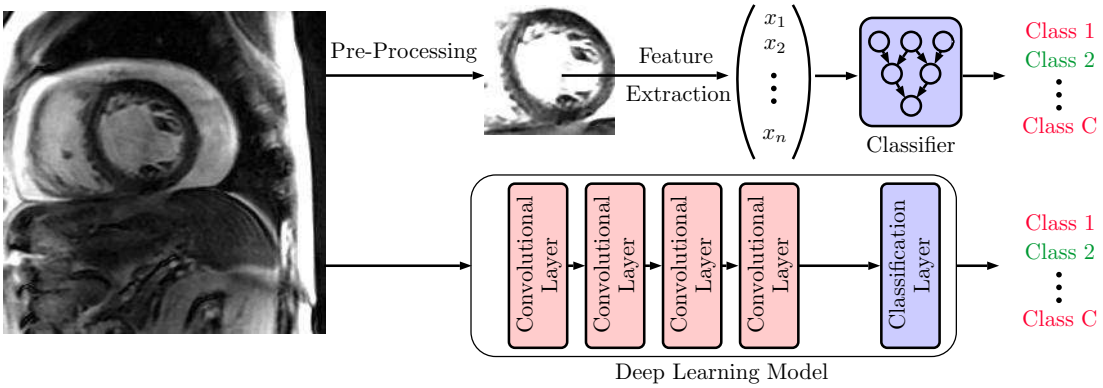


Fig. 1.2: Example of a conventional machine learning pipeline (top), including pre-processing, feature extraction, and a classifier compared to a deep learning model (bottom) where a convolutional neural network directly processes the image and outputs a prediction. A classification task with a cardiac MR image is illustrated.

1.2 Deep Learning and Medical Imaging

The traditional pipeline for automated image analysis has been challenged by end-to-end deep learning approaches, in particular, convolutional neural networks (CNN) [279]. While CNNs have been studied for decades, they remained in a niche role until the 2012 ILSVRC ImageNet classification challenge. Here, a CNN named AlexNet [262] was presented, which substantially outperformed all existing classical methods that relied on feature extraction and conventional machine learning. The key difference between the traditional pipeline and CNN-based image processing is illustrated in Figure 1.2. For the deep learning pipeline, only minimal pre-processing is applied before the images are fed into the model, which implicitly learns both feature extraction and classification or regression simultaneously.

Ever since, deep learning and CNNs have been used for a variety of computer vision problems, including methods for image classification [91], object detection [173], and semantic segmentation [312]. These techniques are applied to a variety of practical problems such as perception for autonomous driving [543], industrial applications [526], and surveillance [550]. Another major field of application is medical image analysis. There are hundreds of classification and segmentation problems for the different organs and diseases affecting the human body [301] as well as applications for computer-assisted interventions [236].

Lung imaging is a primary application of the common imaging modalities radiography and CT. CNNs have been frequently used to process 2D chest radiographs with the goal of nodule detection [311] and classification of different pathologies [446]. Similarly, lung nodule detection is a frequent task in deep learning-based CT image processing [503]. This is often combined with the task of classifying the detected nodules in terms of malignancy [206]. Also, the presence of interstitial lung disease is addressed using CNNs [18].

Another major part of deep learning applications is focused on the brain. Here,

most methods use MRI as the imaging modality. One application is the classification of disorders such as Alzheimer's disease [202], mild cognitive impairment [470], schizophrenia [585], and autism spectrum disorder [197]. Also, different anatomical structures are frequently segmented using deep learning methods, for example, white matter, gray matter, and cerebrospinal fluid [346], brain tumors [375], and lesions [63] have been segmented using CNNs. Other applications include longitudinal disease monitoring [240], survival prediction [359], and image reconstruction or image-to-image translation [591]. Initially, most deep learning approaches used 2D CNNs to process the MR image volumes slice-by-slice. This was partly motivated by the common slice-wise MR acquisition strategy and the discrepancy between inter- and intra-slice resolution. Also, computational and memory requirements increase significantly when processing 3D volumes. More recent approaches have also started using full volumes with 3D CNN processing [191].

Another frequent application is the eye. Here, color fundus imaging is often employed for segmenting anatomical structures [599] or the detection of diseases such as diabetic retinopathy (DR) [386] and age-related macular degeneration (AMD) [65]. The deep learning models are mostly 2D CNNs that have been adopted from the natural image domain. OCT is another imaging modality that is frequently applied in this area. Deep learning models can be used to classify DR and AMD [280] or to segment retinal layers directly [131]. Also, blood vessel segmentation in OCT angiography has been addressed with CNNs [125]. Similar to deep learning for brain MRI, most applications process OCT images slice-by-slice and full 3D approaches are still rare [164].

Cardiac imaging is performed with a more diverse set of imaging modalities. For the assessment of cardiac function, left ventricle quantification is a problem that is commonly addressed with US and MRI. 2D CNNs have been used for segmenting the left ventricle as a pre-processing step for quantification both with US [455] and MRI [384]. While CT was also used for left ventricle quantification [605], the imaging modality is more commonly used for coronary artery segmentation [542] and plaque deposit detection [542] in the context of coronary heart disease. For this problem, intravascular US [570] and more recently also intravascular OCT [4] are also employed. In particular, for the latter, deep learning methods are used for the detection of coronary stents [533] and arterial wall tissue classification [160]. As OCT images consist of individual depth profiles, deep learning methods have been applied to both 1D depth profiles and 2D cross-sectional OCT images [255]. Fully volumetric processing has been addressed for CT images but not for intravascular OCT data.

Other deep learning applications include musculoskeletal image analysis, for example, in terms of vertebrae localization and segmentation [80], and abdominal image analysis, for example, regarding kidney [488], liver [89], or prostate segmentation [575].

Besides medical image analysis, deep learning has also found application in computer-assisted interventions. While the field is generally smaller, there have been numerous successful applications in the context of intraoperative imaging. Here, deep learning methods can extract important information from the live images and provide immediate feedback to the surgeon. One application is the segmentation or detection of tissue structures in endoscopic videos, for example, for polyp detection [47]. Another deep learning application is the detection and segmentation of surgical tools in video data [424]. This can be used in the context of feedback during surgery or surgical skill

analysis [10]. Another recent application is vision-based force estimation [179]. As mechatronic force sensor integration into surgical setups can be difficult, vision-based force estimation has been studied using deep learning methods [27]. Most of these applications are designed for processing a real-time stream of 2D images or 3D volumes. Therefore, deep learning methods for computer-assisted interventions have to deal with the challenge of 2D, 3D, and 4D spatio-temporal data analysis.

Summarized, deep learning has found tremendous success in medical research. There is a vast amount of potential deep learning applications in the areas of medical image analysis and computer-assisted interventions. While there have been a lot of deep learning applications, there are still a lot of open image processing problems that could be improved by the use of deep learning methods. Also, for applications where deep learning has found initial success, there are still significant challenges that need to be addressed.

1.3 Research Questions

There are multiple ways of grouping work on deep learning for medical image analysis, for example, by their particular method, imaging modality, or anatomical region. Another aspect that is commonly neglected but takes a vital role in many applications is data dimensionality. Medical images are often volumetric, and many acquisition devices offer a temporal stream of images. As a result, many problems are multi-dimensional, with data representations ranging from 1D depth profiles to 4D spatio-temporal streams of volumes.

Previously, the most common problem regarding multi-dimensional data representations was the choice between 2D and 3D data processing. For visual assessment, medical images, even if volumetric, are often viewed in 2D, slice-by-slice. Most early deep learning approaches for popular imaging modalities such as MRI or CT have also used slice-by-slice processing [63, 593]. However, recently, more and more full 3D approaches emerged with the goal of exploiting inter-slice context in full 3D volumes. Many approaches have demonstrated that considering full 3D, inter-slice context is beneficial for CNNs [81, 141, 231, 301, 339, 376, 575].

Similarly, early machine learning approaches for OCT image analysis have processed individual 1D depth scans, usually focusing on tissue scattering patterns [134, 308]. While OCT's light scattering is often tissue-specific [557], 1D depth scans can be ambiguous and spatial relationships cannot be captured. Therefore, in recent years, machine learning methods have been increasingly used with 2D data representations that also capture spatial context [2, 131, 171, 282, 305, 397].

Thus, data dimensionality has started to play a significant role for several medical image analysis problems. Promising results presented for 2D and 3D MRI, or 1D and 2D OCT data open up questions for other data representations that have not been addressed so far.

For example, for MRI, there are other multi-dimensional aspects that have not been addressed frequently. Cardiac MR images are typically acquired as a sequence of 2D slices, covering an entire cardiac cycle. For function assessment, left ventricle quantification is required where parameters can be directly derived from individual 2D

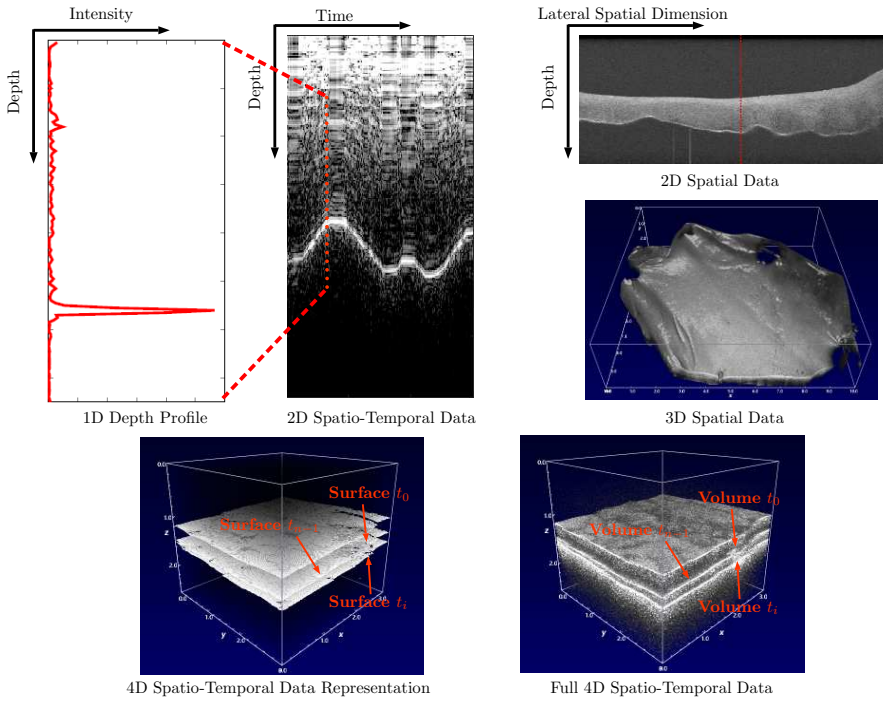


Fig. 1.3: Example for several OCT data representations, including a 1D depth profile, a 2D spatio-temporal series of depth profiles, a 2D cross-sectional image, a full 3D volume, and two 4D spatio-temporal data representations, shown as overlayed image volume renderings.

slices or they can be estimated considering the entire 3D spatio-temporal sequence [561]. This leads to the question of whether temporal context could allow for more consistent estimates.

Another open problem is longitudinal image analysis for tracking disease progression in the context of multiple sclerosis lesion activity [50]. Here, two 3D volumes or an entire 4D sequence of MR volumes needs to be analyzed to derive changes in the brain. Deep learning with 4D data has found few applications, and it is still largely considered an open research problem [88]. Thus, there is a need for more extensive analysis of different MRI data representations and their value for deep learning problems.

The imaging modality OCT also comes with a lot of different multi-dimensional data representations, see Figure 1.3. Fundamentally, OCT images consist of 1D depth profiles. Thus, in the context of tissue analysis, early approaches analyzed 1D intensity images to find patterns for tissue identification [496]. By acquiring multiple 1D depth profiles at neighboring locations, 2D images can be constructed, which are used in clinical practice for assessment of tissue layers [486]. Following the idea from CT and MRI applications that higher-dimensional context might be useful, 2D deep learning techniques have emerged for retina images [416] and intravascular images [3]. For needle insertion scenarios, OCT can also be employed where time series of 1D depth profiles are acquired. While individual 1D profiles have been processed with deep learning models [368], it is still unclear whether temporal context and thus 2D spatio-temporal data is useful. With more advanced scanning procedures, 3D image volumes can be

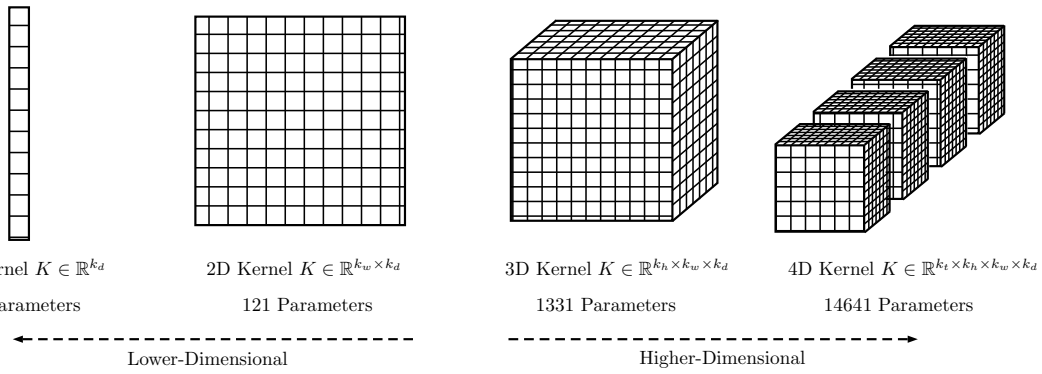


Fig. 1.4: An illustration of the curse of dimensionality and kernel dimensions. We show kernel sizes for different data dimensions where each dimension has a size of 11, as employed in the popular architecture AlexNet [262].

constructed. While finding application in intraoperative imaging [76], 3D deep learning-based processing has rarely been used. Thus, multi-dimensional OCT deep learning problems with an intraoperative context, including pose and motion estimation [276], as well as force estimation [367], remain open problems. Advancements in Mhz OCT devices have even enabled 4D spatio-temporal data generation [537], which could provide even richer context and has not been made use of so far. As a result, there are many opportunities for the use of multi-dimensional OCT data.

As a result, there are numerous multi-dimensional deep learning problems for MRI and OCT data that lack an analysis from a multi-dimensional perspective. When addressing such a problem, there is typically a choice between different data representations that can range from 1D to 4D data. This choice is accompanied by the design of a suitable deep learning method that deals effectively with the data structure.

Historically, deep learning model design for medical image analysis is heavily influenced by methods proposed in the natural image domain where deep learning for images originally emerged [301]. This becomes evident in the extensive use of 2D CNNs that have been originally designed for 2D natural images [451]. This has been largely successful for a lot of medical image analysis problems [301, 447]. Thus, from a multi-dimensional perspective, the question is how to extend common 2D deep learning methods to other data representations.

Moving to lower data dimensions, 1D data does not offer much context, but it comes with the advantage of being computationally cheap. Thus, 1D data is interesting in terms of real-time applications, for example, for computer-assisted interventions. Furthermore, deep learning methods that are computationally expensive otherwise might be easy to employ on 1D data.

Moving towards higher-dimensional deep learning models is particularly challenging as the curse of dimensionality becomes a significant problem. For a CNN, the number of trainable parameters increases exponentially, which leads to a severe risk of overfitting due to overparameterization, see Figure 1.4. Therefore, model design for higher-dimensional data requires a particular focus on efficiency in terms of the number of trainable parameters. At the same time, computational resources and memory usage become critical as processing high-dimensional data is very cost-intensive. Thus,

higher-dimensional deep learning model design is complicated, which might be one of the reasons why it has not gained more traction so far, despite promising results being reported [231]. For example, 3D CNN design has been previously referred to as "a nightmare" [327]. Overall, high-dimensional deep learning model design remains a challenging problem for medical image analysis.

Summarized, multi- and high-dimensional data are a promising opportunity to make use of relevant context. However, the optimal choice of data representations is often unclear, and deep learning model design is challenging, in particular for high-dimensional data. This results in two principal research questions addressed in this thesis:

1. Which data representations should be used for deep learning-based multi-dimensional medical image processing?
2. Which deep learning methods can be designed and used for processing specific data representations?

These research questions are very fundamental and broad in nature, requiring an analysis of different applications and imaging modalities to obtain general insights. Therefore, throughout this dissertation, we address these two research questions for the imaging modalities OCT and MRI in the context of several different applications. We propose multiple novel deep learning approaches and architectures for multi-dimensional image data. Our methods undergo extensive empirical evaluation in different application scenarios for addressing the two research questions in an application-specific context and for gaining a broader understanding and generalizable insights on multi-dimensional deep learning.

1.4 Primary Contributions

In this work, we propose several novel deep learning methods and apply them to various open multi-dimensional deep learning problems with regard to our two research questions. First, we adapt and introduce deep learning methods for different types of multi-dimensional medical image data. Second, we apply the adapted and our proposed methods to biomedical applications where data dimensionality is a key problem that has not been addressed so far. We perform extensive and systematic experiments to validate our adapted and proposed methods across multiple application scenarios.

1.4.1 Multi-Dimensional Deep Learning Methods

1D, 2D, 3D, and 4D CNNs. Convolutional neural networks are the most common method for machine learning-based image processing. Typically, they are applied to 2D images as it is a frequent practice in the natural image domain. As a first step, we design CNNs for 1D OCT image data and explore their capabilities in processing lower-dimensional data. As processing lower-dimensional data is cheap, we explore automized architecture search on 1D image data and the resulting architecture's transferability to higher dimensions. Moving to 2D data and 2D CNNs, we consider both the typical case of 2D spatial data processing as well as 2D spatio-temporal data processing where

convolution operations also process the temporal dimension. Next, we extend spatial 2D CNNs to 3D. Previously, there have been hardly any spatial 3D CNNs, in particular, none for regression problems and OCT data. Therefore, we adopt several concepts such as Inception [476], ResNet [193], and ResNeXt [553] to design 3D CNNs for 3D image data. As an alternative, we consider the immediate extension of existing 2D CNN architectures to 3D. We enable this approach by proposing a multi-dimensional transfer learning strategy with weight scaling for reusing 2D kernels in a 3D CNN. Finally, we also design 4D CNNs for processing 4D spatio-temporal data, a field that is mostly unexplored. We design several different 4D variants that perform efficient processing of the spatial and temporal data dimensions in different ways.

2.5D and 3.5D CNNs. There are many different learning problems where there are exactly two different states or representations that need to be processed. If each of these states is 2D or 3D in nature, we refer to the problem as 2.5D or 3.5D, respectively. In the natural image domain, a particular class of CNNs has been presented for this type of problem, called Siamese CNNs [508]. Here, the idea is to learn similar features for similar images, for example, for matching tasks. We adapt and extend this concept for biomedical learning problems with two 2D or 3D input images. We exploit image similarity with shared processing paths and explore how much parameter sharing or individual learning is beneficial with this type of architecture. Furthermore, we study the properties of feature fusion and propose a novel attention-guided interaction method for improved information exchange between the two paths.

Recurrent-Convolutional Models. Spatio-temporal processing can be performed using convolutions both for spatial and temporal data dimensions. Another class of methods that is suitable for temporal processing is named gated recurrent neural networks [200]. Here, temporal dependencies are learned in a recurrent fashion where relevant information within the sequence is preserved through gating and a state. Previous methods have used CNNs for extraction of feature vectors for each image in a spatio-temporal sequence, which is then processed by recurrent models [108]. We extend this approach by using convolutional gated recurrent units (cGRU) instead, followed by a CNN. Thus, instead of aggregating information from an abstract feature vector, we fuse local information in the initial spatio-temporal sequence while preserving the spatial data structure. We successfully apply this approach for 2D, 3D, and 4D deep learning problems, showing promising results. Based on this idea, we also propose an architecture with cGRU units between encoder and decoder for 4D segmentation problems. Here, we also aggregate temporal information using cGRUs while preserving spatial context for decoding into a segmentation map.

A selection of our proposed 4D deep learning methods is shown in Figure 1.5.

1.4.2 Multi-Dimensional Deep Learning Problems

All our adapted and proposed deep learning methods are tied to one or multiple biomedical applications. For each application, we study the effects of using data representations with different dimensionality. Here, we briefly describe the different problems and our respective insights, followed by our generalized insights.

OCT Fiber-Based Force Estimation. Precise placement of needles is a challenge in several clinical applications, such as brachytherapy or biopsy. Forces acting at the needle

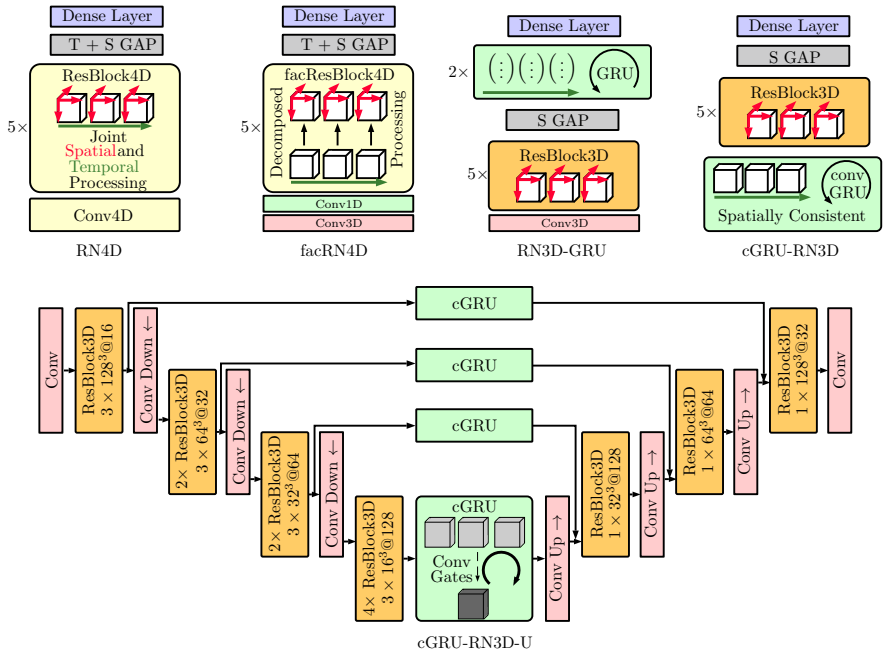


Fig. 1.5: Overview of our central method contribution to 4D deep learning: Top, we show several 4D deep learning architectures that we propose for regression problems, including 4D CNNs (RN4D, facRN4D) and recurrent-convolutional models (RN3D-GRU, cGRU-RN3D). Bottom, we show our cGRU-RN3D-U architecture for segmentation problems with 4D input data.

cause tissue deformation and needle deflection, which in turn may lead to misplacement or injury. Hence, many approaches to estimate the forces at the needle have been proposed. However, integrating sensors into the needle tip is challenging, and a careful calibration is required to obtain good force estimates. For this purpose, we propose a fiber-optical needle tip force sensor design using a single OCT fiber for measurement. The fiber images the deformation of an epoxy layer placed below the needle tip which results in a stream of 1D depth profiles. We study different deep learning approaches to facilitate calibration between this spatio-temporal image data and the related forces. For this application, we apply 1D and 2D CNNs, as well as convolutional-recurrent models, finding that the latter are most effective for the problem.

OCT-based Tissue Classification. A common tissue classification task is the segmentation of different layers in the human retina. Diseases such as diabetic retinopathy lead to the accumulation of fluids in between retina layers, requiring continuous monitoring of the retinal layer structure. Most approaches use CNNs for processing 2D cross-sectional images using custom, hand-crafted architectures [416]. We investigate whether improved 2D CNN architectures can be found with the concept of neural architecture search. As this method is computationally costly, we study whether searching for architectures in the space of 1D CNN models using depth profiles is effective. We demonstrate that architectures found on 1D data transfer well to higher-dimensional 2D data.

Similar to retina imaging, OCT tissue classification in coronary arteries is primarily performed using 2D cross-sectional images and 2D CNNs. Here, the goal is to detect

plaque deposits within the arterial walls in order to guide treatment decisions for preventing stenosis or rupture of vulnerable plaques. Here, we consider 2D Cartesian and 2D polar data representations for processing with a 2D CNN. Also, we extend the 2D data problem to 2.5D by combining the two data representations. The two representations are processed by one of our 2.5D Siamese CNN architectures. We find that Cartesian data representations appear to be preferable if enabled by data representation-specific data augmentation. Furthermore, we show that combining two data representations leads to improved performance.

MRI-Based Left Ventricle Quantification. Another 2D learning problem on tissue data is left ventricle quantification, where clinically relevant parameters are extracted from 2D cardiac MR images. Although the entire relevant anatomical structure is available in a single frame, neighboring temporal frames within the cardiac cycle might allow for more consistent estimates. In this context, we study the use of 2D spatial and 3D spatio-temporal CNNs. In particular, we employ our multi-dimensional transfer learning technique for immediate transfer of architectures to higher dimensions. Furthermore, we propose a segmentation-based regularization scheme to improve geometric left ventricle parameter estimation.

OCT-Based Pose and Motion Estimation. Another deep learning problem that is often addressed with 2D images and 2D projections is pose estimation. Here, the goal is to derive an object's pose from several images or 2D projections. We extend this problem to 3D using spatial 3D OCT volumes and several new 3D CNNs. We find that the additional volumetric information is more beneficial than simply using 2D projections, encoding 3D space. When performing tasks such as tracking or motion compensation, entire motion vectors need to be estimated instead of just individual poses. First, we address this problem in 3.5D with two 3D OCT volumes, processed by our Siamese CNN models. Second, we extend this approach to a full 4D problem, employing our proposed 4D CNN architectures.

OCT Volume-Based Force Estimation. Besides pose and motion estimation, force estimation is an important task for computer- and robot-assisted interventions. In contrast to needle-based force measurement, volume-based force estimation is performed with an external imaging modality. Here, we investigate the use of OCT as an imaging modality. Similar to OCT-based pose estimation, we first demonstrate that 3D volumetric data representations are preferable over 2D projections for deep learning-based estimation. Second, we extend this problem to full 4D spatio-temporal deep learning, finding that the use of 4D data is preferable. Moreover, we find that encoding lower-dimensional data representations in a higher-dimensional space improves force estimation performance. This indicates that higher-dimensional processing might often be preferable.

MRI-Based Multiple Sclerosis Lesion Activity Segmentation. 4D data is also relevant for the problem of longitudinal tracking of disease progression. We address this problem in the context of lesion activity segmentation, where the change in brain lesions between two MRI scans needs to be detected. First, we address this problem with our 3.5D Siamese CNNs using the two MRI scans as the model input. Here, we employ our attention-guided interaction modules for effective information exchange between the two states. Then, we extend the problem to 4D using our architecture with cGRU units between encoder and decoder that is depicted in Figure 1.5. We demonstrate that attention-based interaction modules improve performance and that they produce

interpretable attention maps. Also, we find that full 4D processing is beneficial in this application scenario.

Across all our applications, we find that our proposed cGRU-CNN-based approaches are effective for dealing with spatio-temporal data ranging from 2D to 4D. Similarly, our architecture with a cGRU between encoder and decoder shows promising results for 4D data processing in the context of segmentation problems. For CNNs, we explore different ways of architecture design, including adaptation, custom design, and automated learning. We find that each approach is viable in different contexts, as learning and adapting is more suitable for lower-dimensional data, and custom design is required for higher-dimensional data. For Siamese CNNs, we find that the extent of shared data processing is application-specific and that attention modules are useful for exchanging information between two states. Regarding the choice of data representations, we find that using higher-dimensional data is effective across all our applications. Our deep learning models effectively exploit the additional context and consistency provided by the additional data dimensions. Our insights confirm and extend the current trend of processing full 3D image volumes instead of just slices. In particular, our insights and proposed architectures for 4D deep learning appear to be promising for biomedical applications where high-dimensional image data is available.

1.5 Outline

This dissertation is structured as follows. In Chapter 2, we address medical image data representations and image processing basics. First, we introduce the two imaging modalities OCT and MRI and their different data representations. Second, we explain the basics of medical image processing that are relevant for image-based deep learning methods.

In Chapter 3, we introduce deep learning basics. We start with the fundamentals of statistical learning and the concept of generalization. Then, we describe neural networks and their extensions for images and sequential data. We explain gradient-based neural network training and regularization techniques for achieving a better deep learning model performance. We introduce recurrent and convolutional neural networks for processing temporal and spatial data.

In Chapter 4, we adapt and propose multiple deep learning approaches and models for multi-dimensional medical image data. First, we propose convolutional neural networks in the context of 1D, 2D, 3D, and 4D problems along with multi-dimensional transfer learning and neural architecture search. Second, we introduce new Siamese deep learning models for processing 2.5D and 3.5D data, including an attention-based mechanism. Third, we adapt and propose new recurrent-convolutional models. In this chapter, we describe abstract architectures, without a specific implementation.

In Chapter 5, we introduce the different application scenarios we consider. We outline their development in the scientific literature over the years and the use of different data representations. Then, we highlight open questions that have not been sufficiently studied in the literature. Finally, we consider work that is concerned with multi-dimensional deep learning in the context of applications that we do not consider in this thesis.

In Chapter 6, we present our experimental results for the different applications we

present in Chapter 5. For each application, we show implementations of our adapted and proposed architectures that we describe in Chapter 4. First, we consider lower-dimensional 1D and 2D deep learning in the context of force estimation and tissue classification. Second, we show results for 2D and 3D data representations for MRI-based left-ventricle quantification and OCT-based pose estimation. Third, we consider 3.5D data and the extension to full 4D data for OCT-based motion, longitudinal lesion activity segmentation, and OCT volume-based force estimation. For each application, we discuss problem-specific insights.

In Chapter 7, we discuss the findings for our applications in a broader context. We consider both the aspect of data dimensionality and deep learning methods for multi-dimensional data.

In Chapter 8, we summarize our key contributions, outline and discuss future work, and conclude the dissertation.

2 Medical Imaging Fundamentals

In this chapter, we describe relevant fundamentals of imaging and image processing for this dissertation. While deep learning methods have replaced a lot of conventional image processing techniques, many image processing concepts are part of a typical deep learning pipeline. First, we briefly introduce the imaging modalities and their data representations that we use throughout this thesis. Second, we outline basic image processing techniques and their application to medical images.

Over the years, numerous imaging modalities have been introduced to image the human body [67]. Physical principles such as ionizing radiation, sound wave scattering, magnetic resonance, and optical scattering are able to reveal different types of tissue and bones at varying scales and resolutions. Ionizing radiation is employed in imaging systems such as X-ray, CT [203], and PET [229]. While providing high-quality images and short acquisition times, ionizing radiation can be damaging. Therefore, imaging techniques such as US [230] and MRI [297] are also frequently employed. Also, for high-resolution surface imaging, optical methods using the visible light spectrum are applied, for example, for microscopy [494]. Other optical methods, such as OCT, rely on infrared light to also capture subsurface properties [207].

In this thesis, we focus on the two imaging modalities OCT and MRI. OCT can provide high-resolution images with a micrometer-level resolution, however, its field-of-view is usually limited to several millimeters. MRI, on the other hand, usually comes with millimeter-level resolution, and its field-of-view can cover multiple organs. Thus, OCT and MRI can be used for studying different types of spatial data dimensions. For each imaging technique, we describe the process of image formation that leads to the different data representations we employ.

2.1 Image Data Representations

2.1.1 Optical Coherence Tomography

Fundamentals. Optical coherence tomography is an imaging modality that makes use of low coherent near-infrared light to acquire depth information in a region of interest (ROI). It is based on the principle of interferometry and functions similar to a Michelson interferometer. A low-coherence light source emits a beam that is split up in two, such that a part of the beam hits a reference mirror, and the other part hits the ROI. Light is reflected back from the reference mirror and the ROI, interferes, and finally hits a detector. Inside the ROI, light is usually partially reflected at multiple surfaces along the beam's direction. When the reference arm's light and the reflected light from the ROI travel the same distance, positive interference occurs, which is characterized by a high intensity at the detector [136]. In order to acquire a full 1D depth image (A-Scan), the

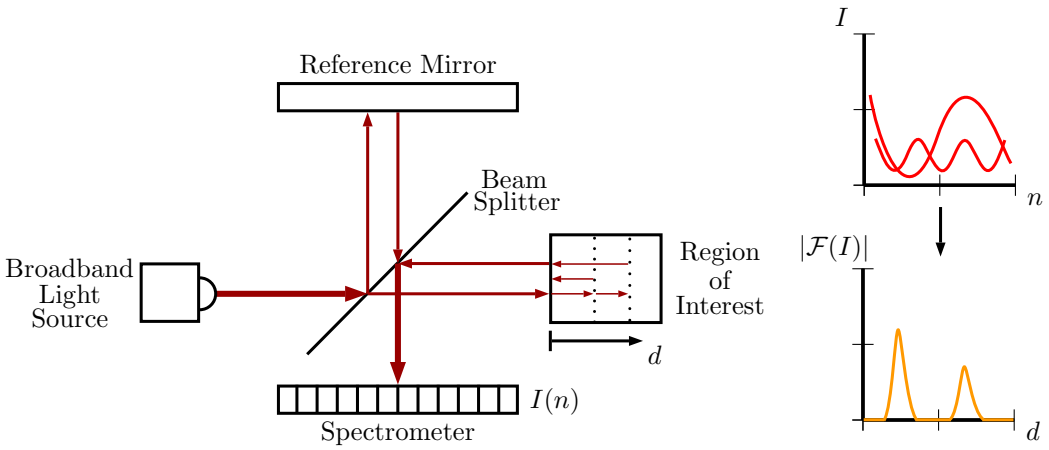


Fig. 2.1: Simplified depiction of the SD-OCT principle. Arrows represent light beams. Own elaboration based on [207]. The spectral image $f_S(n)(n)$ can be transformed to an intensity image $f_I(d)$.

reference arm length needs to be changed continuously in standard time-domain OCT (TD-OCT). A 2D scan can be acquired by repeatedly acquiring A-Scans at different lateral locations, resulting in a 2D image (B-Scan). Adding another lateral scanning direction leads to 3D volumes (C-Scans) [207]. Typically, lateral scanning is performed by using mirrors, redirecting the light beam. Finally, 4D image sequences can be acquired by repeatedly acquiring 3D volumes over time.

Besides TD-OCT, Fourier domain OCT (FD-OCT) has been proposed. One of its variants is spectral-domain OCT (SD-OCT) which recently gained popularity. This method does not require a moving reference arm, as a broadband light source is used to capture information at different depths. The recombined signal is captured by a spectrometer and a line scan camera. A Fourier transform directly results in a depth profile. This leads to much higher A-scan rates and an improved signal to noise ratio [285]. SD-OCT systems typically use a wavelength of $800\,\mathrm{nm}$ to $900\,\mathrm{nm}$, which allows for an imaging depth of a few millimeters [61]. A simplified setup for SD-OCT acquisition is shown in Figure 2.1.

Swept-Source OCT (SS-OCT) represents another FD-OCT method for OCT image acquisition. Here, the light source is a tunable swept laser that emits light at a single wavelength at each point in time. The laser sweeps across a large range of wavelengths over time, which are captured by a single photodetector. SS-OCT uses longer wavelengths above $1000\,\mathrm{nm}$, which is higher than SD-OCT and allows for deeper tissue imaging. Also, higher acquisition speed can be achieved easier with SS-OCT [76]. On the other hand, SS-OCT's spatial resolution is limited due to the limited amount of wavelengths that can be scanned [12].

Axial and Lateral Resolution. An important property of OCT is the independence of axial and lateral resolution. For FD-OCT, the axial resolution is, in principle, determined by the bandwidth and central wavelength of the light source. Assuming a Gaussian

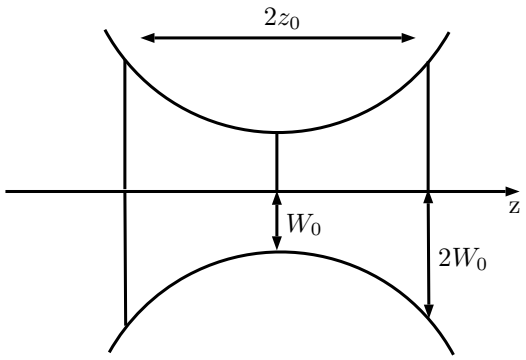


Fig. 2.2: Profile of the Gaussian beam, following [61]. The depth of focus is defined by $2z_0$ and the waist size W_0 determines the lateral resolution.

spectrum, the axial resolution is given by

$$\Delta l_a = \frac{2\log(2)\lambda_b^2}{\pi\Delta\lambda_b} \tag{2.1}$$

where λ_b is the central wavelength and $\Delta\lambda_b$ is the bandwidth [61]. In practice, OCT systems offer an axial resolution of up to $2\,\mu\mathrm{m}$ [12].

The imaging depth along the axial direction is limited by the light's penetration depth and the physical limitations of the detector. For example, for SD-OCT, the imaging depth is determined by the spectral range that can be captured by the spectrometer with a finite number of pixels. Following the Nyquist-Shannon theorem, the spectrometer's spatial grid should be spaced by half the axial resolution in order to avoid aliasing effects [61]. Assuming that the bandwidth $\Delta\lambda_b$ is sampled by N_λ pixels, the maximum imaging depth is given by

$$z_{max} = \frac{\lambda_b^2}{2\frac{\Delta\lambda_b}{N_\lambda}}\,. \tag{2.2}$$

The lateral resolution, on the other hand, depends on the light beam characteristics and the focused spot size. Typically, a light beam is modeled by a Gaussian beam model, which is defined by its waist size W_0, the Rayleigh range z_o, and radius of curvature [61]. A model of the beam's intensity profile is shown in Figure 2.2. The depth of focus is defined by $2z_0$, which resembles the axial depth with sufficient beam collimation. The beam's profile with respect to the axial direction has its minimum radius at W_0, which is also referred to as the spot size and determines the lateral resolution for this model.

Temporal Resolution. OCT system's temporal resolution is often measured by their A-Scan rate. Typical SD-OCT systems offer an A-Scan rate of $40\,\mathrm{kHz}$ to $100\,\mathrm{kHz}$ [12]. While these frequencies allow for fast A-Scan and B-Scan acquisition, real-time volumetric imaging is still difficult, especially if techniques like A-Scan averaging are applied to improve image quality. Thus, newer systems also offer A-Scan rates in the MHz range [252], which allow for fast volumetric imaging [449].

Speckle Noise. An important property of OCT systems is the phenomenon of speckle. Speckle results from random interferences of waves that are mutually coherent, which were reflected on or inside the ROI [433]. In OCT images, it appears as a granular

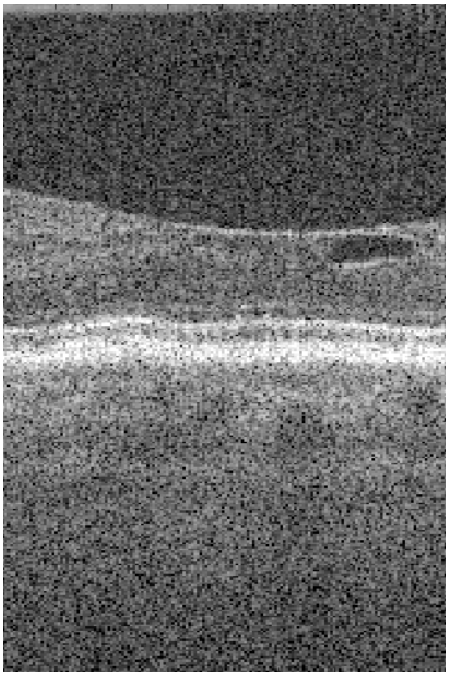


Fig. 2.3: B-Scan of the human eye [134] with granular speckle noise.

structure with no relation to the actual texture of the structure it was reflected from. An example is shown in Figure 2.3.

Speckle constitutes an issue to be considered as it reduces image quality. It is difficult to filter out, but some techniques such as averaging help reducing its negative influence [478]. For some applications, speckle can be useful as it remains almost constant for small movement in the OCT volume. It can be used for tracking tasks where frame to frame comparisons are made [167].

Multi-Dimensional Data Representations. Overall, OCT systems naturally provide 1D A-Scan data. For scanning at a target location, optical fibers can be used. Due to the fiber's small size, they can be integrated into small instruments such as needles to provide image guidance and information during an intervention [294]. Acquiring several 1D scans over time results in 2D spatio-temporal data, often also referred to as an M-Scan. Examples for A-Scan and B-Scan data are shown in Figure 2.4.

Alternatively, scan heads containing adjustable mirrors enable lateral scanning and thus spatial 2D and 3D image data acquisition. Example images for this type of data acquisition are shown in Figure 2.5. This type of data is typically used for imaging of the eye, for example, to detect and track disease progression of age-related macular degeneration, diabetic retinopathy, or retinal dystrophies [7]. By repeating the scan head's scan pattern over time, 3D spatio-temporal data (a sequence of B-Scans) or 4D spatio-temporal data (a sequence of C-Scans) can be acquired. Spatio-temporal OCT data is often used for angiography, which allows imaging of the retina's microvasculature [479].

Another method of data acquisition is employed for intravascular OCT (IVOCT), which is used to scan coronary arteries from the inside of a catheter [58]. Here, an OCT catheter is inserted into a target artery using a guiding catheter and a guidewire. A

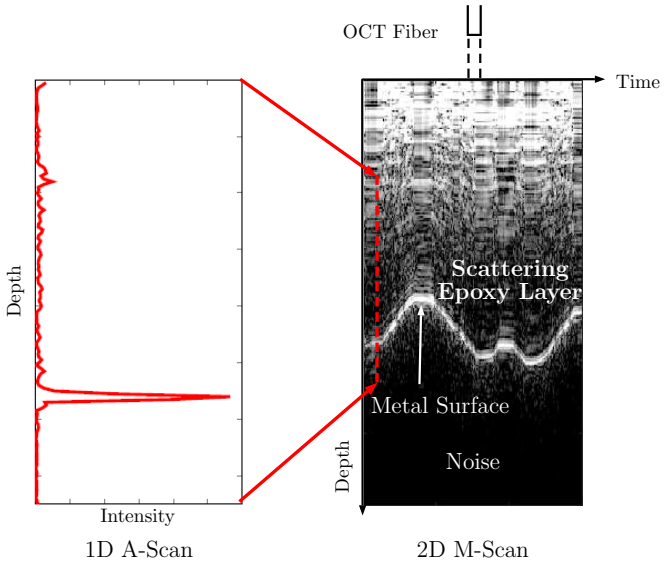


Fig. 2.4: Example for a 1D A-Scan (left) and a 2D M-Scan (right) consisting of multiple A-Scans. An OCT fiber images an epoxy layer which is capped by a metal layer. The epoxy layer is deformed over time. See Gessert et al. [162] for details.

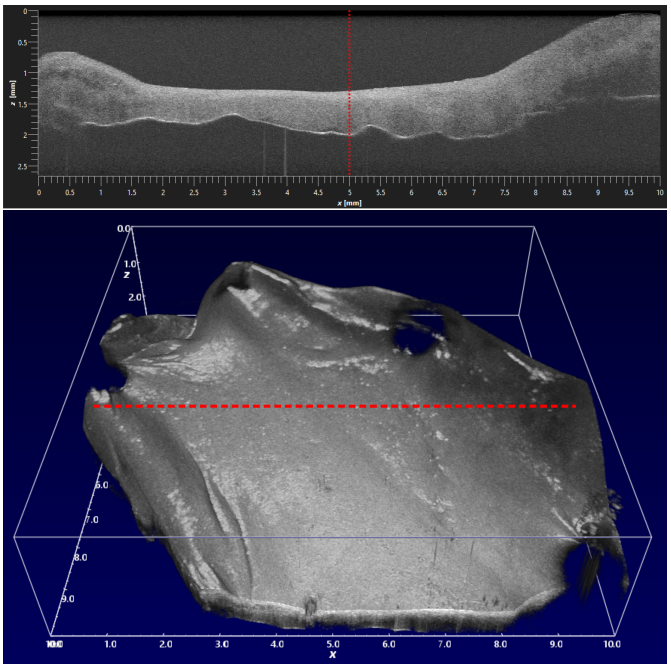
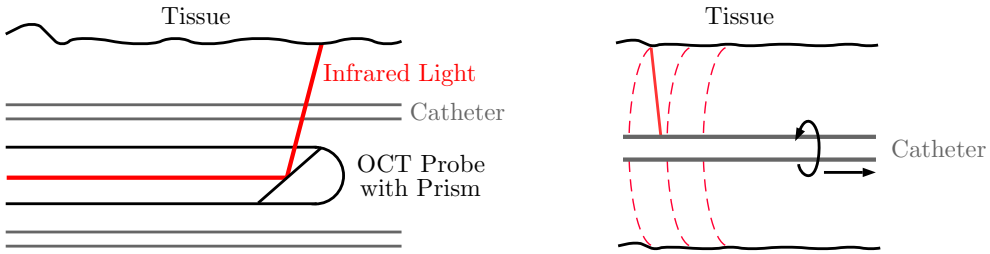


Fig. 2.5: Examples for a 2D B-Scan (top) and 3D C-Scan (bottom). In the B-Scan, the A-Scan direction is indicated. In the C-Scan, the B-Scan direction is indicated. The images show a pig's ex-vivo heart valve.



(a) Scanning Device (b) Scanning Pattern

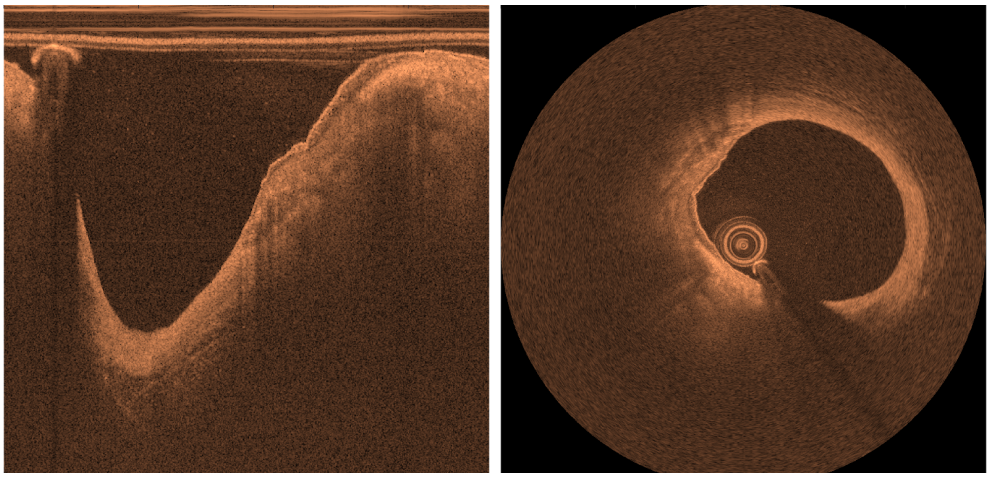
Fig. 2.6: The scanning setup for IVOCT data acquisition. A schematic drawing of a scanning device (a) and a cross section of the acquisition pattern (b) is shown. Due to simultaneous rotation and pullback the pattern in (b) results in a helix.

rotating probe with a prism close to the catheter's tip emits the infrared light and captures the back-scattered light. During acquisition, the probe rotates and is pulled back while continuously acquiring A-Scans. The acquisition process is depicted in Figure 2.6.

This scanning strategy results in a long M-Scan, however, the spatial locations of the A-Scan's point of acquisition form a helix pattern. Thus, image reconstruction is required for assigning A-Scans to their respective spatial locations. A simplified method of reconstruction is to cut the M-Scan into consecutive B-Scans. The cutting points can be determined based on rotation frequency and A-Scan acquisition frequency. The resulting B-scans are usually referred to as the polar representation, as the 2D image's coordinates are the imaging depth d and angle θ. Using a coordinate transform with $x = d\cos(\theta)$ and $y = d\sin(\theta)$, the polar images can be transformed into Cartesian space. This results in 2D cross-sectional image slices of the artery that are easier to interpret for a clinical practitioner. An entire 3D volume can be reconstructed by stacking the Cartesian 2D slices. All three data representations are shown in Figure 2.7.

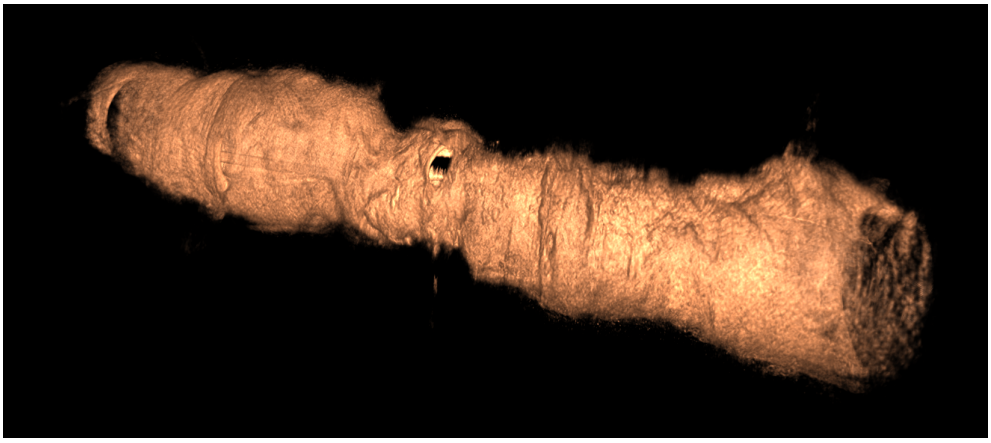
Note that this type of image reconstruction is a simplified approximation. When cutting the M-Scan into B-Scans, we assume that the rotation frequency is significantly faster than the pullback speed. In this way, we can disregard the helix pattern and assume that the starting point of a single rotation approximately coincides with the ending point. Also, when reconstructing the 3D volume, we assume that all scans are acquired along some center line. In reality, the catheter is subject to bending and movement, which is not reflected in this type of reconstruction. An additional external catheter tracking technique such as digital subtraction angiography (DSA) [99] or magnetic particle imaging (MPI) [272] can be used to obtain more accurate 3D volume reconstructions.

Summary. OCT allows for diverse applications with subsurface imaging on a micrometer scale with a centimeter-level FOV. Different scanning techniques and high acquisition frequencies lead to a variety of data representations ranging from 1D to 4D image data. There are different 2D data representations, including spatio-temporal data, cross-sectional B-Scans, and polar and Cartesian representations. While sharing the same underlying imaging principle, these data representations come with different properties and potentially different requirements for deep learning-based processing. A summary of all data representations is given in Table 2.1.



(a) Polar image representation.

(b) Cartesian image representation.



(c) Volume rendering.

Fig. 2.7: The different data representations of IVOCT images.

Tab. 2.1: Overview of the different OCT data representations.

Acquisition Type	Dimensionality	Typical Application
Single Fiber	1D Spatial 2D Spatio-Temporal	Surgical Guidance
Scan Head	2D Spatial 3D Spatial	Tissue Imaging
	3D Spatio-Temporal 4D Spatio-Temporal	Angiography
Catheter	2D Spatial (Polar) 2D Spatial (Cartesian) 3D Spatial	Intravascular Imaging

2.1.2 Magnetic Resonance Imaging

Fundamentals. MRI is an imaging modality that makes use of magnetic fields and radio waves to visualize the human body. The underlying physical principle of MRI is called nuclear magnetic resonance (NMR), which is based on the interaction of atomic nuclei with a magnetic moment [187]. Nuclei with an odd number of protons and neutrons exhibit a non-zero spin Q_s which is associated with a magnetic dipole moment that is given by

$$\mu_d = \gamma_g Q_s \tag{2.3}$$

where γ_g is the nucleus-specific gyromagnetic ratio. Typical nuclei for MR applications include hydrogen ^{1}H, carbon ^{13}C, sodium ^{23}Na and Phosphor ^{31}P. For MRI, the most important nucleus is hydrogen, which is very common in the human body, for example, in water and also larger molecules such as proteins and lipids. In addition, hydrogen's gyromagnetic ratio of of $26.752\,\mathrm{rad/Ts}$ is large compared to other nuclei.

Without any external magnetic field, hydrogen's spins are isotropically distributed. Once an external magnetic field is applied, the spins precess into two different orientations along the direction of the magnetic field, which are associated with different energy levels (Zeeman effect) [71]. For hydrogen, the two spin orientations can be parallel and antiparallel and their respective occupation numbers n_p and n_{ap}, for a steady-state, can be described with a Boltzmann distribution as

$$\frac{n_p}{n_{ap}} = \exp(-\frac{2|\mu_d||B_0|}{k_B T_B}) \tag{2.4}$$

where k_B is the Boltzmann constant, T_B the temperature, and B_0 describes the external magnetic field [452]. As the occupation numbers for both orientations are different, the vector sum of the nuclei's magnetic dipole moments results in a magnetization M, which is in parallel to the external field B_0. A stronger external field leads to a larger difference in occupation numbers and, thus, a larger, measurable magnetization M, which is why clinical MRI devices typically rely on powerful magnetic fields with $1.5\,\mathrm{T}$ to $3.0\,\mathrm{T}$.

The parallel spin orientation is referred to as the fundamental state with a lower energy level, and the antiparallel spin orientation is called the excited state with a higher energy level. The difference in energy levels is given by

$$\Delta E_l = \gamma \frac{h_p}{2\pi} |B_0| \tag{2.5}$$

where h_p is the Planck constant. Nuclear magnetic resonance occurs if the energy ΔE_l is added to the system. An excitation pulse with energy

$$E_{rad} = \frac{h_p}{2\pi} \omega_0 \tag{2.6}$$

leads to the transition from the fundamental state into the excited state. The excitation pulse is an alternating magnetic field, and its frequency ω_0 is chosen such that the resonance condition

$$\omega_0 = \gamma |B_0| \tag{2.7}$$

is met. The excitation pulse deflects the measurable magnetization vector M by an angle θ_E. Typical scenarios include a deflection by $90°$ with an α-Pulse and $180°$ with a β-Pulse. After deflection, the spins start to fall back into the initial steady-state, which is referred to as relaxation. In addition, the spins start to precess along the external magnetic field's orientation. The spins rotating in phase start to dephase, which leads to a second transversal relaxation process. The measurable magnetization along the external magnetic field's orientation M_z is referred to as the longitudinal magnetization. The measurable magnetization perpendicular to the external magnetic field's orientation M_{xy} is referred to as the transversal magnetization.

The longitudinal and transversal relaxation processes show different behavior in different tissue types, which allows for differentiation in the final MR images. Based on the superposition principle, the two processes' superposition is undisturbed. Longitudinal relaxation, which is also called $T1$-relaxation, refers to the process of M_z returning to its original steady state. During the process, energy is emitted, which is equivalent to the difference in the energy levels between the excited and the steady-state. For an α-Pulse, the longitudinal magnetization is defined as

$$M_z(t) = M_\infty(1 - \exp(-\frac{t}{T1})) \tag{2.8}$$

where $M_\infty = M(\infty)$ is the magnetization in steady-state and $T1$ is the relaxation time. The time $T1$ describes the time that is required for M_z to reach $63\,\%$ of M_∞. An example of two longitudinal relaxation processes is shown in Figure 2.8. The relaxation processes also illustrate how the intensity and contrast of MR images with different sampling time points can differ. When sampling at $t_s = 100\,\mathrm{ms}$, the first process shows a higher measurable magnetization. When sampling at $t_s = 500\,\mathrm{ms}$, the second process shows a higher magnetization and the difference between the processes is smaller.

The transversal relaxation process is also called $T2$-relaxation, which is the process of the rotating spin's dephasing after excitation with a pulse. In contrast to longitudinal relaxation, no energy is emitted during the process. After excitation with an α-Pulse, the spins rotate along the axis of the external magnetic field with the resonance frequency ω_0. Due to the interaction of the spin's local magnetic field, the spins start dephasing in the xy-plane, which is perpendicular to the external magnetic field's orientation. Finally, the spins are isotropically distributed in the xy-plane with a measurable magnetization of $M_{xy} = 0$. This process is described by

$$M_{xy} = M_0 \exp(-\frac{t}{T2}) \tag{2.9}$$

where $M_0 = M_{xy}(0)$ and $T2$ represents the transversal relaxation time. M_0 is equal to the magnetization M_z at the time of excitation. Example $T2$-relaxation processes are shown in Figure 2.9. Again, the intensity which is derived from magnetization is very different for different sampling time points. In biological tissue, $T2$ times are generally shorter than $T1$ times as the dephasing processes are much faster than the recovery of the longitudinal magnetization M_z.

In addition, $T2$-relaxation processes are often the sum of multiple exponential processes in biological tissue. This can be observed in fat tissue or at tissue borders if

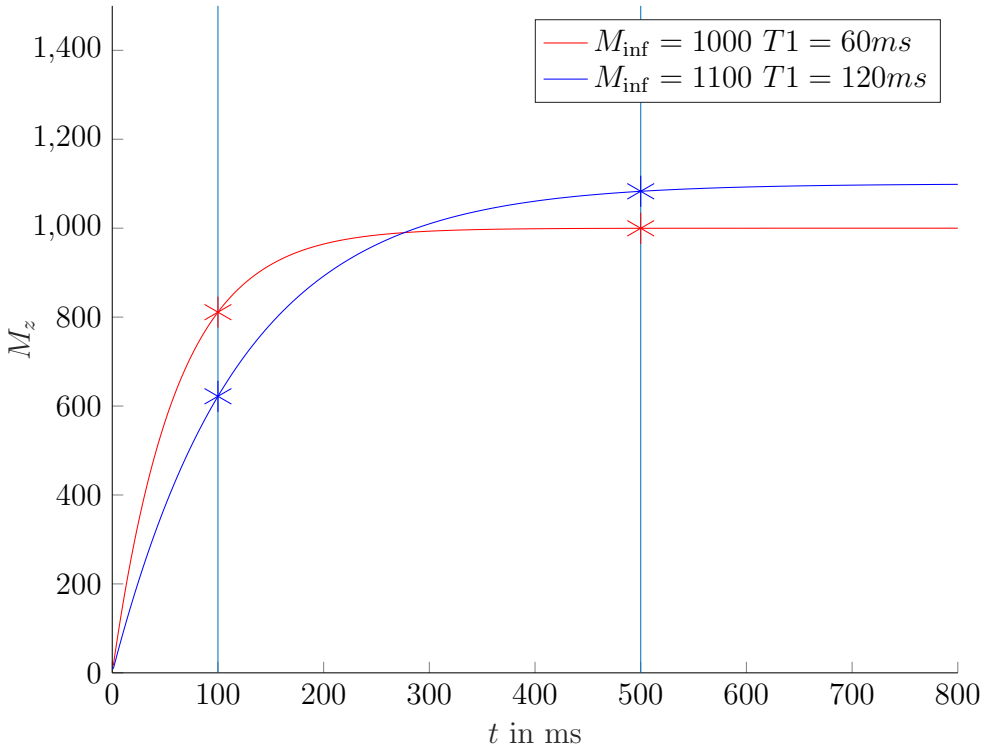


Fig. 2.8: Two examples of longitudinal relaxation processes are shown. Vertical lines represent sampling time points. Marks represent the sampling points.

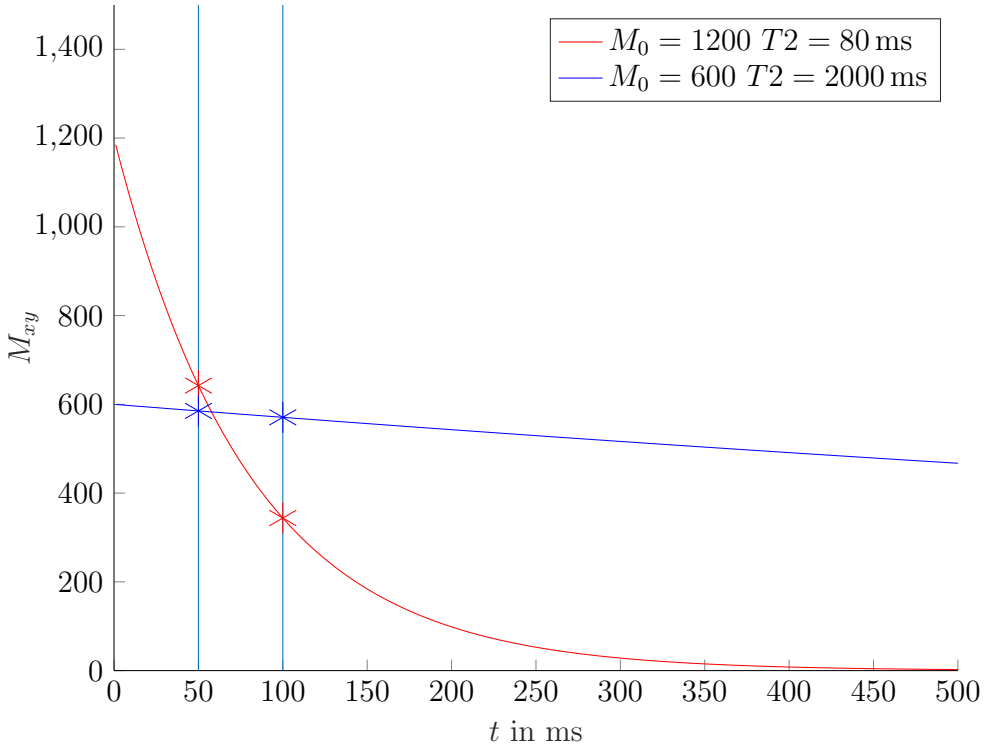


Fig. 2.9: Two examples of transversal relaxation processes are shown. Vertical lines represent sampling time points. Marks represent the sampling points.

multiple tissue types are present in a single voxel. This is also referred to as the partial volume effect. This type of process is defined as

$$M_{xy} = \sum_{i=1}^{k_{sup}} M_0^i \exp(-\frac{t}{T2^i}) \tag{2.10}$$

where k_{sup} is the number of superimposed processes.

Imaging Modalities. Based on the physical properties of relaxation processes, different modalities with different weighting can be produced. For this purpose, relaxation processes are induced inside the tissue, and the magnetization is measured at predefined echo times TE.

When using spin echo sequences, $T2$-relaxation processes are triggered. With a short TE time, proton density-weighted images are created whose intensity is largely governed by the proton density in the current volume element. With a longer TE time, measured signals are more affected by $T2$-relaxation characteristics, thus, the obtained images are called $T2$-weighted images.

Images governed by $T1$-relaxation processes can be obtained by using inversion recovery or saturation recovery sequences which induce an excitation angle of $\theta_E = 180°$ or $\theta_E = 90°$, respectively. Sampling with a short TE leads to $T1$-weighted MR images. For some applications, a gadolinium-based contrast agent is used to enhance, for example, active tumor borders of a glioblastoma.

To visualize and distinguish different tissue types in MR images, relaxometry is employed where tissue-specific parameters such as $T1$- and $T2$-relaxation times are determined. Afterward, by choosing a specific TE time and repetition time TR, certain tissue types can be enhanced or suppressed. The repetition time TR is the period to wait between excitation pulses. For example, fluid-attenuated inversion recovery (FLAIR) is a $T1$-weighted method where fluids such as cerebrospinal fluid are suppressed by choosing suitable times TE and TR.

2D and 3D Spatial Data. The following description is based on Sprawls et al. [462]. For image acquisition, target regions need to be specifically excited and spatially resolved. MRI acquisition is typically performed in a slice-wise manner. As a first step, a slice is selected in the human body using selective excitation. Here, gradient coils are used to target a specific slice. The gradient coils are an additional component in the MRI scanner that creates a gradient vector, which results in a change of the magnetic field along its orientation. Thus, along with the gradient vector's orientation, the magnetic field strength changes, and therefore, the resonance frequency ω_0 of the tissue inside the magnetic field changes as well. As a result, different ω_0 encode different spatial locations and can be specifically targeted by adjusting the excitation pulse frequency accordingly. Given a resonance frequency ω_0, associated with the main magnetic field B_0, the location-dependent resonance frequency is given by

$$\omega(x, y, z) = \omega_0 + \gamma(G_x x + G_y y + G_z z) \tag{2.11}$$

where G is the magnetic field strength of the gradient coils in each direction. Assuming slice selection along the z-axis, an excitation frequency ω_i shifts the slice position by $\frac{\omega_i - \omega}{\gamma G_z}$. Accordingly, slice thickness is determined by the excitation pulse's bandwidth with

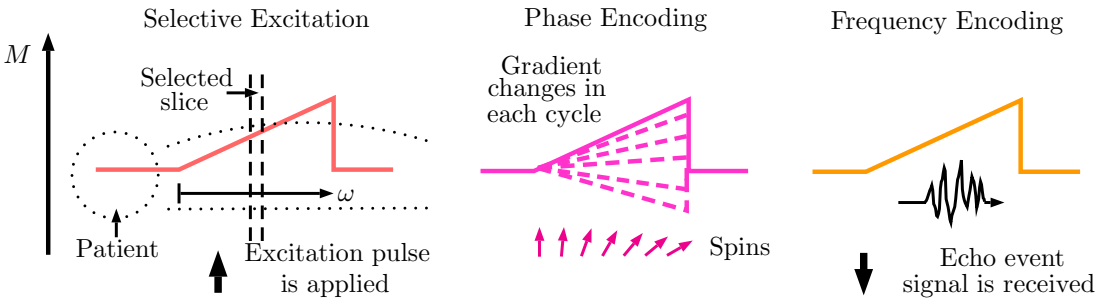


Fig. 2.10: A single MR imaging cycle is depicted. The three events happen in succession. For each imaging cycle, a different phase-encoding gradient is applied. M refers to magnetic field strength. Different colors imply magnetic field gradients along different spatial dimensions.

$\Delta z = \frac{\Delta\omega}{\gamma G_z}$. By adjusting the gradient vector's orientation, arbitrary slice orientations can be achieved.

After slice selection, the other gradient directions are used to resolve the slice spatially. For this purpose, phase and frequency encoding are used. After a selected slice is excited, a phase-encoding gradient is applied, which causes the magnetic spin along a spatial dimension to have different phases. Thus, the received signal will have components with different phases that can be associated with a spatial location. After phase encoding, the slice's second spatial dimension is encoded by frequency. Similar to the slice selection process, a gradient is applied along an additional direction. The change in the magnetic field leads to different resonance frequencies that can be associated with a spatial location. All steps of the acquisition process are shown in Figure 2.10. First, slice selection is performed by turning on a gradient for a short time. Now the spins are in an excited state. Then, the phase-encoding gradient is turned on for a short time period. Last, when the excited tissue emits the signal that is measured by the receiving coil, the frequency-encoding gradient is turned on. This imaging cycle produces a single row for the final 2D image. Thus, for a 2D image with N_{col} columns, the imaging cycle has to be repeated N_{col} times. For each repetition, the phase-encoding gradient is set differently to capture a different spatial location.

A full 3D volume is obtained by acquiring multiple slices. The process can be sped up by a technique called multi-slice imaging. Here, the next slice is already excited while the previous slice is still undergoing phase- and frequency-encoding. Another method for 3D volume acquisition is called 3D imaging. Instead of exciting a selected slice, the entire volume is excited without applying a selection gradient. Then, phase-encoding is used for slice selection. In this process, all three gradient directions are used instead of two for slice-wise acquisition with selective excitation. While 3D imaging requires longer acquisition times, phase-encoding-based slicing allows for thinner slices.

After acquisition, the MR image is encoded in a spatial frequency domain, also referred to as k-space. The final Cartesian image can be reconstructed with a 2D Fourier transform. Example 2D and 3D MR images are shown in Figure 2.11.

3D and 4D Short-Term Spatio-Temporal Data. An important drawback of MRI is its long acquisition time. Still, some imaging techniques allow for capturing short-term temporal processes. A typical method is cardiac cine MRI, which is used for imaging of

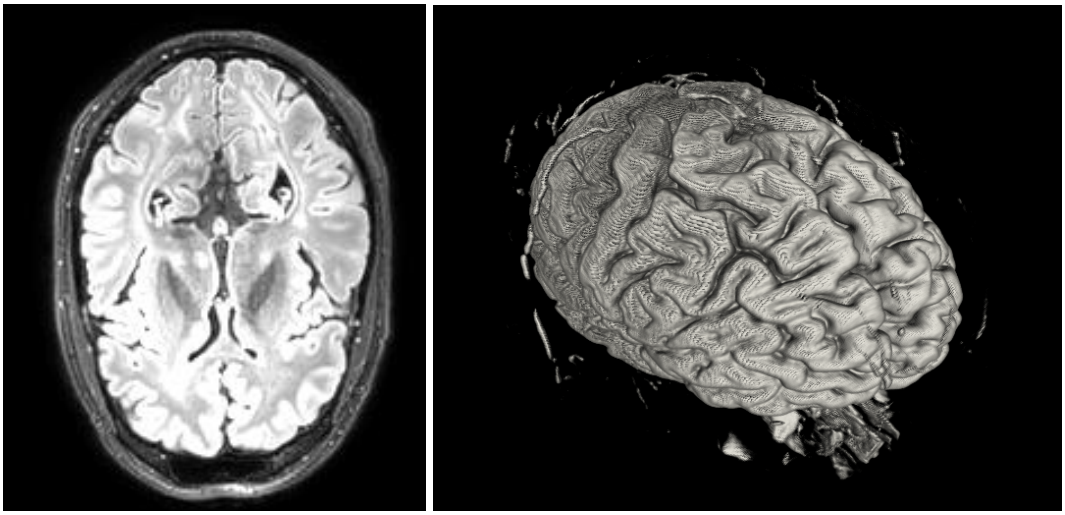


Fig. 2.11: An MRI scan of the human brain is shown. Left, a 2D slice is shown. Right, a 3D rendered volume is shown.

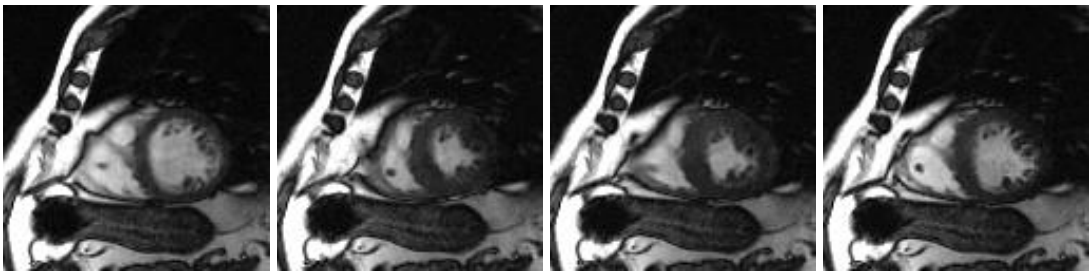


Fig. 2.12: An example of 3D spatio-temporal cardiac cine MRI data. From left to right, different phases of the cardiac cycle are imaged. Four images from a whole sequence of 20 images are shown.

the cardiac cycle in order to assess function and diagnose heart diseases. To enable faster acquisition, specialized gradient-echo techniques such as TrueFISP [75] are employed, which are characterized by very short TE and TR values. Still, the acquisition of a full slice during a phase of the cardiac cycle is infeasible as the imaging cycle needs to be repeated several times, see Figure 2.10. Therefore, electrocardiography (ECG) is used for retrospective gating. Both the MRI and ECG signal are acquired simultaneously over several cardiac cycles. After the acquisition, the MRI signals can be assigned to the respective cardiac phase using the ECG signal. The entire procedure can be completed within a breath-hold of the patient [23]. Often, this procedure is used to acquire a single slice over time, which results in 3D spatio-temporal data. If the entire heart needs to be imaged, the procedure needs to be repeated several times, which results in 4D spatio-temporal data. An example of 3D spatio-temporal cardiac cine MRI data is shown in Figure 2.12.

The same acquisition technique can be used for phase-contrast cine MRI which allows for imaging of blood flow in the human body [322]. The method relies on the fact that changes in the MRI signal phase along a magnetic gradient are proportional to blood flow velocity. When extending the method to 3D spatial data ("4D flow MRI"),

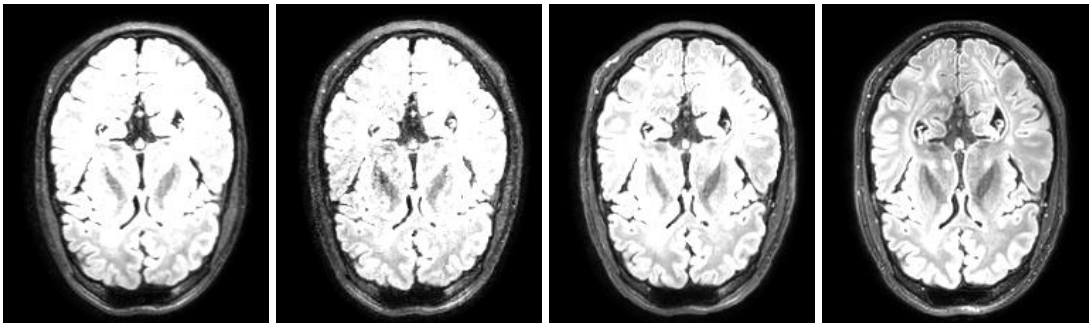


Fig. 2.13: An example of longitudinal 3D spatio-temporal MRI data of the human brain. From left to right, scans taken several months apart are shown.

Tab. 2.2: Overview of the different MRI data representations.

Acquisition Type	Dimensionality	Typical Application
2D/3D Imaging	2D Spatial 3D Spatial	Tissue Imaging
Cine MRI	3D Spatio-Temporal 4D Spatio-Temporal	Cardiac Imaging/Blood Flow
Longitudinal	2.5D Spatio-Temporal 3D Spatio-Temporal 3.5D Spatio-Temporal 4D Spatio-Temporal	Disease Tracking

acquisition becomes more difficult. Often, radial k-space acquisition is used, and due to long acquisition times, respiratory gating is additionally required [322].

3.5D and 4D Longitudinal Spatio-Temporal Data. Besides short-term temporal analysis, MRI is employed for long-term longitudinal analysis. A typical application is tracking disease progression for multiple sclerosis [60]. Here, individual MRI scans are acquired several months or years apart. In terms of the acquisition, there are no specific challenges, however, the individual time points can differ a lot due to different acquisition parameters, different scanners, or other major changes between the long time intervals. Often, only two volumes from two time points are compared to which we refer as longitudinal 3.5D spatio-temporal data. If more than two time points are part of the analysis, we refer to the data as longitudinal 4D spatio-temporal. If 2D slices instead of 3D volumes are used for longitudinal analysis, the data is referred to as 2.5D and 3D, respectively. An example for longitudinal 3D spatio-temporal data is shown in Figure 2.13.

Summary. MRI is an imaging technique with a lot of different medical applications, which are also often associated with different data representations. Typically, 2D slices are acquired, which can be combined to form 3D image volumes. Similar to OCT, spatio-temporal data is also a common occurrence. However, both short-term and longitudinal spatio-temporal data are used. While both are similar in structure, the meaning of the temporal dimension is very different, which might affect deep learning-based processing methods. An overview of the presented data representations is shown in Table 2.2.

2.2 Image Processing

In this section, we briefly address fundamentals of classic image processing that are relevant for this thesis. Image processing techniques are typically applied to 2D and 3D spatial image data or the spatial dimensions of a higher-dimensional data representation. After the acquisition, medical images typically undergo preprocessing for subsequent visualization and assessment by a medical expert or for further automatic processing. The goal of image preprocessing can be alignment by image registration or improvement of image quality in terms of contrast or the reduction of artifacts. In a classic, automatic processing and assessment pipeline, images are typically preprocessed first. Then, relevant features are extracted from the images with advanced processing techniques and finally, a machine learning model is trained with the features to provide an assessment of the image. Deep learning methods have replaced this pipeline with end-to-end models that directly process the images to provide an assessment. Still, a lot of deep learning applications benefit from classic image (pre-)processing. Also, many processing techniques are based on filtering, which is the basic operation of image-based deep learning methods. The following description is based on Handels et al. [187], unless indicated otherwise.

2.2.1 Point Operators

Point operators are one of the most simple types of image processing techniques, often also referred to as point processes [96]. Here, a function transforms every pixel of on image where every output pixel only depends on the corresponding input pixel and potentially some global information. In general, an image processing function m_I operates on a discrete intensity image function $f_I : \mathbb{R}^{N_d} \rightarrow \mathbb{R}$ with

$$g_I(x, y) = m_I(f_I(x, y)) \tag{2.12}$$

where $g_I(x, y)$ is the output image function, if we assume an image dimension of $N_d = 2$. Common point operators are addition and multiplication:

$$g_I(x, y) = a_p f_I(x, y) + b_p \tag{2.13}$$

where a_p and b_p are often referred to as control parameters for contrast and brightness. In the context of deep learning applications, the two point operators are often used for image normalization or standardization. For normalization, the image's intensities are scaled to a range of 0 to 1. For this purpose, we set $a_p = \frac{1}{\max(f_I)-\min(f_I)}$ and $b_p = -\frac{\min(f_I)}{\max(f_I)-\min(f_I)}$. Normalization can be useful to map images to a fixed range, for example, a color range for natural images or a distance range for depth images. When standardizing an image, its mean is mapped to 0 and its standard deviation is mapped to 1. Here, we set $a_p = \frac{1}{\sigma}$ and $b_p = -\frac{\mu}{\sigma}$ where σ is the standard deviation of f_I and μ is the mean value of f_I. For gradient-based neural network training it is often desirable to estimate σ and μ over all images in the dataset as it improves the optimization process [176].

Another application in the context of deep learning is the use point operations for data augmentation, a process that can be described as artificially generating additional image

data. Here, the values a_p and b_p are chosen randomly to generate new images $g_I(x, y)$ that represent slight variations of the original image $f_I(x, y)$.

Contrast and brightness of an image can also be improved by intensity histogram equalization. Here, the idea is to redistribute the intensities in an image such that the entire intensity range is covered evenly. First, we consider the intensity histogram h_I of an image f_I which is defined as

$$h_I(i_I) = \frac{N_{i_I}}{N_I} \tag{2.14}$$

where N_I is the total number of pixels, N_{i_I} is the number of pixels with discrete intensity i_I and $0 < i_I < i_{max}$. Next, we calculate the cumulative histogram c_I with:

$$c_I(i_I) = \sum_{j=0}^{i_I} h_I(i_I) \tag{2.15}$$

Now, for each pixel in the original image, given by intensity image function f_I, we can look up its new value $c_I(i_I)$ using the pixel's intensity i_I. The result is an image where areas of low intensity show increased intensity while areas of high intensity appear slightly darker, leading to an overall more balanced image. While this is mostly helpful for visualization, the method has also been applied as a preprocessing technique for deep learning methods with medical images [298].

2.2.2 Filtering and Local Operators

Local operators are a broad class of image processing techniques, that are characterized by considering local neighborhoods when processing images. Local operators make use of masks that are applied to each pixel location (x, y) in an image. The masks take n_m surrounding pixel locations into account in each image dimension in order to compute the pixel value at location (x, y) in the output image. The mask can be applied efficiently to each pixel location in the image using a convolution operation. Here, the output image's intensity function is computed by

$$g_I(x, y) = \sum_{i=-n_m}^{n_m} \sum_{j=-n_m}^{n_m} f_I(x - i, y - j) m_I(i, j) \tag{2.16}$$

where f_I is the input image's intensity function and $m_I \in \mathbb{R}^{(2n_m+1)\times(2n_m+1)}$ is the mask which also referred to as a convolution kernel. The process of a convolution can be pictured by the mask being slid over the input image while performing computations at each pixel location. This process is visualized in Figure 2.14.

Convolutions are problematic at the image border as there are no immediate, surrounding pixels available for computation. This can be circumvented, for example, by ignoring border pixels and only performing computations where the entire neighborhood is available. In this way, the output image f_I is of size $(n_h - 2n_m) \times (n_w - 2n_m)$ given an input image of size $n_h \times n_w$ with height n_h and width n_w. If the output image needs to have the same size as the input image, padding techniques can be applied. Padding can be performed by periodic repetition of the image or by adding zeros, which resembles using a reduced neighborhood for performing computations at the image border. Note

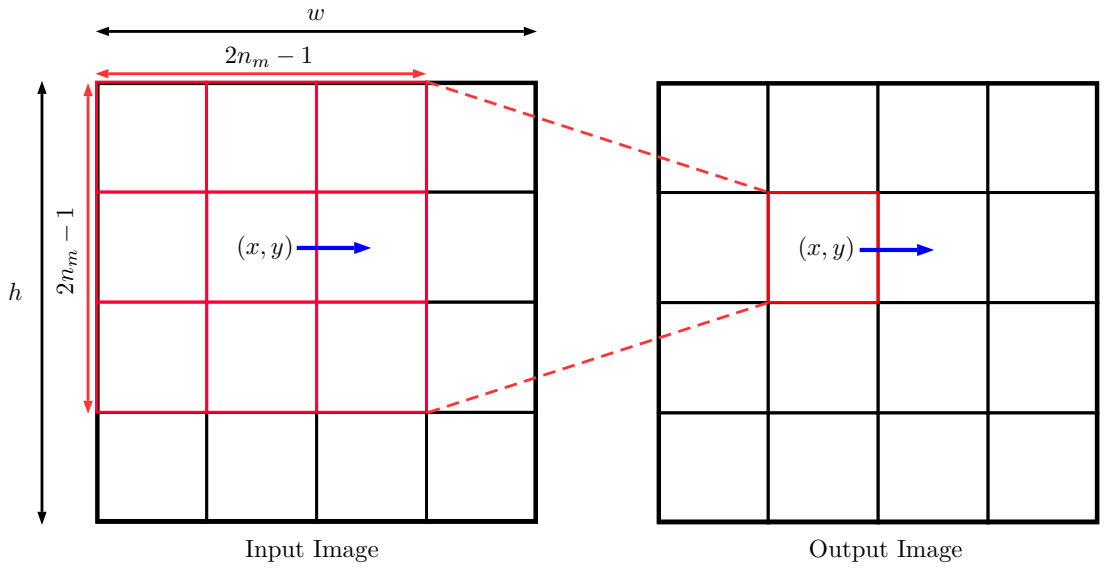


Fig. 2.14: Visualization of a convolution operation on a 2D image. The blue arrow indicates a shift of the convolution kernel on the input image which corresponds to a shift of the pixel location in the output image.

that convolution kernels can also by anisotropic by having a size of $(2n_m^1 - 1) \times (2n_m^2 - 1)$ with $n_m^1 \neq n_m^2$. While this is not typical for filtering applications such as smoothing and edge enhancement, filtering operations used in deep learning models can use different filter sizes for data dimensions with different properties.

Filtering of 2D images with convolution kernels can be easily extended to 3D spatial data with images of size $n_h \times n_w \times n_d$ with kernels of size $(2n_m - 1) \times (2n_m - 1) \times (2n_m - 1)$, where n_d is the image's depth dimension. Applying filter operations to higher-dimensional data, for example, 4D spatio-temporal data, is also feasible but uncommon for classic image processing. However, convolving over the temporal dimension is common in deep learning approaches.

Smoothing Filters. Typical, local filtering applications are image smoothing and noise suppression. Here, local variations in the image's pixel values are reduced by filtering. Smoothing is also called low-pass filtering and removes high-frequency components from the image. Typical filters for smoothing include the mean filter, the Gaussian filter, and the median filter.

The mean filter computes the mean value over the region covered by the filter's mask. The mask is defined as

$$m_M{}^{3\times 3} := \frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \tag{2.17}$$

for a mask of size 3×3. The reduction in local pixel value variation can be quantified by making the simplifying assumption that the distribution of values in the local region can be described by a normal distribution $\mathcal{N}(\mu, \sigma^2)$. In this case, the local pixel value variation is described by the standard deviation σ. After applying the mean filter, the pixel value distribution is given by $\mathcal{N}(\mu, \frac{\sigma^2}{v_m})$. Thus, the variation has been reduced by a

factor of $\sqrt{v_m}$ where $v_m = (2n_m + 1)^2$. This demonstrates that the level of smoothing depends on the kernel size n_m. A downside of mean filtering is its sensitivity to outliers, which can cause smears in the filtered image. Also, mean filtering dampens edges, which makes images appear less sharp.

The Gaussian filter tries to overcome the problem of dampened edges by relying on a 2D Gaussian distribution for the mask values for a 2D spatial image. The expected values μ_x and μ_y and the covariance values σ^2_{xy} and σ^2_{yx} are set to zero. The variances along x and y direction are both set to a value σ^2. Thus, the final distribution is given by

$$M_G(x, y) = \frac{1}{2\pi\sigma^2} \exp(-\frac{x^2 + y^2}{2\sigma^2}) \tag{2.18}$$

which can be approximated by a discrete binomial distribution. Thus, the final mask for Gaussian filtering is defined as

$${m_G}^{3\times3} := \frac{1}{16} \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix} \tag{2.19}$$

for a mask of size 3×3. This type of mask puts a higher weight on the center pixel value, which preserves more image information. In particular, edges are not dampened as strong as during mean filtering.

In contrast to mean and Gaussian filters, the median filter is a nonlinear filter that cannot be computed with a convolution operation. Here, at each pixel location, the median is computed over the mask region of size $(2n_m - 1) \times (2n_m - 1)$. Thus, the output image g_I is computed by

$$g_I(x, y) = \underset{(x_i,y_j)\in m_{med}(x,y)}{\text{Median}} \{f_I(x_i, y_j)\} \tag{2.20}$$

where $m_{med}(x, y)$ is the set of pixels in the region around coordinates (x, y). The operation sorts all pixel values within the mask and selects the middle value for the output image. In this way, outliers in terms of extreme pixel values are removed from the image. At the same time, in contrast to mean filtering, edges are largely preserved within the output image. This image processing method is particularly useful for reducing speckle noise in OCT images.

Edge Enhancement. Edge filters are used to enhance and highlight edges in 2D and 3D spatial images. For medical images, this can be useful for visualization or segmentation tasks where different image regions, for example, tumor tissue and healthy tissue, are separated by edges or borders. Within images, edges are characterized by large local changes in image intensity. Thus, the intensity gradient should show a local maximum around the edge. Assuming an intensity of $f_I(x, y)$ at pixel location (x, y) in a 2D spatial image, the gradient is given by

$$grad(f_I(x, y)) = \nabla f_I(x, y) = \begin{pmatrix} \frac{\partial f_I(x,y)}{\partial x} \\ \frac{\partial f_I(x,y)}{\partial y} \end{pmatrix} \tag{2.21}$$

assuming f_I is continuous. For an N_d-dimensional image, the gradient is a vector of length N_d pointing along the direction of steepest ascent. Therefore, the gradient vector is perpendicular to the edges.

The second derivative of $f_I(x, y)$ also provides information on edges as it approaches zero when the gradient reaches a local maximum. The second derivative is given by:

$$\nabla^2 f_I(x,y) = \begin{pmatrix} \frac{\partial^2 f_I(x,y)}{\partial x^2} \\ \frac{\partial^2 f_I(x,y)}{\partial y^2} \end{pmatrix} \tag{2.22}$$

For the detection of all edges in an image, filter masks need to be defined, which are convolved with the image. Since digital images are discrete, the gradient or second-order derivative of f_I needs to be replaced with a discrete approximation. One approach is to use a difference approximation for the gradient where only the neighboring pixel is considered. Here, the approximation is defined as

$$\frac{\partial f_I(x,y)}{\partial x} \approx \frac{f_I(x,y) - (f_I(x-1,y)}{x-(x-1)} = f_I(x,y) - f_I(x-1,y) \tag{2.23}$$

$$\frac{\partial f_I(x,y)}{\partial y} \approx f_I(x,y) - f_I(x,y-1) \tag{2.24}$$

for the gradient in x and y direction, respectively. This operation can be applied to every image pixel by convolving the image with a filter mask

$${m_{D_x}}^{3\times 3} := \begin{pmatrix} 0 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{2.25}$$

for obtaining gradients along the x direction. The difference operator m_D^y is defined analogously. By applying the filter masks independently to an image, gradient approximations along the x and y directions are obtained. For visualizing a gradient image, the pixel-wise gradient magnitude

$$|\nabla f_I(x,y)| = \sqrt{(\frac{\partial f_I(x,y)}{\partial x})^2 + (\frac{\partial f_I(x,y)}{\partial y})^2} \tag{2.26}$$

can be calculated and plotted as a gray-valued image.

A similar method for gradient approximation is using symmetric difference operators. For the x direction, the symmetric difference operator is defined as:

$${m_{SD_x}}^{3\times 3} := \begin{pmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \tag{2.27}$$

Compared to the normal difference operator, edges appear wider with this approach. However, thin edges with pixel-level width cannot be displayed properly. A disadvantage of both approaches is that not only edges but also noise is amplified.

The Prewitt and Sobel operators approach this problem by incorporating noise suppression based on mean and Gaussian filtering, respectively. Both operators are symmetric. For detecting edges along the verticle direction, the operator is defined as

$${m_{P_x}}^{3\times 3} := \frac{1}{6}\begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} \tag{2.28}$$

for the mean filtering-based approach (Prewitt) and

$$m_{S_x}{}^{3\times3} := \frac{1}{8}\begin{pmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{pmatrix} \tag{2.29}$$

for the Gaussian filtering-based approach (Sobel). The filters can be defined similarly for the horizontal or diagonal directions. For a filter that is independent of the direction, a combined filter can be used. For the Sobel operator, the combined filter is defined as $m_S := \max\{|m_S^x|, |m_S^y|, |m_S^{/}|, |m_S^{\backslash}|\}$ where the last two filters are diagonal filters.

Another approach for edge detection is to make use of the fact that the second derivative of the intensity function $f_I(x, y)$ approaches zero at edges. Assuming a continuous function f_I, we observe

$$|\nabla^2 f_I(x, y)| = \frac{\partial f_I(x, y)}{\partial x} + \frac{\partial f_I(x, y)}{\partial y} = 0 \tag{2.30}$$

in case of on edge in the image. This approach is utilized for the Laplace operator, where the second-order derivative is discretely approximated in a local region around the current image point. For the x direction, we obtain

$$\begin{aligned}\frac{\partial f_I(x, y)}{\partial x} &\approx (f_I(x + 1, y) - f_I(x, y)) - (f_I(x, y) - f_I(x - 1, y)) &(2.31)\\ &= f_I(x + 1, y) - 2f_I(x, y) + f_I(x - 1, y) &(2.32)\end{aligned}$$

when considering the immediate neighboring pixel locations. By considering more neighbors in the approximation, we can obtain a kernel

$$m_L{}^{3\times3} = \begin{pmatrix}-1 & -1 & -1\\ -1 & 8 & -1\\ -1 & -1 & -1\end{pmatrix} \tag{2.33}$$

for the size 3×3. When filtering an image with the Laplace operator, edges are represented by zero values. Thus, for visualization, the image has to be binarized first by assigning pixels with values close to zero to the value one while all other pixels are set to zero. Also, both edges and homogeneous areas within the image are close to zero after filtering with the Laplace operator. Thus, a lot of image points are detected which do not belong to any edges. Therefore, the Laplace operator is often employed in conjunction with other filters, which reduce noise beforehand. For example, a Gaussian filter [323] and a symmetric exponential filter [73] have been proposed for application before Laplace filtering.

Image Downsampling and Upsampling. Often, it is also desirable to change the resolution of an image. Since deep learning methods are often computationally expensive and require a lot of memory, using smaller input images is often required. For this purpose, downsampling methods based on filtering can be used. Conceptually, the image is first filtered with a low-pass filter for removing high frequency components (smoothing), which could cause aliasing effects otherwise. Then, the image is resampled by only selecting every rth pixel or voxel for the output image. This process can be

performed with a single *strided* convolution where the filter being slid over the image is only used for computing an output at every rth location. Here, the convolution is defined as

$$g_I(x,y) = \sum_{i=-n_m}^{n_m} \sum_{j=-n_m}^{n_m} f_I(rx-i, ry-j) m_I(i,j) \tag{2.34}$$

for a stride of r. Typical filters for downsampling include the Gaussian filter introduced above, a bilinear, or a bicubic filter. The bilinear and bicubic filters are based on linear and cubic interpolation. They consider a 2×2 and 4×4 neighborhood, respectively. Thus, the bicubic filter usually leads to a better smoothing effect and is therefore frequently employed. Trilinear and tricubic interpolation are the corresponding approaches for volumetric images.

Downsampling is also frequently applied inside of deep learning models for resizing images during processing, typically with a stride of $r = 2$. Instead of a handcrafted filter such as a Gaussian or bicubic filter, the filter's parameters are learned from data. In some cases, it is also desirable to upsample an image to a larger resolution. While this is often not desirable for the input images, some deep learning models upsample the images while processing them. For upsampling, the image is resized to the desired resolution, and the missing (new) pixels' values are filled by interpolation, for example, by convolution with an interpolating filter. Here, the same filtering concepts as for downsampling can be used, which includes bilinear and bicubic interpolation.

2.2.3 Image Alignment

Image alignment through registration algorithms is an important problem in medical image analysis. Often, several images of the same or several patients are aligned for visualization and direct comparison. For this purpose, different images with different coordinate systems need to be transformed into the same coordinate system. This can be performed for several scans of the same patient at different points in time, which allows for longitudinal comparison, for example, for monitoring disease progression. In the context of deep learning with a longitudinal sequence of images, alignment is also important as the underlying convolutional operations initially exploit local context. Thus, ensuring that local context matches between time points should improve the methods' performance. Another example is the alignment of images of the same patient acquired with different imaging techniques, often referred to as multi-modal image fusion. This can be relevant for the use of, for example, different MR imaging modalities in deep learning models. Also, registration between patients is relevant in the context of visualization and comparison. While alignment between patients can be helpful for deep learning models, it is also possible to force models to learn invariance towards incorrect patient alignment through data augmentation techniques.

During image registration, a fixed image $f_{I_F} : \mathbb{R}^{N_d} \to \mathbb{R}$ is aligned with a moving image $f_{I_M} : \mathbb{R}^{N'_d} \to \mathbb{R}$ using a transformation $t_{reg} : \mathbb{R}^{N_d} \to \mathbb{R}^{N'_d}$ with $N_d,\ N'_d \in \{2,3\}$. This is an optimization problem where t_{reg} is chosen such that $f_{I_M}(t_{reg}(x_c))$ becomes similar to $f_{I_F}(x)$ for all image coordinates x_c, measured by a similarity metric.

In general, registration algorithms can be categorized through three properties. The

first property is the image dimensionality. Typically, there are 2D-2D, 2D-3D, and 3D-3D registration methods. 2D-3D can be relevant for the alignment of preoperative 3D CT scans with intraoperative 2D US or X-ray images. 2D-2D and 3D-3D registration are relevant for longitudinal alignment or multi-modal registration. The second property is the type of information that is used for registration. Typically, landmarks, curves and surfaces, or voxels are used for registration. For landmarks, curves, and surfaces, prior detection or segmentation of the relevant features for registration is required. Third, registration methods can be categorized by being parametric or nonparametric. Parametric methods include rigid, affine, and perspective transforms, which perform a global transformation of the moving image. Nonparametric transforms can also consider local deformation in the images to be registered.

In the context of this thesis, rigid transformations are particularly relevant. Here, the moving image is rotated and translated to match the reference image. Thus, we assume that the transformed objects in the image are rigid. Here, the transformation $t_{reg} : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ transforms a coordinate vector $x_c = (x, y, z)^T$ with a rotation matrix $R_{rot} = (r_{ij})_{i,j=1,\ldots,3} \in \mathbb{R}^{3\times3}$ and a translation vector $s = (s_x, s_y, s_z)^T \in \mathbb{R}^3$:

$$t_{reg}(x_c) = Rx_c + s \tag{2.35}$$

The rotation matrix R_{rot} is defined by three rotation angles α_{rot}, β_{rot} and γ_{rot} which form the rotation matrix by:

$$\begin{aligned}
R_{rot} &= R_{\alpha_{rot}} R_{\beta_{rot}} R_{\gamma_{rot}} \text{ with} && (2.36)\\
R_{\alpha_{rot}} &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha_{rot} & -\sin\alpha_{rot} \\ 0 & \sin\alpha_{rot} & \cos\alpha_{rot} \end{pmatrix} && (2.37)\\
R_{\beta_{rot}} &= \begin{pmatrix} \cos\beta_{rot} & 0 & \sin\beta_{rot} \\ 0 & 1 & 0 \\ -\sin\beta_{rot} & 0 & \cos\beta_{rot} \end{pmatrix} && (2.38)\\
R_{\gamma_{rot}} &= \begin{pmatrix} \cos\gamma_{rot} & -\sin\gamma_{rot} & 0 \\ \sin\gamma_{rot} & \cos\gamma_{rot} & 0 \\ 0 & 0 & 1 \end{pmatrix} && (2.39)\\
& && (2.40)
\end{aligned}$$

Thus, the entire rigid transformation is defined by the six parameters α_{rot}, β_{rot}, γ_{rot}, s_x, s_y and s_z. An example application in the context of this thesis is the alignment of several longitudinal MRI scans for multiple sclerosis lesion activity segmentation, see Figure 2.15. Here, the images were taken several months apart, and in each case, different acquisition parameters such as slice orientation were chosen for acquisition. Thus, the images need to be rotated and shifted for proper alignment. Another example of rigid image registration is the task of motion tracking and compensation, for example, in the context of intraoperative motion compensation with OCT. Here, patient or surgical tool movement causes a shift or rotation. Image registration can be used to estimate the translation and rotation between and initial image and an image acquired after shift or rotation. In this thesis, this task is also addressed using deep learning methods.

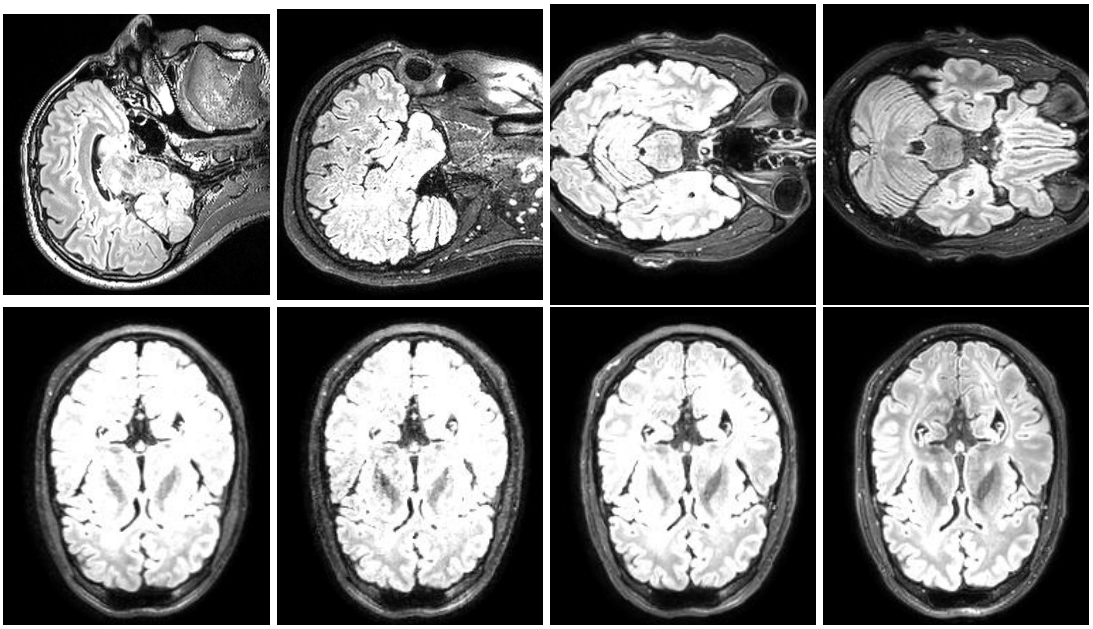


Fig. 2.15: An example of longitudinal 3D spatio-temporal MRI data, both unregistered (top) and registered using rigid registration (bottom). For the unregistered images, a slice along the slice acquisition orientation is shown.

Other parametric registration methods extend the rigid registration by additional degrees of freedom. For an affine transformation, shearing is also covered. The matrix $R \in \mathbb{R}^{3\times3}$ is no longer a pure rotation matrix, and all its nine entries are parameters, leading to a total of twelve parameters. For a perspective transform, perspective distortions are also considered, which leads to 15 parameters in total.

Nonparametric registration methods are useful if local deformation between the fixed and the moving image also need to be covered. This can be useful for intraoperative scenarios where soft tissue is being deformed and a preoperative scan needs to be registered to an intraoperative imaging modality. Also, the registration can be used to estimate motion fields from spatio-temporal image data or perform atlas-based image segmentation. For a nonparametric registration, the transformation t_{reg} is characterized by a displacement field $u_d : \mathbb{R}^{N_d} \rightarrow \mathbb{R}^{N_d}$ which shifts the original coordinate vector x_c:

$$t_{reg}(x_c) = x_c - u_d(x_c) \tag{2.41}$$

In order to find a plausible deformation field u_d, regularization is also required when formulating the optimization problem. Thus, the optimization problem can be defined as

$$\min D_{reg}[f_{I_F}, f_{I_M} \circ t_{reg}] + \lambda_{reg} Regl[t] \tag{2.42}$$

where D_{reg} is a distance metric $Regl$ is a regularization method. λ_{reg} is a weighting factor. The regularization method is chosen based on the specific problem. Regularization is also an important part of deep learning methods and is discussed further in Section 3.3.5. Note that registration itself is a problem that can be solved using deep learning methods. This problem is not addressed in this thesis. An overview of these techniques is given by Haskins et al. [189].

All registration methods are applied by solving an optimization problem where a distance metric D_{reg} and sometimes a regularization term $Regl$ are part of a loss function to be minimized. The distance metric D_{reg} depends on the information being used for registration. In the case of voxel-based registration, a typical example is the sum of squared intensity differences which is defined as

$$SSD_{reg} := \sum_{i=1}^{N} (f_{I_F}(x_c^i) - f_{I_M}(t_{reg}(x_c^i)))^2 \tag{2.43}$$

where N is the number of voxels in the image. This distance metric is useful if the two images' intensities are within the same range, for example, for monomodal registration of longitudinal MRI scans. This condition is likely not fulfilled, for example, in case of different imaging modalities being registered. For this purpose, mutual information can be used instead. The mutual information M_I between the moving and the fixed image is defined by the entropy H of the two images' intensity distribution and their joint intensity distribution

$$M_I = H_{f_{I_F}(x_c)} + H_{f_{I_M}(t_{reg}(x_c))} - H_{f_{I_F}(x_c), f_{I_M}(t_{reg}(x_c))} \tag{2.44}$$

where the images' entropy of the intensity distribution is, in general, given by

$$H_{i_I^1} = -\sum_{l=1}^{a_1} p(i_I^1(l)) \log(p(i_I^1(l))) \tag{2.45}$$

$$H_{i_I^1, i_I^2} = -\sum_{l=1}^{a_1} \sum_{j=1}^{a_2} p(i_I^1(l), i_I^2(j)) \log(p(i_I^1(l), i_I^2(j))) \tag{2.46}$$

where $i_I^1(l)$ and $i_I^2(j)$ are the pixel intensities with amplitude sets a_1 and a_2, respectively. The probabilities $p(i_I^1(l))$, $p(i_I^2(j))$ and $p(i_I^1(l), i_I^2(j))$ can be estimated based on the relative frequency of the intensities $i_I^1(l)$ and $i_I^2(j)$. In contrast to the sum of squared differences metric, the goal is to maximize mutual information which leads to more similar images.

Summary. Although deep learning methods perform end-to-end image processing, image processing and preprocessing techniques are still relevant. Point operators can be used for simple image transformations such as normalization. Convolution-based filtering is the fundamental principle behind most deep learning methods for images. Here, a kernel is swept over the image which reveals specific properties that depend on the type of kernel. In traditional image processing, kernels are defined for highlighting features such as edges or to smooth images. For preprocessing, medical images often have to be aligned where registration methods can be employed.

2.3 Summary

For imaging the human body, a lot of imaging modalities have been introduced, including CT [203], PET [229], US [230], MRI [297], microscopy [494] and OCT [207]. MRI is an imaging modality with a resolution in the millimeter range, allowing for imaging organs or whole body parts. OCT, on the other hand, uses infrared light for imaging smaller structures with a micrometer-level resolution. While covering different application scenarios, both imaging techniques share the property of multi-dimensionality. MRI is often acquired as 2D slices, forming 3D volumes when stacked. Also, temporal sequences, for example, for imaging the cardiac cycle or longitudinal analysis form 3D or 4D spatio-temporal data. OCT data acquisition can also be performed in very different ways. While a basic OCT image is a 1D depth profile, scanning mechanisms can enable 2D up to 4D data acquisition. Thus, both MRI and OCT come with many multi-dimensional data representations, opening up challenges and opportunities for deep learning-based processing.

Although deep learning methods represent an end-to-end approach that covers both image processing and feature extraction, image preprocessing still plays an important role. Simple point operators are useful for normalization or standardizing images before a deep learning model processes them. Local operators have been employed for conventional image processing, for example, for noise reduction or edge enhancement using handcrafted filter matrices. The concept can also be used for resizing images, which is often performed as a preprocessing step. In addition, filtering is the basis of image-based deep learning methods, which are introduced in the following section. The key difference is that the filter matrices are automatically determined based on statistical learning instead of handcrafting. Typically, the filters are also used for resizing the processed images inside the deep learning model. Before images are fed into a deep learning model, alignment can be necessary, for example, through rigid registration for longitudinal image analysis. Also, the concept of registration is used for pose or motion estimation in the context of intraoperative navigation, which is addressed by new deep learning approaches in this thesis.

3 Deep Learning Fundamentals

In this chapter, we introduce deep learning, a subfield of machine learning. First, we address fundamentals by explaining how deep learning models learn from data, the concept of generalization, and the associated problem of model design and optimization. Second, we introduce neural networks, the most common type of deep learning models. Third, we describe two variations of neural networks, namely convolutional neural networks for spatial data processing and recurrent neural networks for temporal data processing. Fourth, we outline how deep learning models are trained in practice, and we introduce regularization, a key concept for training neural networks. Last, we explain typical deep learning tasks, their model architecture requirements, and loss functions.

3.1 Problem Definition

Machine learning and deep learning algorithms are often described as algorithms that learn from data [15]. Formally, a deep learning algorithm can be described by a task T_M, performance P_M, and experience E_M. The algorithm learns from E_M with respect to T_M, such that P_M improves with experience E_M [343].

There are a variety of possible tasks T_M that can be addressed with deep learning. Some of the most common tasks are classification, regression, and segmentation. For a general deep learning problem, a function f_M needs to solve the task

$$f_M : \mathbb{R}^{d_1 \times d_2 \times \cdots \times d_n} \rightarrow \mathbb{R}^{\hat{d}_1 \times \hat{d}_2 \times \cdots \times \hat{d}_m} \tag{3.1}$$

where $d_i \in \{d_1, ..., d_n\}$ and $\hat{d}_i \in \{\hat{d}_1, ..., \hat{d}_m\}$ are spatial or temporal data dimensions. The function f_M is also referred to as a deep learning model. The model's input is $\mathrm{x} \in \mathbb{R}^{d_1 \times d_2 \times \cdots \times d_n}$ and the target is $\mathrm{y} \in \mathbb{R}^{\hat{d}_1 \times \hat{d}_2 \times \cdots \times \hat{d}_m}$. In practice, a deep learning model f_M approximates y with predictions $\hat{\mathrm{y}} = f_M(\mathrm{x})$.

Having defined the task T_M with Equation 3.1, we need a performance P_M to evaluate how well the algorithm solves the task. We obtain performance P_M with

$$P_M = J_P(\mathrm{y}, \hat{\mathrm{y}}) \tag{3.2}$$

where J_P is a performance measure that takes a model prediction $\hat{\mathrm{y}}$ and actual target value y as its inputs.

Next, the experience E_M is required for training a deep learning algorithm. Typically, deep learning algorithms experience a complete dataset $\mathcal{X}_{train} = \{\mathrm{x}_1, ..., \mathrm{x}_m\}$ with N_{train} examples. In this thesis, we consider supervised deep learning algorithms that experience datasets where examples are also paired with a target value y, leading to a dataset with tuples $\mathcal{X}_{train} = \{(\mathrm{x}_1, \mathrm{y}_1), ..., (\mathrm{x}_m, \mathrm{y}_m)\}$.

Based on these concepts, we can train a deep learning model f_M, parameterized by a set of adjustable weights w_M, to solve task T_M. We solve the optimization problem

$$w_M^* = \arg\min_{w_M} \frac{1}{N_{train}} \sum_{i=1}^{N_{train}} J_P(\mathrm{y}_i, f_M(\mathrm{x}_i, w_M, h_M)) \tag{3.3}$$

where h_M are hyperparameters of the deep learning model. We discuss their role and selection in the following section. Often, solving the optimization problem given in Equation 3.3 is referred to as model training. The next step is to define the model's real-world performance, which is referred to as generalization.

3.2 Generalization, Model Design, and Hyperparameters

The goal of a deep learning algorithm is to produce a model that performs well on inputs that were unobserved during training. This concept is usually referred to as generalization [324]. In the previous section, we referred to a single training dataset $\mathcal{X}_{train}$, which was experienced by the model through optimization. To assess generalization, we also need to consider a validation set $\mathcal{X}_{val}$ and test set $\mathcal{X}_{test}$ with $\mathcal{X}_{val} \nsubseteq \mathcal{X}_{train}$, $\mathcal{X}_{test} \nsubseteq \mathcal{X}_{train}$, and $\mathcal{X}_{val} \nsubseteq \mathcal{X}_{test}$. The validation set $\mathcal{X}_{val}$ is used to fit hyperparameters h_M. We assess the model's performance with respect to the test set $\mathcal{X}_{test}$, as it indicates how well the model did learn the task. The performance on the training set $\mathcal{X}_{train}$ reflects how well the algorithm fits the parameters w_M to the training dataset. While providing no indication about generalization, the training performance is relevant to observe the training process itself and ensure that the model optimization process leads to the desired goal.

In the medical image analysis domain, datasets are often limited in size. Splitting of a validation and test set, for example, based on random sampling, could induce a bias. By chance, the test set might be chosen in a way such that generalization performance is over- or underestimated. To overcome this problem, cross-validation (CV) has been proposed. In its most common form, k_{CV}-fold CV, the entire dataset is partitioned into k_{CV} subsets. Then, k_{CV} different deep learning models are trained where each uses $k_{CV} - 2$ subsets for training, and the remaining two subsets are used for validation and testing. If k_{CV} is the number of examples in the datasets, the method is referred to as leave-one-out CV.

The reason why generalization can be achieved is still part of current research [507] and not part of this thesis. In this thesis, we are concerned with deep learning model design for finding models that empirically generalize well for deep learning tasks. Designing deep learning models is the problem of finding a suitable set of hyperparameters h_M. The hyperparameters define the hypothesis space of the deep learning model. The hypothesis space is the set of all possible functions that are considered for solving the optimization problem given in Equation 3.3. The selection of h_M, and thus, the hypothesis space determines the *capacity* of a deep learning model. Capacity describes a deep learning model's ability to fit complex functions [176]. A model with low capacity is only able to fit simple functions, such as linear regression models. A model with high capacity is also able to fit complex nonlinear functions.

Formally, finding a suitable set of hyperparameters extends the optimization problem given in Equation 3.3 to the bilevel optimization problem:

$$h_M^* = \arg\min_{h_M} \frac{1}{N_{val}} \sum_{i=1}^{N_{val}} J_P^1(\mathrm{y}_i, f_M(\mathrm{x}_i^v, w_M, h_M)) \tag{3.4a}$$

$$\text{subject to } w_M \in \arg\min_{w_M} \frac{1}{N_{train}} \sum_{i=1}^{N_{train}} J_P^2(\mathrm{y}_i, f_M(\mathrm{x}_i^{tr}, w_M, h_M)) \tag{3.4b}$$

Here, we have $\mathrm{x}_i^v \in \mathcal{X}_{val}$ and $\mathrm{x}_i^{tr} \in \mathcal{X}_{train}$. Note that the upper and lower optimization problem in Equations 3.4a and 3.4b, respectively, can have different performance measures J_P^1 and J_P^2. Also, the upper optimization problem aims to minimize the error for the validation data, while the lower optimization problem minimizes the error for the training data. The test dataset is not part of the optimization problem to provide a performance estimate on unseen data.

Solving the upper optimization problem in Equations 3.4a is often infeasible due to high computational requirements. Nevertheless, there are several optimization methods that deal with the problem efficiently. A simple solution is to perform a grid search over a predefined subset of possible parameters $\hat{h}_M \subset h_M$. A downside of this approach is high computational effort if $\hat{h}_M$ is not chosen to be very small. However, prior knowledge on the parameter's behavior can be used to set up a reasonable subset $\hat{h}_M$ for the search.

When trying to reduce the necessary number of iterations for optimization, other techniques such as random search [46] and Bayesian optimization [456] have been introduced. In general, grid search is the most common method for hyperparameter optimization, as a lot of prior knowledge can be integrated when setting up the grid. As a result, choosing potential hyperparameters is the major engineering part that needs to be performed when employing deep learning methods.

If the choice of hyperparameters and thus the model's capacity is not optimal, the model often exhibits behavior that is referred to as underfitting and overfitting. Underfitting describes the issue when a model is not capable of fitting to the data, which results in a high training error, for example, when trying to use a linear model for a nonlinear task. Overfitting occurs when a model fits the training data perfectly, usually also including noise, leading to a large difference between training and test error and, overall, a large test error. Underfitting can occur for models with low capacity, while models with high capacity tend to overfit data.

Often, the terms bias and variance are used in conjunction with under- and overfitting. Bias is an error made by a model due to incorrect assumptions in the learning algorithm [254]. An example is the assumption of a linear relationship for a nonlinear problem. Variance is an error made by a model that also captures small variations such as random noise instead of the target relationship [254]. An example is a high fluctuation of predictions from a nonlinear model that was used for a simple linear problem. Therefore, underfitting with low capacity relates to a high bias problem with large deviations from the true value. Overfitting comes with a high variance problem as predictions for the test set are likely to deviate a lot. The relationship between training and test error, overfitting and underfitting, capacity, and bias and variance are shown in Figure 3.1.

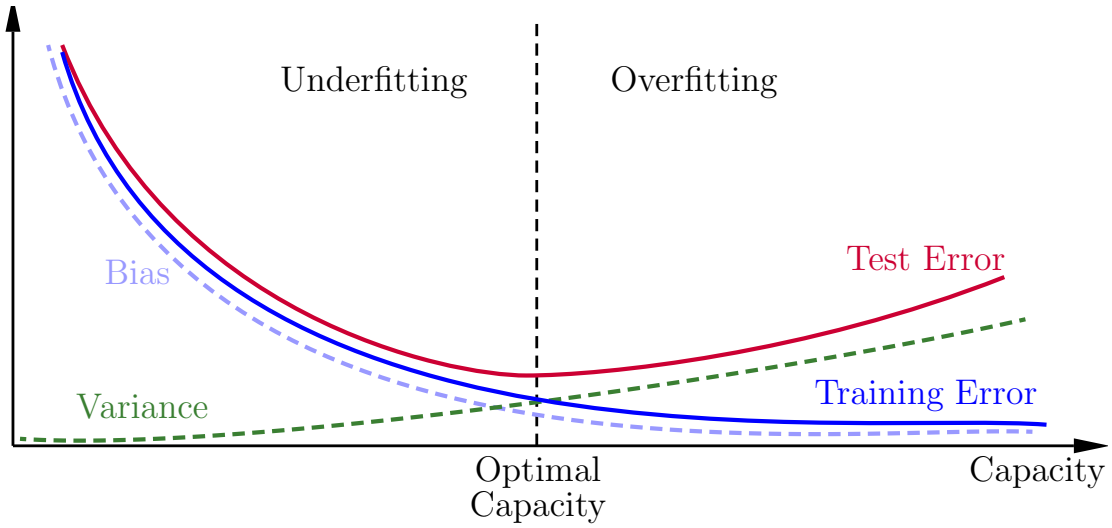


Fig. 3.1: The relationship between training and test (generalization) error, overfitting and underfitting, capacity, and bias and variance. With increased capacity, bias tends to decrease while variance increases. Up to the point of optimal capacity, both training and test error decrease. Here, the model's capacity is still in the underfitting zone. Once the optimal capacity is passed, the model enters the overfitting zone, and the test error begins to increase while the training error decreases further. Based on Goodfellow et al. [176].

Adjusting a model's capacity by choice of hyperparameters alone is often difficult in practice. Therefore, other methods, such as *regularization*, can be used to control a model's capacity. Here, a model's capacity is chosen to be large with a large hypothesis space, and regularization techniques are used to limit the model's tendency for overfitting. In a general model, a regularizer $\Omega(w_M)$ adds a penalty to a model's weight such that parameter selection of a training algorithm is handicapped. The optimization problem given in Equation 3.4 is therefore extended to

$$h_M^* = \arg\min_{h_M} \frac{1}{N_{val}} \sum_{i=1}^{N_{val}} J_P^1(\mathrm{y}_i, f_M(\mathrm{x}_i^v, w_M, h_M)) \tag{3.5a}$$

$$\text{subject to } w_M \in \arg\min_{w_M} \frac{1}{N_{train}} \sum_{i=1}^{N_{train}} J_P^2(\mathrm{y}_i, f_M(\mathrm{x}_i^{tr}, w_M, h_M)) + \lambda_w \Omega(w_M) \tag{3.5b}$$

where λ_w is a weighting parameter that trades off the regularization and the model's ability to fit the training data. Thus, instead of including or excluding functions from the hypothesis space, regularization more gradually forces a tendency towards a type of function. We introduce typical regularization methods in Section 3.3.5.

Summarized, a deep learning model's generalization capability is measured by an independent test set. For achieving good generalization performance, the model needs to be designed appropriately by selecting suitable hyperparameters h_M. Hyperparameters are chosen based on performance on a validation set that has no overlap with the training and test set. The selection of hyperparameters influences a model's capacity. If the

model's capacity is too low, a bias error can occur, and the model could underfit the training data. If the model's capacity is too high, a variance error can occur, and the model could overfit the training data. Regularization methods can also be employed to control a model's capacity.

3.3 Neural Networks

Neural networks first emerged in the 1950s [411]. Ever since, neural networks have undergone continuous development but remained a niche field due to limited generalization capabilities for practical problems [119]. Recently, neural networks have gained a lot of attention due to significant performance improvements that have come close to human-level performance [129, 341, 515].

In this section, we introduce different types of neural networks that form the basis of the deep learning methods that we employ in this thesis. We start with standard, fully-connected neural networks (FC-NN), which are typically used with feature vectors. Second, we introduce CNNs, which are neural networks for processing spatial image data. Third, we address recurrent neural networks (RNNs) that are frequently employed for processing temporal data.

3.3.1 Fully-Connected Neural Networks

Feedforward or fully-connected neural networks are a popular and versatile machine learning method. They form the basis for many advanced neural network methods such as convolutional or recurrent neural networks. An FC-NN consists of several artificial neurons, a mathematical model inspired by biological neurons in the brain [324]. The model has a number of weighted inputs that are summed up, and if the value surpasses a threshold, it is activated, also called firing. The firing is modeled by an activation function, such that a neuron's output is described as

$$o_{NN} = g_a(\mathrm{x}^\intercal w_M + b_M) \tag{3.6}$$

where g_a is the activation function. The vector $\mathrm{x} \in \mathbb{R}^{d_1}$ is an input feature vector, w_M and b_M are learnable model parameters, also referred to as weights and biases. A simple activation function g_a is a step function for representing the firing process. However, since neural networks are usually trained with gradient-based methods, other activation functions are used whose gradient is well-defined. Typically, sigmoid-shaped functions have been used instead. While sigmoid-shaped functions have a well-defined gradient around inputs with magnitudes close to zero, the gradient vanishes for inputs with a huge magnitude. Therefore, in modern FC-NNs, rectified linear units (ReLU) are preferred due to several advantageous properties [223]. The ReLU function is defined as $g_a(z) = \max\{0, z\}$. It is a nonlinear function but still piece-wise linear, which preserves some important properties of linear models, such as easy optimization with gradient-based methods [175]. Since this function always outputs 0 for $z \leq 0$, gradient-based optimization has no effect for inputs with values below zeros. Therefore, extensions to

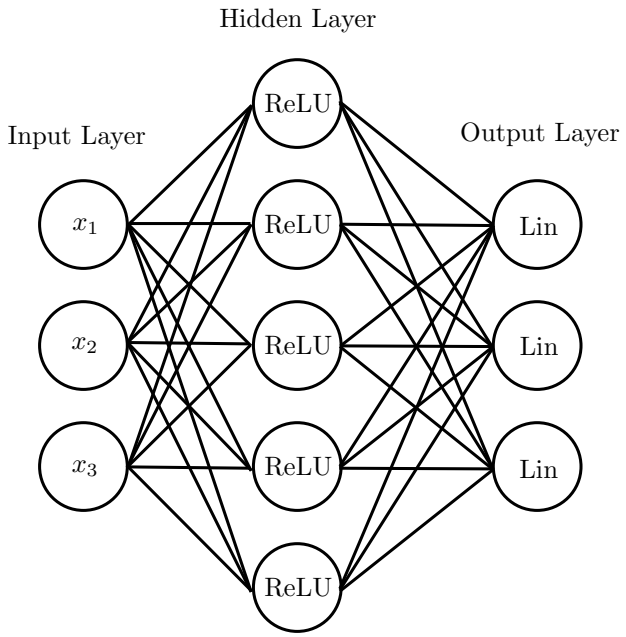


Fig. 3.2: A neural network with an input layer x one hidden layer with activations of type ReLU and an output layer with a linear activation function. Black lines represent learnable weights.

ReLU were introduced for this case by introducing a variable α_{ReLU}, such that

$$g_a(z) = \max\{0, z\} + \alpha_{ReLU} \min\{0, z\}. \tag{3.7}$$

Parametric ReLU is an example where α_{ReLU} is considered to be a trainable parameter [192], which promises to improve performance without a significantly higher computational cost.

Multiple, stacked, artificial neurons form a single-layer neural network. In case the activation function is a step function, it is also called a perceptron. Due to the disadvantages outlined above, perceptrons are rarely used, and neural networks with other activation functions are often preferred. A neural network now computes

$$o_{NN} = g_a(w_M \mathrm{x} + b_M) \tag{3.8}$$

where $\mathrm{x} \in \mathbb{R}^{d_1}$ is the input feature vector, $w_M \in \mathbb{R}^{d_{NN} \times d_1}$ is a weight matrix and $b_M \in \mathbb{R}^{d_{NN}}$ is a bias vector. The output $o_{NN} \in \mathbb{R}^{d_{NN}}$ is now a vector instead of scalar. The number of neurons d_{NN} is also called the width of the neural network. According to the universal function approximation theorem, such a network is capable of modeling a large set of functions [201]. This implies that any machine learning task should be solvable using such a network, however, it is not guaranteed that we find that function when fitting the model to data.

The neural network can be extended further by adding more layers. Assume that Equation 3.8 represents a single layer f_M^1. Adding layers corresponds to computing the function $o_{NN} = f_M^3(f_M^2(f_M^1(\mathrm{x})))$ for $l_{max} = 3$ layers. Often, adding more layers significantly improves the network's performance [177]. Layers between the input and

the output are referred to as hidden layers. If f_M^3 is the neural network's last layer, o_{NN} corresponds to the model output $\hat{y}$. An FC-NN with one hidden layer is shown in Figure 3.2. When more than one hidden layer is present in the model, it is referred to as a *deep* learning model.

3.3.2 Convolutional Neural Networks

CNNs are a modified version of FC-NNs that are specifically designed for processing image data $\mathrm{x} \in \mathbb{R}^{d_1 \times d_2 \times \cdots \times d_n}$ instead of feature vectors $\mathrm{x} \in \mathbb{R}^{d_1}$. CNNs gained initial attention for digit recognition in 1998 [279]. With a quick increase in computing power over the last decade, training large scale CNNs became feasible, and, in 2012 Alex Krizhevsky popularized deep learning with outstanding results at the ImageNet classification competition [262]. Here we explain the characteristics of convolutional layers which make deep learning models very effective for image processing.

Moving from fully-connected layers, as given in Equation 3.8, to convolutional layers is motivated by the concepts of sparse connectivity and parameter sharing [176]. Classic NNs contain several neurons in a layer where each neuron is connected to every input with individual weights. Considering a 2D image or 3D volume input, this leads to millions of weights per layer and thus an increased risk of overfitting, as described in the previous section. To overcome this issue, neurons are reduced to only have a smaller receptive field $v \in \mathbb{R}^{k_1 \times k_2 \times \cdots \times k_n}$ with $k_i < d_i$. Hence, they are only connected to a small portion of the input.

A next step is to enforce parameter sharing among all neurons in a layer. Thus, the same parameters $K \in \mathbb{R}^{k_1 \times k_2 \times \cdots \times k_n}$ are used for every receptive field v. This reduces the number of parameters even further. The motivation for this is our goal to learn similar features in every region of the input, for example, the detection of edges in an image. As a result, we obtain a kernel K instead of a weight matrix w_M. Before, the layer's operation was a matrix multiplication of the inputs with the weight matrix.

Now, we can picture the operation as the kernel being slid over the input which resembles the mathematical operation of a discrete convolution which was introduced in Section 2.2.2. As a result, the convolutional layer is closely related to classic image processing with local operators. We introduced several different kernels for tasks such as smoothing or edge detection. Here, we learn the kernels from data instead. Thus, there is no need to handcraft kernels based on, potentially flawed, assumptions about which properties in an image are important for a learning problem.

In the general case, a convolutional layer l receives a tensor $\mathrm{x}^{l-1} \in \mathbb{R}^{d_1 \times d_2 \times \cdots \times d_n \times n_c^{l-1}}$ from a previous convolutional layer. Here, n_c is the number of input channels. In case of the first convolutional layer of a CNN and an RGB image, $n_c^1 = 3$. The kernel $K \in \mathbb{R}^{k_1 \times k_2 \times \cdots \times k_n \times k_c^l}$ is used to produce $n_c^l = k_c^l$ new feature maps. This results in an output tensor $\mathrm{x}^l \in \mathbb{R}^{d_1 \times d_2 \times \cdots \times d_n \times n_c^l}$. To produce n_c^l new feature maps, $k_c^l n_c^{l-1}$ individual convolutions are performed within one convolutional layer. For the example of an RGB image $\mathrm{x}^{l-1} \in \mathbb{R}^{d_1 \times d_2 \times n_c^{l-1}}$, the convolution operation is defined as

$$(K^l * \mathrm{x}^{l-1})(q_1, q_2, q_c) \coloneqq \sum_{i_1}^{k_1} \sum_{i_2}^{k_2} \sum_{i_c}^{n_c^{l-1}} K(i_1, i_2, i_c) \mathrm{x}(q_1 + i_1, q_2 + i_2, q_c + i_c) \tag{3.9}$$

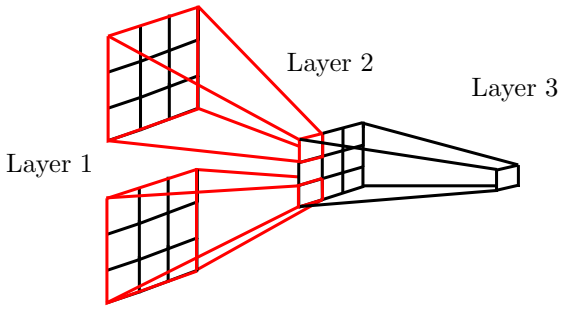


Fig. 3.3: Receptive fields of kernels over three layers. The deeper layer is indirectly connected to a larger receptive field, allowing for more abstract feature learning.

for $k_c^l = 1$. Note Equation 3.9 actually describes a cross correlation. A cross correlation is equivalent to a convolution with the same, flipped kernel. Also, note that in the following, we also refer to the size of a kernel $K \in \mathbb{R}^{k_1 \times k_2 \times k_c^l}$ with $k_1 \times k_2 \times k_c^l$, following conventions in the deep learning literature.

Each location (q_1, q_2, q_c) corresponds to a receptive field v_{q_1,q_2,q_c}. By visiting only a subset of receptive fields $v_{r_1 q_1, r_2 q_2, q_c}$ with $r_j \in \mathbb{N}$ we obtained a strided convolutional layer. Again, this corresponds to the strided convolution we introduced for image processing, see Section 2.2.2. Thus, we downsample the layer's output x^l by a factor of r_j in each dimension j.

The convolutional layer can be extended to the general case with N_d dimensions. The discrete convolution operation for N_d dimensions is defined as

$$(K * \mathrm{x})(q_1, \ldots, q_n) \coloneqq \sum_{i_1} \cdots \sum_{i_n} K(i_1, \ldots, i_n)\mathrm{x}(q_1 - i_1, \ldots, q_n - i_n) \tag{3.10}$$

which can be employed for a general convolutional layer, as given in Equation 3.9. While the mathematical extension to arbitrary dimensions is straight-forward, high-dimensional CNNs are difficult to design, which we address in Section 4.

Another important property of CNNs is the stacking of local receptive fields. Sliding the same filter over an input implies that only local properties can be captured. However, deep CNN architectures also allow for learning more abstract features, as the field of implicitly grows when stacking more layers. This is visualized in Figure 3.3.

In Section 2.2.2, we also addressed the problem of border effects for convolutions. A receptive field's location close to an image border is restricted by the kernel size. Thus, the maximum number of receptive fields u_{v_j} for dimension j is given by

$$u_{v_j} \leq \frac{d_j - k_j}{r_j} + 1. \tag{3.11}$$

Thus, the size of output tensor x^l for each axis j is at most $d_j - k_j + 1$ for $r_j = 1$, which leads to shrinking tensor dimensions when stacking convolutional layers. In some cases, this behavior is tolerated or desired. This mode of operation is referred to as VALID convolutional layers [176]. If we aim to keep the size d_j of all dimensions

constant across layers, we can employ zero padding with p_{d_j} zeros added around the tensor's border in all tensor dimensions d_j. We obtain

$$u_{v_j} \leq \frac{d_j + 2p_{d_j} - k_j}{r_j} + 1 \tag{3.12}$$

such that p_{d_j} can be chosen to adjust the new tensor's size. For $r_j = 1$ we can keep the image size constant with:

$$p_{d_j} = \frac{k_j}{2}. \tag{3.13}$$

This is referred to as SAME convolutional layers [176].

After the convolution operation in a layer, a nonlinear activation function is applied to the output x^l, similar to standard FC-NNs. Afterward, another operation called pooling is often applied. This operation reduces the spatial dimensions of a layer's feature map by grouping a region of the feature map with a summary statistic [176]. Usually, max pooling [594] is used which selects the largest value from a receptive field. Thus, we obtain

$$v^l = \max(v^{l-1}). \tag{3.14}$$

Other choices can be an average pooling operation or a more recent approach such as stochastic pooling [582]. Still, the standard approach in state-of-the-art architectures is to use max pooling over a 2×2 or $2\times2\times2$ region for the 3D case [475]. Max pooling comes with the advantage of introducing small-scale invariance towards minor translations. Assuming pixels shift inside of the pooling region, the max pooling operation will still output the same maximum value. While this is a desirable property for some applications, pooling operations can also be effectively replaced by convolutional layers with a stride of $r_j = 2$. This has been studied extensively with the result that CNNs without pooling operations can achieve similar or better performance with convolutional layers using a stride of $r_j = 2$ instead [463]. Thus, both approaches have been frequently used in practice.

Convolutional layers, including kernels, an activation function, and pooling, form the basis of a CNN. Usually, there are several consecutive convolutional layers that can be seen as a feature extraction stage, which is followed by an output layer where the standard FC-NN structure with matrix multiplications is used once again to produce a prediction vector. When moving from convolution-based processing to matrix multiplications, the feature tensor needs to be reshape into a vector. Thus, before entering the output layer, the feature tensor $\mathrm{x}^l \in \mathbb{R}^{d_1\times\cdots\times n_c^l}$ is flattened to $\hat{\mathrm{x}}^l \in \mathbb{R}^{d1\ldots n_c^l}$. This can be problematic if the dimensions d_j are large as the output layer will have a large number of parameters. A typical approach to overcome this is problem is global average pooling (GAP). Here, we apply an averaging pooling operation with a receptive field of size $v^{l-1} \in \mathbb{R}^{d_1\times\cdots\times d_n}$ to the final feature tensor x^l [299]. In this way, the output layer's number of parameters is substantially reduced.

The concepts we introduced so far would allow for construction of a CNN for image-based learning problems. However, from a practical perspective, a lot of hyperparameter choices are still unclear. As outlined in Section 3.2, hyperparameter optimization

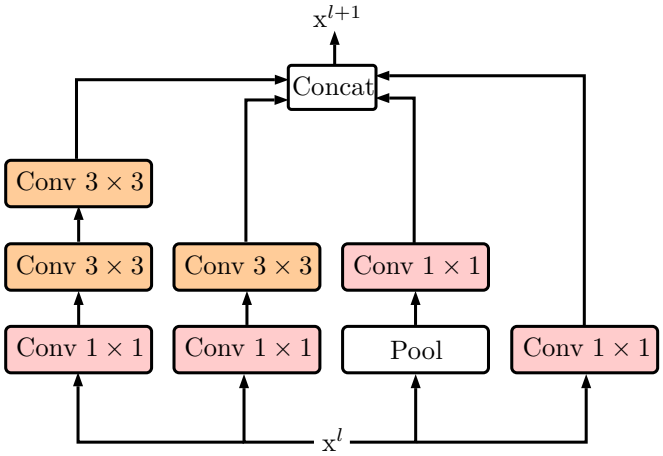


Fig. 3.4: Example for an Inception layer, based on [475].

is extensively influenced by search space design of the deep learning engineer. For example, the number of layers, the choice of k_c^l at each layer and the size of K appear to be arbitrary. Therefore, Simonyan et al. [450] introduced several fundamental design principles in their **VGG** CNN model which are still relevant and applied today. Simonyan et al. suggest that using kernel sizes $k_j = 3$ for all dimensions is the optimal choice for CNNs. Larger kernels have the advantage of covering a larger, local region, allowing for more context to be taken into account. However, considering Figure 3.3, we can observe that stacking multiple layers also leads to a larger implicit receptive field. As a result, a convolutional layer with kernel sizes $k_j = 5$ can be represented by two layers with kernel sizes $k_j = 3$. While reducing the number of parameters, for example, from 25 to 18 for the 2D case, this also allows for more nonlinearities being introduced in between layers which should allow the CNN to learn more abstract features [450]. Furthermore, Simonyan et al. suggest to use an initially small number of output feature maps k_c^l in the first convolutional layer. Then, k_c^l is doubled every time the spatial dimensions are reduced. In this way, architecture design is simplified as only the initial feature map size, the number of layers, and spatial reduction stages need to be defined.

Additional architecture design principles have been introduced with the **Inception** (IN) architecture [476]. Here, the idea is to split a single convolutional layer into parts. Consider the output feature map x^l of a layer in a CNN. In an Inception layer, this output is fed into several different convolutional layers at the same time. The outputs of these layers are concatenated back together to one output. This is motivated by the idea to extract features at different scales from the same input. The different layers in the Inception layer can be convolutions with different kernel size or pooling operations with stride $r_j = 1$. An example for an Inception layer is shown in Figure 3.4. This architecture was introduced in an improved iteration of the Inception architecture [475]. Note, that a convolutional layer with $k_j = 1$ for the spatial dimensions is placed in front of convolutional units. This serves the purpose of dimensionality reduction in the number of feature maps of the original input. This ensures that the computational effort does not become too large. Moreover, one branch contains two consecutive layers with spatial kernel sizes $k_j = 3$, following the idea of VGG [450]. Several Inception-like

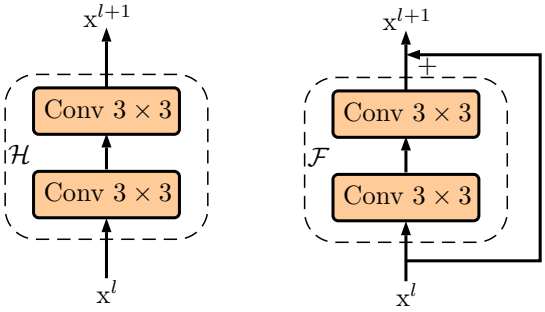


Fig. 3.5: Illustration of the ResNet principle. Left, two normal, consecutive convolutional layers are shown. Right, a typical ResNet block with two convolutional layers and a residual connection is shown.

layers have been introduced so far. While working well in practice, a downside of this concept is the extensive manual architecture engineering that is required.

ResNets (RN) are another concept that is crucial for architecture design. He et al. [193] observed that there was a general trend towards deeper networks with more layers. However, there is a limit for the number of layers that is effective in a CNN model, and for deeper versions, test performance goes down. At some point, the *degradation* problem occurs. Here, very deep networks show both a higher test *and* training error than shallower versions. Thus, the deep networks do not fit the data well, indicating that the optimization process does not work well. He et al. proposed residual connections inside the network as a solution. We assume that several convolutional layers should learn a target mapping $\mathcal{H}$ at layer l. Instead, we can also learn a mapping:

$$\mathcal{H}(\mathrm{x}^l) = \mathcal{F}(\mathrm{x}^l) + \mathrm{x}^l. \tag{3.15}$$

Here, the key assumption is that a mapping $\mathcal{F}$ is easier to learn as the network is only required to learn a residual or some small deviation from x^l instead of a full mapping. The implementation of this idea can be realized using a skip connection, see Figure 3.5. In practice, residual connections have been proven to be very effective, enabling very deep and higher-performing CNNs. Almost all modern CNN architectures utilize residual connections in some way.

Summarized, CNNs generally consist of multiple convolutional layers, activation functions, pooling, and an output layer. A generic architecture overview is shown in Figure 3.6. Most modern architectures follow this concept and largely focus on improving the structure of the convolutional blocks.

3.3.3 Recurrent Neural Networks

As discussed in the previous section, CNNs can also be used to process temporal dimensions using convolution operations. As an alternative, recurrent neural networks have also been introduced, which are specifically designed for processing sequential data. Processing sequences of examples is necessary for tasks where the learning target cannot be derived from a single example, for example, when trying to forecast the movement of an object. One approach could be to consider all previous examples as

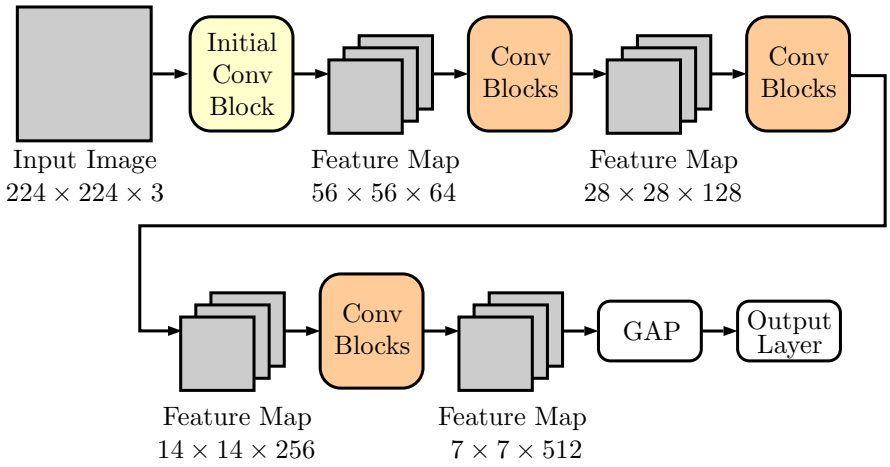


Fig. 3.6: An example of a full CNN architecture for a 2D input image of size 224×224. The initial convolutional block typically consists of several convolutional layers. The other convolutional blocks can be ResNet blocks, Inception blocks or other variants.

the input for a neural network. However, this approach can be very inefficient due to the large number of model parameters required, similar to processing an image with an FC-NN. Therefore, RNNs have been developed with the core idea of sharing parameters for several computations within a sequence. Often, sequences represent temporal data where each example within the sequence corresponds to a time point t_i. While most applications include temporal sequences, the only necessary assumption for RNN applications is that the sequence follows some order [176].

In the following, we will assume that some type of time series is being processed. In general, an RNN processes a sequence $\mathrm{x}_i = \{\mathrm{x}_{t_{i-n_t}}, \ldots, \mathrm{x}_{t_i}\}$ of length n_t. We refer to t_i as the current time step that comes with history of $n_t - 1$ predecessors. At a time step t_i, we compute

$$h_{t_i} = f_M(h_{t_{i-1}}, \mathrm{x}_{t_i}, w_M) \tag{3.16}$$

where h_{t_i} is the RNNs internal state at time step t_i and w_M are the RNNs parameters. The RNN computes the next state h_{t_i} using a previous time step $h_{t_{i-1}}$, the current sequence example x_{t_i}, its weights w_M, and some function f_M. In a typical RNN application, we try to predict future values from a history of data $\mathbf{x}_i$. Here, h_{t_i} can be interpreted as a compressed representation of previous information contained in $\mathbf{x}_i$. As x_i is often a long sequence with a lot of examples, h_{t_i} is chosen to be limited in size such the only the most relevant information for the task at hand can be stored.

The transforming function f_M can be chosen in different ways. In a typical RNN, feature vectors x_{t_i} and states $h_{t_{i-1}}$ are transformed using matrix multiplications, similar to FC-NNs. To produce usable outputs o_{t_i}, an additional transformation of h_{t_i} is performed at each time step. Thus, for an example RNN we compute

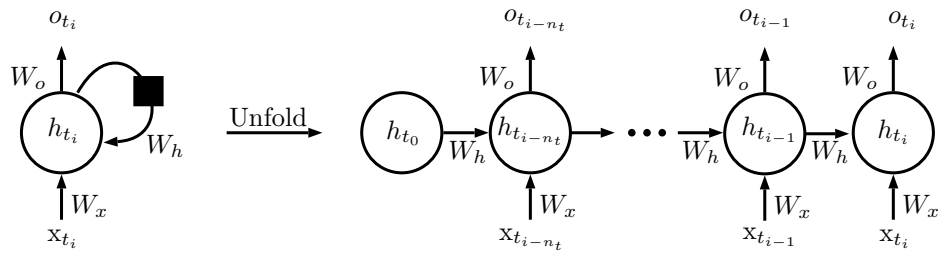


Fig. 3.7: An example of an RNN, both in its single-node representation (left) and its finite unrolled representation (right). For the single-node representation, the black box at the recurrent connection indicates a delay of one time step. Note that for the unrolled representation, the computations at all time steps share parameters.

$$a_{t_i} = W_h h_{t_{i-1}} + W_x \mathrm{x}_{t_i} \tag{3.17}$$
$$h_{t_i} = g_a(a_{t_i}) \tag{3.18}$$
$$o_{t_i} = W_o h_{t_i} \tag{3.19}$$

where g_a is an activation function and matrices $W \in w_M$ are the learnable model parameters. These computations can be visualized in two different ways, see Figure 3.7. One way is to represent the computation by a single node with a recurrent connection. Another way is to unfold the RNN into a computation graph where the computations for a finite set of time steps are depicted. While the single node is closer to a physical representation of an RNN, for example, in a biological neuron, the unfolded version explicitly shows all the computations that are involved. The unfolded version is limited in terms of the number of time steps which represents realistic computations. In practice, an RNN is trained with truncated sequences using the unfolded graph representation.

Note that Equation 3.19 is just one option for RNN design. State, input example, and output can be connected in arbitrary ways, which allows for designing specialized RNNs for different applications. Furthermore, matrix multiplications are not necessarily required. Some RNN extensions are designed to deal with image-like data by performing convolution operations instead of matrix multiplications [556].

One important aspect of RNNs is the design of their output. Following Equation 3.19, we can produce a prediction at every time step. Some tasks do not require an output at every time step, for example, when predicting only a current or future estimate at the final time step t_i in an unfolded RNN. Here, previous examples only serve as a source of information for the current output, and we are not interested in outputs of previous time steps. This is important for training RNNs as we only create a loss signal for learning at the RNN's final time step.

A major shortcoming of RNNs is their ability to deal with very long sequences and long-term dependencies. Assuming that a prediction at the final time point t_i depends on an input example from the past, transferring the information through multiple time steps in the state h_{t_i} can be difficult. The state h_{t_i} undergoes a transformation where its values are multiplied with the same weight matrix multiple times. If values in h_{t_i} are

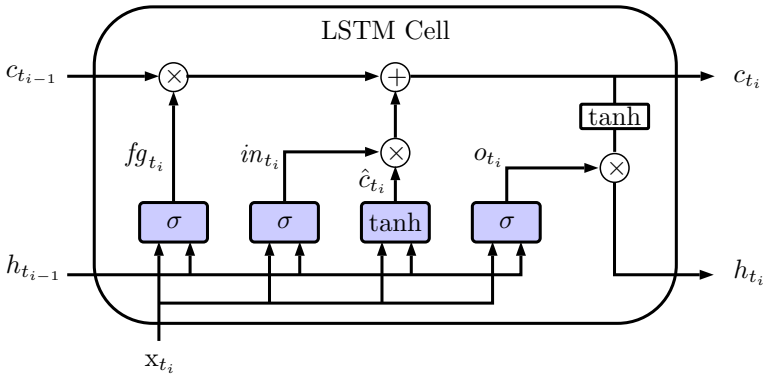


Fig. 3.8: An example of an LSTM cell [200]. Blue blocks indicate a neural network layer with learnable weights. White blocks indicate element-wise operations. $\sigma(\mathrm{x})$ refers to a sigmoid activation function.

smaller than 1, chained multiplications will eventually pull the value to zero, and if the values are larger than 1, they will explode. While exploding values can be avoided with sigmoid-like activation functions, values will saturate instead. In the case of saturation, the vanishing gradient problem occurs, which prevents RNNs from training effectively. The problem of long-term dependencies in long sequences has been extensively studied, for example, by Bengio et al. [41] and Hochreiter et al. [199].

One approach for overcoming this problem are *gated* RNNs. Here, the idea is to create connections between time steps which do not result in vanishing or exploding gradients. Hochreiter and Schmidhuber [200] implemented this idea using self-loops within the RNN block, which allows the gradient to flow within the network for a long time. This RNN model is termed long short-term memory (LSTM) cell, see Figure 3.8. A core part of the LSTM is its cell state c_{t_i}, representing the LSTM's self-loop. The cell state can remain the same for a long time as it is only manipulated by element-wise multiplications or additions. Changes to the cell state, for example, for adding or removing information, can be introduced through gates.

Gates are neural network layers with a sigmoid-like activation function g_a. The first gate is the forget gate, which controls what kind of information is kept in the cell state and what is removed. If the gates' activations are zero, the element-wise multiplication will remove said values from the cell state. If the activations are one, they pass through unchanged. The update gate computes

$$fg_{t_i} = \sigma(W_{fh} h_{t_{i-1}} + W_{fx} \mathrm{x}_{t_i}) \tag{3.20}$$

where we consider a similar input and state transformation as in Equation 3.19. Next, new information can be added to the cell state with the input gate. First, a new candidate cell state $\hat{c}_{t_i}$ is computed using another neural network layer. Typically, this layer uses a $\tanh$ activation function. Here, any activation function can be used as the output values do not directly affect the cell state c_{t_i}. Next, another gating layer determines which parts of the candidate state are kept for addition to the previous cell state $c_{t_{i-1}}$. Thus, for the input gate, we compute

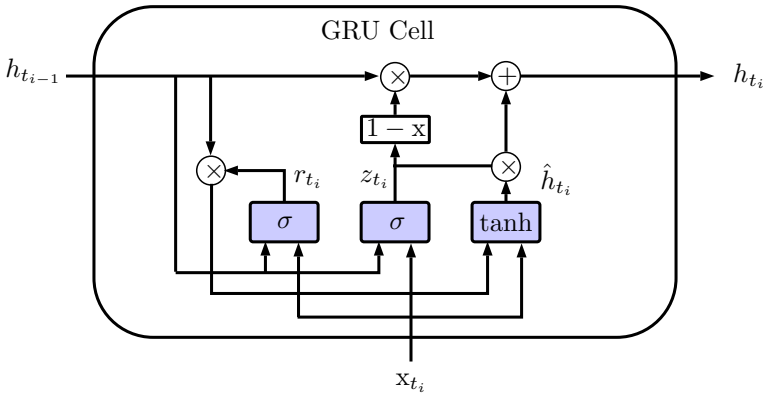


Fig. 3.9: An example of a GRU cell [87]. Blue blocks indicate a neural network layer with learnable weights. White blocks indicate element-wise operations. $\sigma(\mathrm{x})$ refers to a sigmoid activation function.

$$
\begin{aligned}
in_{t_i} &= \sigma(W_{ih} h_{t_{i-1}} + W_{ix} \mathrm{x}_{t_i}) && (3.21)\\
\hat{c}_{t_i} &= \tanh(W_{ih} h_{t_{i-1}} + W_{ix} \mathrm{x}_{t_i}). && (3.22)
\end{aligned}
$$

Thus, the new cell state c_{t_i} is computed by

$$
c_{t_i} = fg_{t_i} c_{t_{i-1}} + in_{t_i} \hat{c}_{t_i}. \tag{3.23}
$$

Finally, the LSTM's output needs to be computed. Here, the output is the new cell state c_{t_i} transformed with a $\tanh$ activation. Then, an output gate computes which values are kept within the transformed cell state:

$$
\begin{aligned}
o_{t_i} &= \sigma(W_{oh} h_{t_{i-1}} + W_{ox} \mathrm{x}_{t_i}) && (3.24)\\
h_{t_i} &= o_{t_i} \tanh(c_{t_i}). && (3.25)
\end{aligned}
$$

Many different LSTM variants have been introduced where the gating process is slightly modified. A very popular modification are gated recurrent units (GRUs) [87]. They are slightly more efficient than standard LSTMs as they combine the input and forget gate into a single update gate. Furthermore, the cell state $\hat{c}_{t_i}$ and the hidden state $\hat{h}_{t_i}$ are fused into a single representation. The GRU performs the following computations:

$$
\begin{aligned}
z_{t_i} &= \sigma(W_{zh} h_{t_{i-1}} + W_{zx} \mathrm{x}_{t_i}) && (3.26)\\
r_{t_i} &= \sigma(W_{rh} h_{t_{i-1}} + W_{rx} \mathrm{x}_{t_i}) && (3.27)\\
\hat{h}_{t_i} &= \tanh(W_{hh} h_{t_{i-1}} + W_{hx} \mathrm{x}_{t_i}) && (3.28)\\
h_{t_i} &= (1 - z_{t_i}) h_{t_{i-1}} + z_t \hat{h}_{t_i} && (3.29)
\end{aligned}
$$

A visual interpretation of a GRU cell is shown in Figure 3.9. Note that the GRU only uses three instead of four neural network layers. Also, the fused hidden representation reduces the model's memory footprint. These advantages have made GRUs a popular alternative for LSTMs in a lot of tasks.

3.3.4 Training Neural Networks

Training neural networks follows the optimization problem given in Equation 3.5, where a loss function J_P is minimized by adjusting the network's parameters w_M given a training dataset $\mathcal{X}_{train}$. While the different neural network architectures can have a very different internal structure, their parameters can be chosen by solving the same optimization problem.

In general, the optimization problem for neural networks is non-convex [278]. Therefore, optimization is difficult due to the presence of local minima in the loss landscape. The optimization problem for neural networks is usually solved using a type of gradient descent [278]. First, the network's parameters w_M are initialized, typically with small values sampled from a normal distribution. More advanced initialization strategies are discussed by Glorot et al. [174] and Sutskever et al. [473].

Then, the gradients of w_M with respect to the loss function $\nabla_{w_M} J_P(w_M, \mathrm{x}_i, \mathrm{y}_i)$ are obtained using the average loss value over the entire training dataset. All trainable parameters w_M^j at training iteration j are updated by the rule

$$w_M^j = w_M^{j-1} - \alpha_{lr} \frac{1}{N_{train}} \sum_{i=1}^{N_{train}} \nabla_{w_M} J_P(w_M^{j-1}, \mathrm{x}_i, \mathrm{y}_i) \tag{3.30}$$

where α_{lr} is the learning rate which controls by how much the current parameters are updated. This step is repeated several times until the value of the loss function J_P converges to a small value.

The gradients are computed using the back-propagation algorithm [420]. The algorithm computes the gradients of the network's parameters in an iterative fashion where the neural network and its operations are interpreted as a computational graph. First, forward-propagation is performed where we obtain $\hat{\mathrm{y}}_i = f_M(\mathrm{x}_i, w_M^j)$. During forward-propagation, all intermediate outputs of operations in the network's computational graph are stored. Then, the gradient of each parameter and each operation in the network is computed. Here, the partial derivative of every operation with respect to each of its inputs is required. Starting at the loss function's value, all partial derivatives can be computed iteratively using the chain rule and the values obtained by forward-propagation. A more detailed description is given by Rumelhart et al. [420] and Goodfellow et al. [176].

After having obtained the gradients of the network's parameters, the update can be performed. For this algorithm, we usually use the complete training set for each update step. All training samples are propagated forward into the network and the gradient updates are performed with respect to the average of all their errors. In practice, this is usually not possible as all intermediate layer outputs have to be stored in memory. In case of images as the input, the memory requirements become very large. For higher-dimensional 3D and 4D data, in particular, a different procedure is necessary. One approach is to only use a small batch $\mathcal{B}_{train} \subset \mathcal{X}_{train}$ of all available training examples for each update, which is called mini-batch stochastic gradient descent (SGD) [56]. Strictly speaking, SGD only assumes one example in each update iteration; however, using a mini-batch of input examples is still computationally feasibly and promises faster convergence [176]. For mini-match SGD, the update rule for trainable parameters

Algorithm 1 The generic SGD algorithm for a variable batch size N_b and variable learning rate.

Require: α_{lr}^0: Initial learning rate
Require: w_M^0: Initial parameters
Require: $J_P(w_M, \mathrm{x}, y)$: Loss function
Require: $A(\alpha_{lr}, j)$: Learning rate update function
  $j \leftarrow 0$
  while w_M not converged **do**
    $j \leftarrow j + 1$
    Sample $\mathcal{B}_{train} \subset \mathcal{X}_{train}$
    $g_t \leftarrow \frac{1}{N_b} \sum_{(x,y)\in\mathcal{B}_{train}} \nabla_{w_M} J_P(w_M, \mathrm{x}, \mathrm{y})$
    $w_M^j = w_M^{j-1} - \alpha_{lr}^{j-1} g_t$
    $\alpha_{lr}^j \leftarrow A(\alpha_{lr}^{j-1}, j)$
  end while
  return w_M

is given by

$$w_M^j = w_M^{j-1} - \alpha_{lr} \frac{1}{N_b} \sum_{i=1}^{N_b} \nabla_{w_M} J_P(w_M^{j-1}, x_i, y_i) \tag{3.31}$$

where N_b is the batch size. Using a single sample per update with $N_b = 1$ can still be advantageous as it often leads to a better generalization error. The small batch size is assumed to have a regularizing effect as it adds noise to the gradient [539], which can help avoiding local minima. However, this requires very long training times as parallel computation for multiple examples cannot be exploited.

The learning rate α_{lr} is an important hyperparameter with a lot of influence on the final training result. Generally, the learning rate's magnitude is a trade-off between the choice of larger values for faster convergence and smaller values for better convergence to a minimum and reduced risk of overshooting the target. Therefore, larger values are more appropriate in early stages of training where larger steps can be made without an elevated risk of missing out a minimum. In later stages, when the parameter values are closer to a minimum, smaller learning rates are preferred to make sure the minimum is not skipped. This motivates an adaptive learning rate. Often, an exponentially decaying learning rate is used such that

$$\alpha_{lr} = \alpha_{lr_0} d_{dc}^{N_i/N_{dc}} \tag{3.32}$$

where α_{lr_0} is the base learning rate, d_{dc} the decay rate, N_i the number of iterations, and N_{dc} the decay steps. It is also possible to reduce the learning rate in fixed steps, which would resemble Equation (3.32) with an integer division in the exponent. These techniques promise to provide better convergence than a fixed learning rate [219]. Some more recent adaptations of SGD also incorporate an adaptive learning rate directly into the algorithm [581]. Summarized, the SGD algorithm for neural network training is given by Algorithm 1.

Algorithm 2 The Adam algorithm. The parameter ϵ is added for numerical stability.

Require: α_{lr}^0: Initial learning rate
Require: w_M^0: Initial parameters
Require: $J(w_M, \mathrm{x}, \mathrm{y})$: Loss function
Require: $A(\alpha_{lr}, j)$: Learning rate update function
  $j \leftarrow 0$
  $\mu^0 \leftarrow 0$
  $var^0 \leftarrow 0$
  while w_M not converged **do**
    $j \leftarrow j + 1$
    Sample $\mathcal{B}_{train} \subset \mathcal{X}_{train}$
    $g_t \leftarrow \sum_{(x,y)\in\mathcal{B}_{train}} \nabla_{w_M} J_P(w_M, \mathrm{x}, \mathrm{y})$
    $\mu^j = \beta_1 \mu^{j-1} + (1-\beta_1) g_j$
    $var^j = \beta_2 var^{j-1} + (1-\beta_2) g_j^2$
    $\hat{\mu}^j = \frac{m^j}{1-\beta_1^j}$
    $\hat{var}^j = \frac{var^j}{1-\beta_2^j}$
    $w_M^j = w_M^{j-1} - \alpha_{lr}^{j-1} \frac{\hat{\mu}^j}{\sqrt{\hat{var}^j}+\epsilon}$
    $\alpha_{lr}^j \leftarrow A(\alpha_{lr}^{j-1}, j)$
  end while
  return w_M

SGD is the basis of more advanced techniques which have been introduced for shorter training times and improved convergence properties. One of the most popular methods is Adam, which is an algorithm for stochastic objective functions that is based on first and second order statistical moment estimates [250]. Here, the objective function is stochastic in the sense that it is evaluated at different subsets of data at different points in time while training a neural network. Gradient updates in each training iteration are performed with several steps.

First, the gradients w_M^j are obtained with respect to the objective function at iteration j with parameters w_M^{j-1}. The gradients are used to update the first and second order statistical moment estimates of the gradient, the mean μ^j and variance var^j. In this way, gradient information from previous updates is still considered in the current update. Before being used for the gradient update, the mean and variance estimates are corrected for their bias. Two parameters β_1 and β_2 decay the estimates, thus limiting the influence of previous updates. A learning rate α_{lr} controls the step size. This leads to the improved procedure given by Algorithm 2.

This method promises to perform well as different parameters are individually updated through the adaptive moment estimates. The algorithm is based on the popular methods RMSprop [489] and AdaGrad [116] and outperforms both in most scenarios, as shown by Kingma et al. [250]. More recently, adjustments and improvement have been introduced for Adam. For example, Nadam [113] also incorporates Nesterov momentum [355] into the algorithm and Radam [304] proposes a rectified estimation of the gradients' variance.

3.3.5 Regularization

Regularization describes methods that try to prevent overfitting in a model, as explained in the context of generalization in Section 3.2. There is a vast amount of regularization strategies. In many cases, techniques related to deep learning have an additional regularizing effect, although their primary purpose is focused on something else. Therefore, regularization is a loose term that can refer to a lot of different techniques. Goodfellow et al. define regularization as "any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error" [176].

Conventional regularization techniques usually enforce explicit regularization. A common method is to impose a penalty on the neural network's weights, which is added to the loss function

$$\hat{J}(w_M, \mathrm{x}, \mathrm{y}) = J_P(w_M, \mathrm{x}, \mathrm{y}) + \lambda_w \Omega(w'_M) \tag{3.33}$$

where λ_w trades off parameter fitting to the data and regularization. Note that w_M is the set of all parameters and w'_M the set of regularized parameters with $w'_M \subseteq w_M$. One of the most popular penalty choices is the L^2 norm, also referred to as weight decay, which is given by

$$\Omega(w'_M) = \frac{1}{2}\|w'_M\|_2^2. \tag{3.34}$$

Similarly, the L^1 norm penalty is given by

$$\Omega(\theta) = \|w'_M\|_1. \tag{3.35}$$

Both approaches behave differently. The L^2 norm keeps weights at smaller magnitudes by adding a term to the gradient that scales with the weight's value. Therefore, larger weights that deviate a lot from zero are penalized more. The L^1 norm, on the other hand, always contributes a constant term. This leads to the L^1 norm enforcing sparsity in the weights and driving some of them to zero. This property can be used as a feature selection mechanism [356].

Another approach that is typically employed is the use of early stopping. When training a model, the validation and training error will initially both go down as the model is fitted to the data. At some point, if the model capacity is large enough, the model will start overfitting, and the validation error starts to increase [387]. Thus, a simple algorithm can be used to save the current best model and stop training when the validation error keeps increasing for several iterations. The regularizing effect is achieved by limiting the parameter space to which the model can extend [52]. Assuming a bounded gradient, a learning rate, and a certain number of iterations before we stop, the model's parameter space volume around the initial parameter set w_M^0 is limited. In fact, early stopping limits a model's capacity similar to the L^2 norm [176].

A technique already introduced in the context of convolutional and recurrent neural networks is parameter sharing which also imposes a regularizing effect. While a norm constraint forces parameters to be close to zero, another approach is to force parameters to be similar to each other, where the extreme case is to force them to be equal. This is motivated by the idea that we often search for the same kind of features in different

regions of an input. Therefore, CNNs and RNNs already have an in-built regularization effect.

Another popular method is the use of model averaging or ensemble methods. For this approach, multiple models are trained, and their outputs are aggregated, for example, by calculating a weighted average of the models' outputs [334]. Given a set of trained models $\mathcal{F}_M = \{f_M^1, \ldots, f_M^{N_{en}}\}$ with N_{en} trained models, we obtain the new ensemble prediction with

$$\hat{\mathrm{y}}_{en} = E_{en}(\mathcal{F}_M(\mathrm{x})) \tag{3.36}$$

where E_{en} aggregates individual model predictions, for example, by averaging. This usually leads to better performance compared to a single model, however, the performance is traded off for significantly higher computational costs as several models have to be trained and evaluated.

The methods presented above have been introduced before the emergence of deep learning, but they are still relevant and effective in today's models. In addition, more recently, methods have been introduced which were designed for modern deep learning architectures.

Dropout was introduced by Hinton et al. [198] as a computationally efficient method to regularize FC-NNs. The basic idea is to randomly drop hidden units by setting layer outputs to zero during training with a probability of p_d to keep weights from co-adapting too much [464]. Thus, when setting a hidden unit to zero, its connections to the following layer all contribute a value of zero. This corresponds to sampling a network out of a set of potential networks in each training iteration, where all networks share the same set of parameters. For a single-layer network with d_{NN} neurons, a new network is sampled from $2^{d_{NN}}$ possible networks at each training iteration. For evaluation after training, this would lead to an ensemble of $2^{d_{NN}}$ networks that need to be evaluated. Following Equation 3.36 with averaged aggregation, we need to calculate

$$\hat{\mathrm{y}}_{en} = \frac{1}{2^{d_{NN}}} \sum_{i=1}^{2^{d_{NN}}} f_M^i(\mathrm{x}). \tag{3.37}$$

As this is computationally infeasible, the weight scaling inference rule can be used instead. Here, all neurons are scaled with the dropout probability, which is an approximation of evaluating the entire ensemble. Thus, only a single model is required, which makes the method very efficient while also showing considerable performance improvements across different tasks [464].

Dropout inspired other methods that follow the idea of stochastic model averaging. For example, DropConnect does not set entire units to zero but drops only some connections between input and output units [517]. Typically, these methods are used for FC-NNs, but they have also found applications with CNNs and RNNs. For CNNs, we can set pixels or voxels within a feature map to zero instead of setting hidden units to zero. Alternatively, Tompson et al. proposed that dropping entire feature maps instead of individual pixels is more beneficial for CNNs [490]. As neighboring pixels are usually strongly correlated, single pixel dropout does not enforce independence but only scales the gradient by the dropout probability. Thus, preventing co-adaptation between entire feature maps is deemed preferable for CNNs [490].

Similarly, vanilla dropout is not optimal for RNNs. Introducing dropout within recurrent cells has been shown to be ineffective as long-term information appears to be lost [578]. Therefore, most approaches only applied dropout to the input and output of the RNN, not to recurrent connections within cells [378]. Another popular approach found that dropout can be applied effectively if the same dropout mask is used for all recurrent steps within a training step [144].

Batch normalization [214] is a technique that accelerates training while also providing a regularizing effect. The main problem addressed by batch normalization is called internal covariate shift. When training deep networks, intermediate layer outputs and gradients tend to vary significantly. As gradients depend on surrounding layer outputs and weights within the computational graph, different changes surrounding different weights make the decision for a single learning rate difficult. Therefore, it would be desirable to keep all intermediate layer outputs and gradients in a similar range. For an FC-NN, Ioffe et al. suggest normalizing the output of each neuron in a layer independently such that

$$\hat{\mathrm{x}}_j^l = \frac{\mathrm{x}_j^l - \mathrm{E}[\mathrm{x}_j^l]}{\sqrt{\mathrm{Var}[\mathrm{x}_j^l]}} \tag{3.38}$$

where x^l is the layer's output for the j^{th} neuron in layer l. The mean and variance estimates are calculated over the mini-batch $\mathcal{B}_{train} \subset \mathcal{X}_{train}$ in each training step. As a result, layer outputs depend on the current batch composition and other examples in the batch. This would lead to inconsistent results during inference. Therefore, the calculated means and variances are saved as moving averages during training and applied as a fixed normalization during inference.

After a batch normalization layer, the network's capacity can be reduced. For example, for a sigmoid activation function, the normalized values would always reside within the function's linear region between 0 and 1. To avoid this problem, a trainable linear transformation layer is added after normalization. The transformation is defined as:

$$\tilde{\mathrm{x}}^l = \gamma_M^l \hat{\mathrm{x}}^l + \beta_M^l \tag{3.39}$$

where ${\gamma_M}^l$ and ${\beta_M}^l$ are learnable parameters in layer l. Ioffe et al. suggest to include the batch normalization layer directly in front of activation functions within an FC-NN. Note that the mean subtraction eliminates the bias variable that is usually present. The bias is implicitly included again with the transformation in Equation 3.39.

In practice, batch normalization has become a core component of almost all deep learning models. The normalization significantly enhances the training process with well-conditioned gradients, leading to both faster convergence and better performance. Usually, networks using batch normalization can be trained with a larger learning rate while still converging correctly. In addition, the technique comes with an implicit regularizing effect. The mean subtraction and division by variance are a source of noise added to the data, which varies with each batch. Thus, in each training epoch, training samples are transformed differently, given that batches are assembled randomly. This forces the model to be more robust towards distorted inputs. Therefore, Ioffe et al. suggest to tone down other regularization measures such as L^2 regularization or dropout as batch normalization does already provide sufficient regularization.

When applying batch normalization with CNNs and RNNs, modifications are required. For CNNs, the typical approach is to apply batch normalization across the batch dimension and all spatial image dimensions. Thus, each feature map within the network is normalized independently [214]. For RNNs, recurrent batch normalization has been introduced where batch normalization layers applied both to the input transformation and the hidden state transition [94]. There are no batch normalization layers for the cell state update in order to keep the long-term gradient flow untouched.

Over the years, several improvements for batch normalization have been proposed. For example, batch re-normalization addresses the problem of batch size dependence [213]. Here, additional parameters are introduced to obtain better results for small batches. Furthermore, instance normalization has been proposed where each example in a batch is normalized independently [498]. Thus, the method also works well for a batch size of 1, where batch normalization cannot be applied. Group normalization is another extension of batch normalization for small batch sizes [548]. Here, feature maps within the network are split into different groups, and each group is normalized independently. While batch normalization is still the standard normalization method for most applications, instance normalization, and group normalization are particularly well suited for higher-dimensional data where batch sizes need to be small due to the exponential increase in memory requirements.

Data Augmentation is a regularization method that addresses overfitting by changing the network inputs instead of the network itself. Typical data augmentation strategies apply a transformation $\phi(\mathrm{x})$ to an image x, for example, rotation, which allows for generating new, artificial examples. While this can be interpreted as a way to generate additional data, its actual purpose is to force a deep learning model to become invariant towards the transformation $\phi(\mathrm{x})$. If we show the same image with different rotations to the network while the image's label stays the same, the network should learn to ignore rotations and achieve robustness towards that transformation. In terms of regularization, this means that we can keep the network from overfitting to a particular pose and orientation for the example of rotations.

As a result, we can design data augmentation strategies that transform inputs in a way that is not relevant to the learning task at hand. For example, if we want to classify a disease based on an image of a lesion, the pose is likely not relevant, and data augmentation by rotation could be beneficial. However, if we want to determine an object's pose within the image, rotations are likely detrimental as the object's pose is changed. Therefore, data augmentation strategies need to be carefully designed for each application.

Besides rotation, flipping, scaling, and shear are popular data augmentation strategies. In the natural image domain, brightness and contrast, as well as color adjustments, are popular. Random cropping is another technique related to data augmentation. Here, the idea is to crop a small part out of a larger image for model training. In this way, the model only observes a part of the input image at a time. This can prevent overfitting to image background or clutter, which is not relevant for the learning task. For application, we need to ensure that the most relevant part of the image is still contained in the crop.

In general, data augmentation strategies are only applied during training and not evaluation, where performance is judged based on real images. As model evaluation should be deterministic, random cropping needs to be replaced, for example, by taking

a single center crop from the image. As an alternative, multi-crop evaluation can be used where we take N_{crops} crops $\tilde{\mathrm{x}}_j$ from the image x, evaluate them individually with the trained model, and then aggregate the result. Calculation is performed similar to Equation 3.36, except the same model is evaluated on different image patches. Thus, for multi-crop evaluation, we calculate

$$\hat{\mathrm{y}}_{mc} = \frac{1}{N_{crops}} \sum_{j=1}^{N_{crops}} f_M(\tilde{\mathrm{x}}_j). \tag{3.40}$$

Similar to ensembling, this approach usually improves robustness and performance, however, it is traded off for an increased computational effort.

3.3.6 Machine Learning Tasks

In the previous sections, most aspects of deep neural network design and training have been covered. Here, we briefly discuss how neural network models differ at the output for different learning tasks. The three most common learning tasks are classification, segmentation, and regression.

Classification. The most common deep learning problem for medical diagnosis tasks is classification. Here, the network needs to distinguish between one or more classes. In terms of network design, a common deep learning model uses a structure as described in Section 3.3.2, where the network input is gradually resized by several consecutive layers until a GAP layer is reached. Afterward, a fully-connected output layer needs to map features to classes. To obtain a categorical output that matches the classification problem, a one-hot approach paired with a softmax activation function can be used. One-hot encoding is an approach where N_c classes are represented by a vector $o_{class} \in \mathbb{R}^{N_c}$, which is chosen as the size of the output layer. While N_c classes could also be represented by a scalar that takes value from 0 to $N_c - 1$, the one-hot encoding enables us to formulate the classification problem in terms of approximating our target data distribution. By applying the softmax function

$$\mathrm{softmax}(x) = \frac{\exp(\mathrm{x})}{\sum_{j=1}^{N_c} \exp(\mathrm{x}_j)} \tag{3.41}$$

to the output of our last layer, we obtain an approximation of a probability distribution over all target classes. Now we can formulate a loss for approximating our target distribution by using the negative log-likelihood, also called cross-entropy between training data and deep learning model's prediction distribution. For the binary case with two classes, the cross-entropy loss is defined as

$$J_{CEB} := -(\mathrm{y} \log(\hat{\mathrm{y}}) + (1 - \mathrm{y}) \log(\hat{\mathrm{y}})) \tag{3.42}$$

where $\hat{\mathrm{y}}$ is a softmax model output. Similarly, for the multi-class case, the cross-entropy loss is defined as

$$J_{CEM} := -\sum_{j=1}^{N_c} \mathrm{y} \log(\hat{\mathrm{y}}). \tag{3.43}$$

Tab. 3.1: Example of a confusion matrix for binary classification where class A is the positive class and class B is the negative class.

		Ground-Truth	
		Class A	Class B
Prediction	Class A	TP	FP
	Class B	FN	TN

Note that softmax is used to approximate distributions with mutually exclusive classes. For some classification problems, an example can belong to multiple classes at the same time. Here, the model outputs can be mapped to probabilities by individually scaling them using a sigmoid function.

For evaluation, the cross-entropy is usually not employed as it is difficult to interpret. Instead, metrics such as accuracy, sensitivity, specificity, precision, and F1-score are often used. These metrics are defined as

$$Acc := \frac{TP + TN}{TP + TN + FN + FP} \tag{3.44}$$

$$Sens := \frac{TP}{TP + FN} \tag{3.45}$$

$$Spec := \frac{TN}{TN + FP} \tag{3.46}$$

$$Prec := \frac{TP}{TP + FP} \tag{3.47}$$

$$F1 := 2\frac{PrecSens}{Prec + Sens} \tag{3.48}$$

for the case of binary classification. True positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) can be derived from a confusion matrix, as shown in Table 3.1. These can be extended to a multi-class problem with a one-vs-all approach. The accuracy provides an indication of how often a model classifies an example correctly in relation to all examples available. Due to class imbalance, this metric can be misleading, for example, if one class is more common than the other classes. For this purpose, sensitivity and specificity can provide more information. Intuitively, the sensitivity indicates how well a model detects a class. A high sensitivity is often traded for a lower specificity as a model that finds class A very consistently, while often classifying examples of class B as class A, has a high sensitivity and a low specificity. The $F1$-score is a more balanced metric as it combines precision and sensitivity in a single metric.

Segmentation. In many medical applications, the goal is not to assign a class to an image but to segment different structures within the images. For example, for multiple sclerosis, lesions in the brain need to be segmented in MRI scans. Thus, the model input is an MRI image, and the output is a classification of every pixel or voxel, which corresponds to a segmentation map. As given in Equation 3.1, the output needs to be spatially resolved. Thus, CNN architecture design needs to be different for this task. Encoder-decoder models are a popular approach for this problem, which were

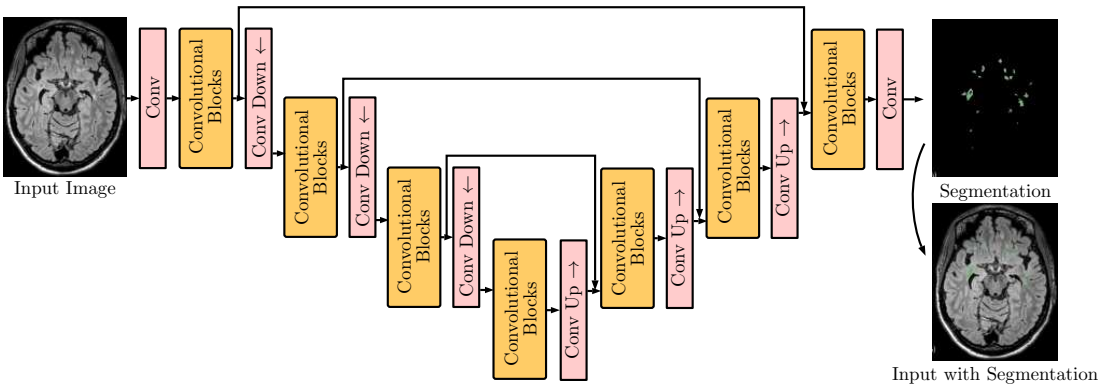


Fig. 3.10: An example for a segmentation task using an encoder-decoder CNN.

popularized by U-Net [409]. An example structure, including the input image and the output segmentation map, is shown in Figure 3.10.

In terms of the network structure, the encoder processes the image similar to a normal CNN. This results in a compressed representation that captures important information in the image. Then, the decoder upsamples this representation until it reaches the same spatial size as the input.

Segmentation is similar to classification as we perform a pixel- or voxel-wise classification. Therefore, the loss functions J_{CEB} and J_{CEM} can be applied to every pixel or voxel output independently, which is averaged into a final, scalar loss value. A problem that commonly occurs is a severe imbalance between classes within the image. For example, for multiple sclerosis, most of the voxels in the image belong to the background class, and only a few correspond to the foreground class. Therefore, other loss functions have been developed. A popular approach is the dice loss function [339]. Here, we minimize a variant of the Sorensen-Dice coefficient [107, 461] which is defined as

$$J_{DB} := 1 - \frac{\mathrm{y}\hat{\mathrm{y}} + \epsilon}{\mathrm{y} + \hat{\mathrm{y}} + \epsilon} - \frac{(1-\mathrm{y})(1-\hat{\mathrm{y}}) + \epsilon}{2 - \mathrm{y} - \hat{\mathrm{y}} + \epsilon} \tag{3.49}$$

for the binary classification case where ϵ is a small scalar for numerical stability. For the multi-class case, the generalized dice loss is more popular, which is defined as

$$J_{DM} := 1 - 2\frac{\sum_{j=1}^{N_c} \alpha_j \mathrm{y}_j \hat{\mathrm{y}}_j}{\sum_{j=1}^{N_c} \alpha_j (\mathrm{y}_j + \hat{\mathrm{y}}_j} \tag{3.50}$$

where α_j are class weightings, often chosen as the inverse volume of class j [97, 467]. The dice coefficient is also commonly used as an evaluation metric for segmentation tasks. Besides dice coefficient variants, classifications metrics such as accuracy, sensitivity, and specificity are often used as metrics for evaluating segmentation problems.

Regression. For a regression task, the goal is to predict a continuous value instead of a binary or nominal output. While digital processing requires discretization, the outputs for a regression task are much more fine-grained. In terms of network design, we can use architectures similar to those used for classification problems. Only the output layer needs to be changed where we employ linear neurons without an activation function.

A suitable loss function for this problem punishes the amount of deviation from the true value. A popular example is the mean squared error (MSE) which is defined as

$$J_{MSE} := \frac{1}{N_b} \sum_{i=1}^{N_b} (\mathrm{y}_i - \hat{\mathrm{y}}_i)^2 \tag{3.51}$$

where N_b is the number of examples being evaluated. Squaring the error leads to increased punishment of large deviations from the desired value, which can help faster convergence. As an alternative, the mean absolute error (MAE) can be used which is defined as

$$J_{MAE} := \frac{1}{N_b} \sum_{i=1}^{N_b} \|\mathrm{y}_i - \hat{\mathrm{y}}_i\| \tag{3.52}$$

where N_b is the number of examples being evaluated. As the MSE gives more weight to outliers, the MAE can lead to different results after optimization. The MAE is also useful as a metric for evaluation as it reflects the deviation from the target values without a squared distortion.

Another useful metric for evaluation is Pearson's correlation coefficient (PCC). The PCC is defined as

$$PCC := \frac{\sum_{i=1}^{N_b} (\mathrm{y}_j^i - \bar{\mathrm{y}}_j)(\hat{\mathrm{y}}_j^i - \bar{\hat{\mathrm{y}}}_j)}{\sqrt{\sum_{i=1}^{N_b} (\mathrm{y}_j^i - \bar{\mathrm{y}}_j)^2} \sqrt{\sum_{i=1}^{N_b} (\hat{\mathrm{y}}_j^i - \bar{\hat{\mathrm{y}}}_j)^2}} \tag{3.53}$$

where N_b is the number examples being evaluated, $\bar{y_j}$ is the mean predicted value for output j and $\bar{\hat{y}_j}$ is the mean target value for output j. This metric provide a relative estimate of how similar predictions and targets are without depending on a unit. As an alternative, the MAE can be normalized to achieve independence from the target's unit and range. When dividing the MAE by the target's standard deviation, we obtain the relative MAE (rMAE) [55].

Regression is a typical task for deep learning applications in the context of computer-assisted interventions, for example, for force estimation, pose estimation, or tracking. For diagnostic tasks, regression is employed for estimating quantitative tissue parameters such as the size of the left cardiac ventricle.

3.4 Summary

In this chapter, we introduced the fundamentals of neural networks and deep learning. Deep learning is a method to learn an unknown function from data. While optimization is used to find deep learning model parameters using a training dataset explicitly, the overall goal is to achieve generalization. This property describes a model's ability to perform well on new data that was not part of the training process. The deep learning problem can be extended to a bilevel optimization problem, where hyperparameter selection is incorporated as well. The hyperparameter search space is usually designed by a domain expert and depends on prior knowledge.

Neural networks are the basis of most modern deep learning methods. In their standard version, fully-connected neural networks learn a nonlinear transformation of a feature

vector for predicting an example's class or a continuous value. The feature vector's nonlinear transformation is comprised of linear matrix multiplication, where matrix entries are learnable parameters, followed by a nonlinear activation function such as a rectified linear unit.

Medical images are difficult to process when represented by a feature vector. Therefore, neural networks have been extended for directly processing images using convolutions instead of matrix multiplications. Here, the convolution's filters are learnable parameters. Also, medical image data can have a temporal component, for example, in terms of a series of images capturing a process, including motion or several images for longitudinal analysis. For this type of data, recurrent neural networks are often used. Temporal information is aggregated through recurrent connections and sequential processing. In particular, gating mechanisms are effective for capturing long-term information within sequential data.

Neural networks are typically trained using a labeled training dataset and gradient descent. The neural network's parameters are optimized such that they minimize a loss function. In general, this optimization process is difficult as the problem is nonconvex, and the actual goal of generalization is not explicitly captured in the process. Recent gradient descent algorithms such as Adam improve the optimization procedure by using adaptive learning rates and exponential moving averages of the gradients' statistical moment estimates. Regularization methods such as dropout, batch normalization, and data augmentation can also be used to improve generalization. For learning tasks such as classification, segmentation, and regression, different loss functions and evaluation metrics are used for assessing generalization performance.

4 Multi-Dimensional Deep Learning Methods

In this chapter, we adapt and propose a plethora of methods for deep learning-based medical image analysis in the context of multi-dimensional data and different data representations. A method's suitability for multi-dimensional data can be characterized by two requirements. First, the method needs to be scalable to lower and higher dimensions, for example, for considering multiple spatial data dimensions. Second, the method needs to be well-tailored to the different dimension's characteristics. For example, spatial, short-term temporal, and long-term temporal dimensions might benefit from a different treatment. As a result, deep learning methods that are used for exploring multi-dimensional data characteristics need to be both adaptable and well-tailored for the problem at hand.

The first set of methods we consider are multi-dimensional CNNs. Traditionally, CNNs have been used for 2D natural image data. As described in Chapter 2, the discrete convolution can be easily extended from 1D to 4D, making CNNs very versatile. Yet, architecture design is very challenging as naive extensions from lower to higher data dimensions are accompanied by substantial increases in model parameters and computational cost. We address this problem by careful handcrafting of architectures, automated architecture search, and multi-dimensional transfer learning strategies. While useful for many different problems, convolutions are not immediately suitable for small data dimensions that are part of 2.5D or 3.5D data. For this purpose, we propose different types of Siamese-like CNNs that can be applied to a large set of problems where one of the dimensions only encompasses two samples. In particular, we introduce a novel attention-based information exchange mechanism for two-time point data. Furthermore, CNNs treat all data dimensions equally, which might not be optimal in some cases. For this reason, we consider recurrent-convolutional models that process the temporal data dimension differently, using learnable gating mechanisms. In this context, we propose a novel convolutional-recurrent model that demonstrates promising results across multiple applications.

Throughout this chapter, we describe general architecture concepts that are not necessarily tied to a specific implementation or application. We implement these concepts for different applications in Chapter 6.

4.1 1D, 2D, 3D and 4D CNNs

Convolutional neural networks have been studied extensively for medical image analysis. As elaborated in Chapter 5, a key component that is missing is a multi-dimensional perspective. In other words, how should CNNs be transferred between different data

dimensions? We propose a number of methods to address this question, starting at 1D and 2D CNNs. Due to the low data dimensionality, we propose an automatic search strategy to obtain suitable architectures that are effective both for 1D and 2D data. The largest amount of CNNs are designed and proposed for 2D data, as this is the standard natural image dimensionality. Therefore, we address the question of how to extend 2D CNN concepts to 3D next. Here, we also propose a multi-dimensional transfer learning approach, which allows for effective architecture reuse in higher dimensions. Last, we consider the extension of 3D CNNs to 4D, which is a field that has hardly been addressed.

4.1.1 Neural Architecture Search on Low-Dimensional Data

Designing CNNs for lower-dimensional 2D image data has been addressed extensively. Also, the reduction to 1D is straightforward as one dimension can be removed from all mathematical operations. Moving from 2D to 1D also leads to lower computational requirements and fewer trainable model parameters. Thus, 1D and 2D CNN design for medical learning problems is generally not problematic as 2D approaches can be readily adapted from the natural image domain. However, the property of being computationally cheap opens up new opportunities for CNN model design that could also benefit more challenging, higher-dimensional problems.

Manual feature engineering has been largely replaced by deep learning approaches for numerous medical, image-based learning problems over the last few years. As most CNN architectures are designed manually, there has been a shift from handcrafting features to handcrafting architectures. CNNs themselves are often difficult to design, and it is unclear what kind of architecture is suitable for which learning problem. Thus, the next intuitive step is to move from handcrafting architectures to learning architectures.

In Chapter 3, we introduced the bilevel optimization problem given in Equation 3.4. Formally, the design choices for the architecture can be considered a subset $h_M^A \subset h_M$ of hyperparameters that need to be selected. As described in the previous chapter, the machine learning engineer usually chooses hyperparameters h_M^A.

Approaches for learning h_M^A are often termed *neural architecture search* (NAS). Typical NAS approaches include grid search, genetic algorithms, bayesian optimization, or random search [232]. Recently, reinforcement learning (RL) methods have been proposed where a recurrent controller is trained to predict an architecture's structure by maximizing the architecture's expected validation performance as a reward [602]. This approach has been successful for 2D image classification problems, however, the amount of computing resources required are often enormous [303, 603]. Early NAS approaches required thousands of GPU hours for learning an architecture with 2D image data [377].

The concept of NAS is very promising for the medical image domain as there is a vast amount of imaging modalities and learning problems that require architecture design. However, the time and resource requirements of NAS are problematic for medical image data, which is often 3D or 4D in nature [291].

We propose an efficient NAS approach for segmentation with multi-dimensional medical image data. To overcome long architecture search times, we perform the search on lower-dimensional data, which leads to shorter search times. Then, we transfer the learned architecture to the higher target dimension. So far, NAS approaches have

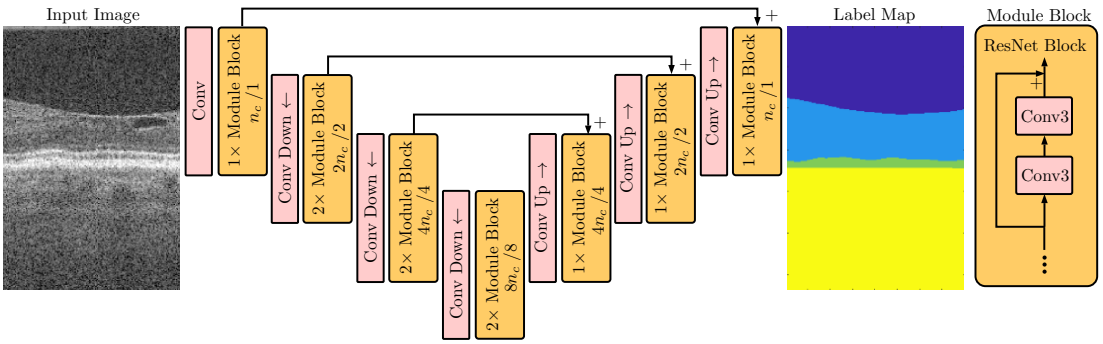


Fig. 4.1: The baseline segmentation architecture for 1D and 2D image data, shown for the example of retinal layer segmentation. We employ ResNet blocks as the main processing units. n_c is the base number of feature maps that is doubled every time we downsample. $/2$ indicates that the original images' spatial dimensions have been reduced with a stride of $r = 2$.

largely been explored in the natural image domain, not for medical learning problems. Therefore, we propose a strategy for typical U-Net-like [409] architectures, where we learn the submodules within the architecture at each level. Given the relatively cheap computational requirements for the processing of lower-dimensional data, we investigate the approach for 1D and 2D image data and the task of medical image segmentation.

Baseline Model. As a baseline we use a U-Net-like model, as described in Section 3.3.6. The model takes a 1D or a 2D image as its input and predicts a segmentation map with the same size as the input. For the long-range connections, we use summation, following [575]. Inside the network, we use ResNet [193] blocks, which we introduced in Section 3.3.2. Convolutions use a kernel size of 3, following Simonyan et al. [450], and extensions from 1D to 2D are performed by extending all kernels isotropically by an additional dimension. At each level, we employ one or several ResNet blocks. Thus, the architecture shown in Figure 4.1 is a slightly modified U-Net with ResNet blocks and summation for the long-range connections.

Neural Architecture Search. We follow the general framework of neural architecture with a recurrent LSTM controller and a reinforcement learning-based approach for learning new architectures [602]. The general idea of this concept is shown in Figure 4.2. The controller is an RNN-based architecture that provides probabilistic predictions of a CNN architecture in a sequential way. The controller's parameters are denoted as w_M^c. Given the controller's predictions, a single architecture is sampled with probability p_a. This child architecture is constructed, and its parameters w_M^m are trained for the task to be solved. After convergence, the child CNN's performance is determined by a reward metric R on a validation set. This metric is used to scale the policy gradient of p_a, which can then be used for updating the controller's parameters w_M^c.

In detail, the goal of training is to maximize the controller's expected reward which is defined as

$$J_P(w_M^c) = \mathrm{E}_{P(a_{1:T};w_M^c)}[R] \tag{4.1}$$

where $a_{1:T}$ is a set of actions that can be taken by the controller. The reward R is

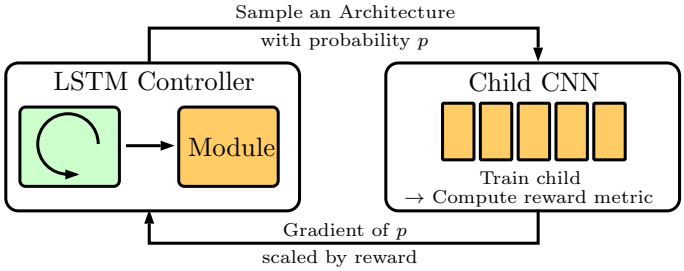


Fig. 4.2: The general concept of neural architecture search with a recurrent LSTM controller and reinforcement learning is shown.

not differentiable with respect to w_M^c, therefore a policy gradient method needs to be used. The REINFORCE method proposed by Williams [538] provides an empirical approximation of the gradient:

$$\nabla_{w_M^c} J_P(w_M^c) \approx \frac{1}{m} \nabla_{w_c} \log P(a_t | a_{(t-1):1}; w_M^c)(R_k - b_e) \tag{4.2}$$

Here, b_e is an exponential moving average of previous architecture's reward metrics, which is used for reducing variance during training. N_{sample} is the number of architectures that are sampled by the controller and trained to convergence afterward. T is the number of possible actions and thus represents the number of hyperparameter choices available to the controller.

While this concept has led to architectures that outperformed all handcrafted alternatives at that time [602], the procedure is very costly. For every controller update, N_{sample} CNN architectures need to be trained from scratch until convergence. To overcome this problem, efficient neural architecture search (ENAS) [377] has been proposed. The key idea behind the improvement is that child CNN architectures are not retrained from scratch at every iteration, but the trained weights are kept instead. Controller and child CNN are trained in an interleaved way where each is trained for one epoch while the other one's weights remain fixed. This strategy reduces computational effort by a factor of 1000 and has also led to competitive architectures [377]. In terms of implementation, this method is more challenging as all possible model configurations need to be implemented simultaneously. Given the current action a, the architectures connections are rerouted, and individual operations are activated or deactivated. Throughout the entire search process, all possible architectures and their weights are maintained.

ENAS U-Net. Next, we propose ENAS U-Net, an adaptation of the ENAS framework [377] for image segmentation tasks with a U-Net. To keep computational effort bounded, we simplify the architecture search space by keeping the general U-Net structure fixed and only learning new *module blocks*, similar to the micro search space in ENAS. The input/output and downsampling/upsampling convolutional layers also stay fixed. Considering our baseline architecture in Figure 4.1, we now learn a module block that replaces the ResNet block. For the module block search space, we let the controller learn the properties of several cells, where each cell contains 2 subcells. The cells' output is the summation of the subcells' output. For each subcell, the controller defines its input, which is the module input or another cell's output, and its operation. Similar to ENAS, we allow five basic operations for the controller to choose from: convolutions

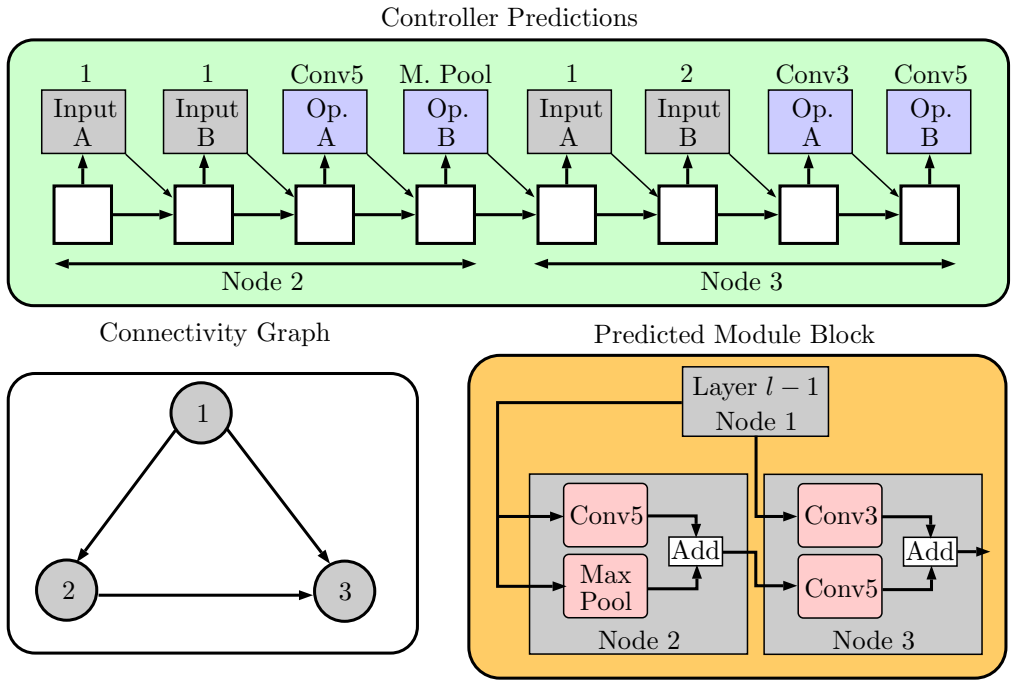


Fig. 4.3: An example module block prediction by the current controller is shown. The controller sequentially predicts both the connectivity and operations within the module block (top). The connectivity can be visualized by a directed acyclic graph (bottom left). The final implementation of the module block at layer l is shown right.

with kernel size 3 or 5, average- and max-pooling with kernel size 3 and the identity transform. We explore search scenarios with different numbers of cells to learn. Our controller setup for an example prediction of a module block is shown in Figure 4.3. Overall, we carefully engineer our search space to be small such that search times remain within a reasonable bound.

Training Strategy. We perform simultaneous training of model and architecture parameters as follows. First, an initial U-Net architecture is sampled based on random initialization. This architecture is trained by gradient descent for one epoch using a training set. Then, the current model weights are fixed, and the controller is trained for one epoch using a dice score reward, computed from a reward training set. Training is performed by following the REINFORCE algorithm described above. Then, the controller weights are fixed once again, a new architecture is sampled from the controller, and normal model training is performed using the training data set. Thus, the training procedure follows an interleaved schedule where all parameters are reused in each epoch. After training for N_e epochs, we fixed all weights and sample N_{sample} candidate architectures from the controller. Each architecture is evaluated on a validation set. The best-performing model is selected based on its validation performance. This model is trained from scratch with randomly initialized parameters, and finally, it is evaluated on a test set. In contrast, the baseline U-Net model is trained on the training set for N_e epochs. The hyperparameters for the baseline U-Net model are selected based on validation set performance. Finally, the model is evaluated on the test set. The training

set, reward training set, validation set and test set are disjoint.

We explore both searching for architectures on 1D data and 2D data. Thus, for both searches we obtain architecture hyperparameters h_M^{1D} and h_M^{2D} respectively. After the search, we evaluate the architecture found with a search on 1D data both as a 1D and as a 2D CNN architecture. We consider the machine learning models:

$$f_M^{1D}(h_M^{RN}) : \mathbb{R}^{n_d} \rightarrow \mathbb{R}^{n_d} \tag{4.3}$$

$$f_M^{1D}(h_M^{1D}) : \mathbb{R}^{n_d} \rightarrow \mathbb{R}^{n_d} \tag{4.4}$$

$$f_M^{2D}(h_M^{RN}) : \mathbb{R}^{n_w \times n_d} \rightarrow \mathbb{R}^{n_w \times n_d} \tag{4.5}$$

$$f_M^{2D}(h_M^{1D}) : \mathbb{R}^{n_w \times n_d} \rightarrow \mathbb{R}^{n_w \times n_d} \tag{4.6}$$

$$f_M^{2D}(h_M^{2D}) : \mathbb{R}^{n_w \times n_d} \rightarrow \mathbb{R}^{n_w \times n_d} \tag{4.7}$$

where h_M^{RN} are the architecture hyperparameters of the baseline model with ResBlocks. As described above, the 1D CNN found by ENAS U-Net is extended to 2D by isotropically extended all mathematical operations of the CNN to 2D. Architectures found through NAS are always trained from scratch.

Summary. Lower-dimensional CNNs, in particular, 1D CNNs come with the advantageous property of being computationally cheap. This makes architecture design comfortable as existing 2D architecture from the natural image domain can be easily adopted. At the same time, the low computational requirements open up a new path for architecture design by automatically learning the architecture's structure with an additional optimization level. Typically, neural architecture search is severely limited by its immense computational requirements of up to thousands of GPU hours. We explore a neural architecture search strategy that is particularly efficient by searching on low-dimensional data before extending architectures to higher dimensions. This approach is promising for the medical imaging domain, as there are many problems that require problem-specific architecture design. Therefore, we propose an adaptation of the ENAS framework for segmentation problems, the most common problem in the medical image domain.

4.1.2 Extending 2D CNNs to 3D

When moving to higher data dimensions with CNNs, architecture design becomes more difficult as the natural increase in model size and trainable parameters becomes problematic. Automatically searching for CNN architectures becomes more difficult or even infeasible. Therefore, a lot of applications require handcrafted architecture design when using higher-dimensional data. One approach for architecture design is taking successful 2D CNN architecture concepts from the natural image domain and extending them to 3D. Thus, while taking inspiration from other architecture concepts, architecture hyperparameters $h_M^A \subset h_M$ are chosen by the machine learning engineer. In the following, we explore this approach and propose multiple 3D CNN architectures processing 3D spatial data $\mathrm{x} \in \mathbb{R}^{n_h \times n_w \times n_d}$ that can be employed for multi-output regression or classification problems.

The complete 3D CNN consists of several convolutional layers that represent a feature extraction stage and an output layer for regression or classification. Our principle

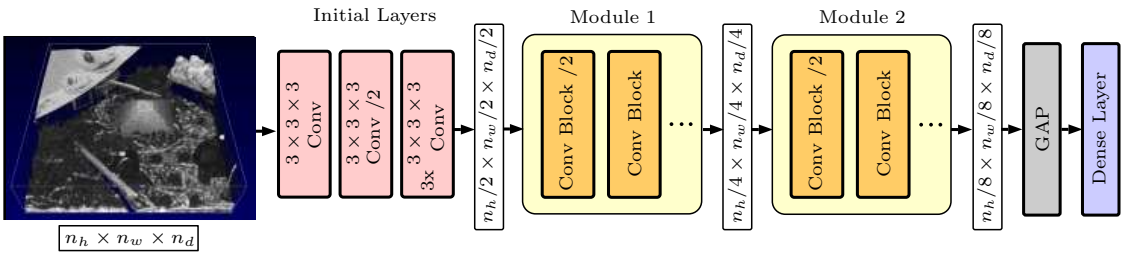


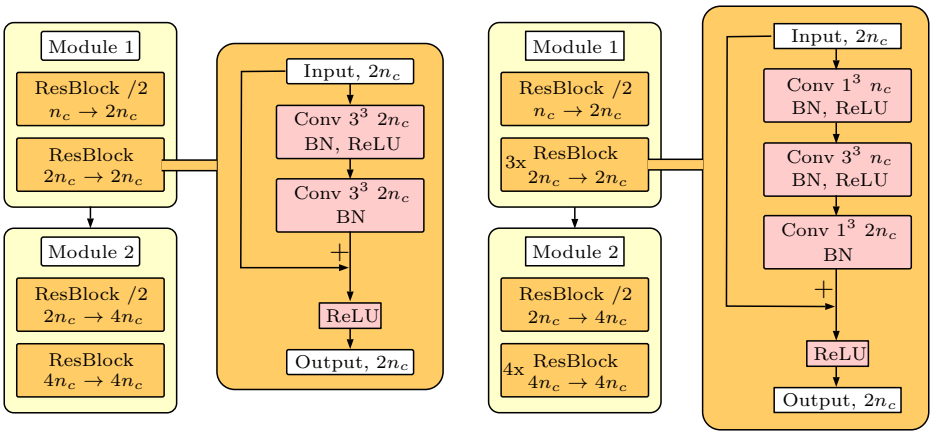
Fig. 4.4: The generic architecture we employ for our 3D CNNs. We show input volumes and targets for the example of position estimation from an OCT volume. The initial part, intermediate volume sizes, and the output part are identical for every architecture. The modules are individually designed for each specific architecture. All modules start with a convolutional block that reduces the spatial dimension by half with a stride of 2.

network design is shown in Figure 4.4. After the volumetric input, some initial layers follow, which are identical for all architectures we build. Immediately after the first layer, we halve the input's spatial dimension. We employ convolutional layers with stride two instead of the typical max pooling layer, following the idea of simplistic design [463]. Then, groups of architecture-specific layers follow, which we refer to as modules. At the module input, the first layer always reduces the input size by half in all spatial dimensions. Every architecture comes with two modules, representing our main feature extraction stage with the most model parameters and the largest influence on performance. After two modules, we apply global average pooling to reduce the current feature volume to a feature vector. This approach acts as a regularization, as described in Section 3.3.2. The feature vector is fed into the output layer that predicts the target for the respective task.

For the modules in Figure 4.4, we employ different types of architectures to explore how well 2D CNN design concepts transfer to 3D. Each model introduces a different additional property that leads to our design of IN3D, the main architecture we propose for 3D data processing. To maintain a fair comparison, we try to keep the architectures similar with respect to the number of trainable parameters ($4\,000\,000$) and features learned.

To keep architecture design straight forward, we follow previous design principles from the 2D domain. Simonyan et al. [450] showed that smaller kernel sizes are preferable for CNNs, which is why we only employ $3 \times 3 \times 3$ filters for feature learning and $1 \times 1 \times 1$ filters for changing feature map sizes. Moreover, we increase the number of feature maps in our modules each time the spatial feature dimensions are halved. Additionally, we employ batch normalization before every activation to reduce the covariate shift [214]. The activation functions are of type ReLu [175].

RN3D-A is an architecture that we base on state-of-the-art 3D segmentation CNNs such as [81, 575] to provide a meaningful comparison to our other models. Several blocks of this architecture are joined to modules, as shown in Figure 4.5. As described in Section 3.3.2, residual connections are frequently used in the 2D image domain with numerous variations [475, 576] and recently the concept was employed for 3D prostate segmentation [81]. Therefore, we see this model as a baseline architecture, reflecting



(a) RN3D-A (b) RN3D-B

Fig. 4.5: The architecture of the RN3D-A and RN3D-B model is shown. For RN3D-A, each module contains two residual blocks where the first block in each module reduces the spatial dimension by half and increases the feature map dimension by a factor of two. Conv 3^3 indicates a filter size of $3 \times 3 \times 3$. n_c is the number of feature maps in a block or layer.

the application of 2D design principles in the 3D image domain. Note that this model is expensive regarding its number of parameters as it does not employ downsampling in the number of feature maps, which is introduced next. Therefore, the network comes with a smaller depth to maintain a similar amount of parameters.

RN3D-B is a model that extends the concept of residual blocks from RN3D-A by adding $1 \times 1 \times 1$ convolutions for downsampling and upsampling of the feature map dimension, as shown in Figure 4.5. Often, this idea is described as a *bottleneck*. Furthermore, the method should be distinguished from *spatial* downsampling, which acts on the images' width, height, and depth and helps to build more abstract features representations. Reducing the feature map dimension follows the idea of dimensionality reduction, which assumes that most of the input's information can be preserved in a lower-dimensional embedding. This concept was also used in the original 2D ResNet architecture [193]. However, to our knowledge, it has not been employed for 3D CNN learning tasks. This concept is particularly important for costly 3D CNNs as this method reduces the number of parameters and computational effort of the model. Note that this design principle allows for a deeper model with more layers than RN3D-A.

We propose **IN3D** as a new 3D CNN architecture which is inspired by Inception-ResNet [475]. We make use of the previous models' properties and additionally introduce the concept of multi-path convolutional blocks, as shown in Figure 4.6. As outlined in Section 3.3.2, the multi-path approach is motivated by the idea of feature extraction at different scales, which is expected to yield more representative features [476]. Note that this architecture is difficult to design, in particular, as more design choices need to be made. We address this problem by simplifying IN3D without taking away its core concepts. Compared to [475], we employ a single type of Inception module with the same number of feature maps for all filters in each path. Compared to our other models, we individually choose each block's width, and we augment the architecture

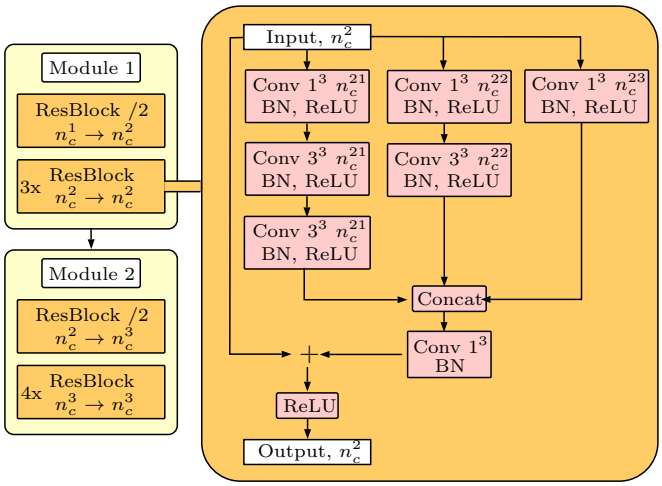


Fig. 4.6: The architecture of the IN3D model is shown. Each module contains four and five residual blocks, respectively, where the first block in each module reduces the spatial dimension by half and increases the feature map dimension by a factor of two. Conv 1^3 and 3^3 indicate filter sizes of $1 \times 1 \times 1$ and $3 \times 3 \times 3$, respectively. The final $1 \times 1 \times 1$ convolution in each inception block recovers the original feature map size. n_{c_i} is the number of feature maps in a block or layer.

with long-range residual connections.

The idea of long-range residual connections is inspired by Yu et al. [575], where connections between the same feature map stages are applied in a U-net-like [409] encoder-decoder network. We adapt this idea by transferring features between different feature map scales. For comparison, we also use the original idea of U-net for feature transfer [409]. While residual connections perform an addition operation when features are fused, U-net concatenates the features to a larger feature map. For the latter, we perform a subsequent $1 \times 1 \times 1$ convolution that reduces the feature map size back to the original size after concatenation. In this way, the network can learn which combination of high- and low-level features is needed. The idea behind this approach is that a lot of tasks require both local and global features. Both skip connection approaches are shown in Figure 4.7.

RX3D is similar to the Inception idea with a multipath architecture that is inspired by [554], see Figure 4.8. The key idea is to utilize all of the above models' ideas with simplified design principles. The multiple paths idea from Inception is adopted by splitting up the single convolution path from RN3D-B. The number of paths is referred to as cardinality n_{cd} which is considered the key hyperparameter to choose for this type of architecture [554]. The resulting architecture is easy to tune as all paths are identical compared to Inception, where each path is carefully tuned individually. Therefore, the key difference between RX3D and IN3D is a simpler architecture design for the former.

Summary. We propose four different architectures for 3D spatial images and regression or classification problems. RN3D-A with residual blocks is a baseline as it is similar to previous 3D CNNs [575]. For RN3D-B, we introduce the use of downsampling in the feature map dimension for more effective feature representation with the same amount

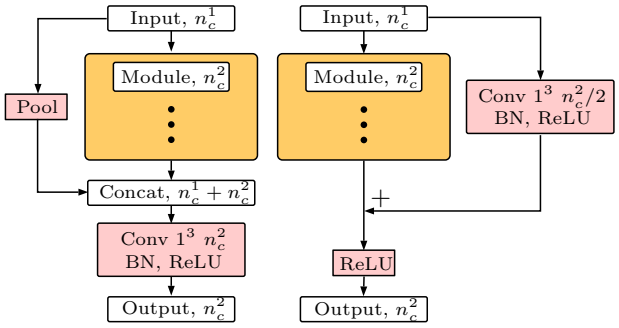


Fig. 4.7: Two types of long-range connections for the modules of IN3D are shown. Left, the transfer of features between stages is shown with a concatenation of features from different levels. Right, feature transfer through a long-range residual connection is shown. n_{c_1} denotes the number of input feature maps, n_{c_2} the number of output feature maps from the module. Pool indicates $2 \times 2 \times 2$ average pooling to match the module's spatial dimensionality reduction. Conv 1^3 indicates $1 \times 1 \times 1$ convolutions for adjustment of the number of feature maps. For the residual connection, the convolution is applied with a stride of two to match the module's spatial dimensionality reduction.

of parameters. We augment IN3D further with multi-path blocks and long-range residual connections for improved performance. Lastly, RX3D shows how a network with little design effort compares to our similar but carefully tuned IN3D architecture. These architectures highlight how different design principles affect performance. Compared to the automated architecture learning approach for 1D data, our approach for 3D CNNs requires a lot of architecture engineering. This is traded off for more control in the design process, interpretable architecture variations, and computationally efficient models.

4.1.3 Multi-Dimensional Transfer Learning

While carefully handcrafting efficient architectures for 3D deep learning is a viable approach, there are some downsides such as significant engineering effort. Another downside is the difficulty of employing transfer learning, which is very popular in the medical image analysis domain [447]. Here, the idea is to copy a 2D CNN architecture that was trained for a task in the natural image domain and (partially) retrain it for a target task in the medical image domain. As natural image datasets are generally large, the pretrained model often learned very generic feature representations that can be effectively reused for a different task. In this type of setting, we reuse architecture hyperparameters $h_M^A \subset h_M$ that have been chosen for other problems. For 3D deep learning problems, this concept is difficult to use as 3D image data is rare in the natural image domain. One approach for enabling transfer learning in 3D or higher data dimensions is to transfer 2D architectures to 3D by reusing weights. In the following, we propose an approach for this type of multi-dimensional transfer learning for 2D spatial and 3D spatio-temporal image data.

2D CNN Approaches. Similar to the extension of spatial 2D CNNs to 3D, we consider a regression or classification task where images are assigned to a category or

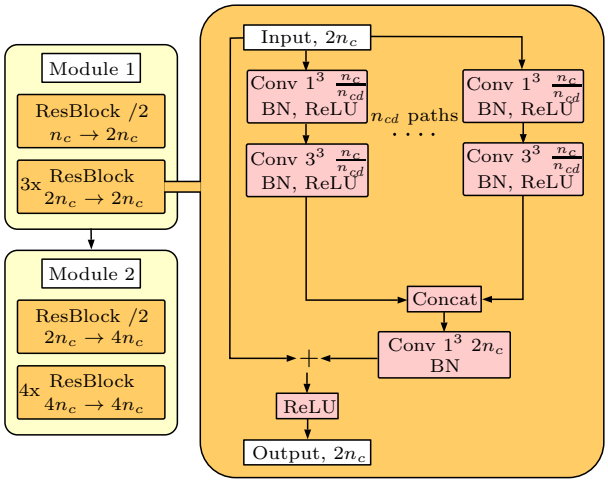


Fig. 4.8: The architecture of the RX3D model is shown. Each module contains four and five residual blocks, respectively, where the first block in each module reduces the spatial dimension by half and increases the feature map dimension by a factor of two. Conv 1^3 and 3^3 indicate filters sizes of $1 \times 1 \times 1$ and $3 \times 3 \times 3$, respectively. n_c is the number of feature maps in a block or layer.

continuous values are regressed from the image. Instead of designing efficient 2D and 3D CNNs based on existing architecture concepts, we directly make use of already proposed architectures. As outlined in the previous section, this approach is generally problematic for medical image analysis tasks, as medical datasets are usually much smaller and there is a substantial risk of overfitting. This problem can be partially circumvented by making use of transfer learning. Here, the CNN is pretrained on the ImageNet dataset, which contains $1\,200\,000$ images that have to be classified into $N_c = 1000$ classes. Then, the CNN is trained for the target task. For this purpose, the CNN's last layer is removed and replaced by a layer with the target number of outputs.

We consider a pool of pretrained CNN architectures, including the previously introduced ResNet [193], ResNeXt (RX) [553] as well as the newer architectures DenseNet (DN) [208] and Squeeze-and-Excitation Networks (SE) [205]. The DenseNet model is focused on efficiency even more than previous architectures, as it relies on heavy feature reuse in each block. The idea of this architecture is to use all features from previous layers for current layer l. The output of the l^{th} layer is computed as

$$\mathrm{x}^l = \mathcal{H}([\mathrm{x}^0, \mathrm{x}^1, ..., \mathrm{x}^{l-1}]) \tag{4.8}$$

where $\mathcal{H}$ is the layer's transformation, including convolutions. In this way, the output feature maps of all previous layers are reused. This allows for overall significantly smaller feature maps and a reduced number of parameters. Due to this structure, the number of feature maps grows linearly with a growth rate g_k. Furthermore, the architecture uses compression layers in between dense blocks in order to keep the feature map sizes low. A 1×1 convolution downsamples the feature maps by a factor $n_{c_{red}}$. This architecture also makes use of bottlenecks, as described for the ResNet model in the previous section.

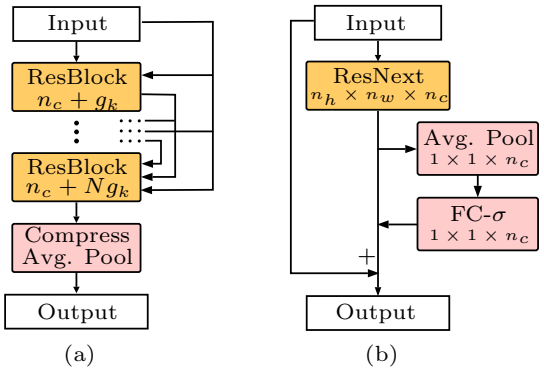


Fig. 4.9: The DenseNet and the SENet architecture are shown. FC-σ is an FC-layer with a sigmoid activation function. Note that the original implementation of SENet uses a bottleneck with 2 FC-layers to keep the number of trainable parameters in a moderate range [205].

The SENet principle is an architecture augmentation that can be added to any existing design principle such as ResNets or DenseNets. The proposed SE-blocks perform a recalibration of a CNN's feature map. In a normal convolutional layer, new feature maps are computed by individually convolving each previous map with a separate filter, followed by averaging. Thus, the combination is implicit by summation. Instead, SE-blocks explicitly learn a reweighting of a layer's feature maps. First, the feature maps are pooled into a feature vector with spatial GAP. Then, a sigmoid FC-layer performs a nonlinear transformation of the feature vector. The same-sized output vector is then multiplied with the original feature tensor, thus, reweighting each feature map. For large feature map sizes n_c, the authors also use a bottleneck-like setup for the FC-layer where two subsequent FC-layers with $n_c/n_{c_{red}}$ neurons and n_c neurons, respectively, are used. The DenseNet and SENet architecture principle are shown in Figure 4.9.

3D CNN Approaches. As a next step, we propose an extension of the previously described state-of-the-art 2D CNNs to 3D for a 3D spatio-temporal learning task. Thus, instead of processing a single 2D image, the CNN receives a temporal sequence of 2D images. While the data is higher-dimensional, the 3D image tensor still consists of the same 2D images. This property opens up the idea of directly reusing 2D CNN architectures for 3D data. The straightforward extension can be performed by extending each convolutional kernel isotropically with an additional dimension. The identical architecture also opens the opportunity to perform *multi-dimensional transfer learning*. Thus, we propose to initialize each 3D kernel with its 2D counterpart by copying the 2D kernel multiple times. In detail, the 3D convolutional kernels are initialized by copying the 2D kernels that were pretrained on ImageNet of size $K_{2D} \in \mathbb{R}^{k_h \times k_w \times k_c}$ exactly k_d times into the new kernel of size $K_{3D} \in \mathbb{R}^{k_h \times k_w \times k_d \times k_c}$. Intuitively, this should work well as a 3D convolution on a 3D stack of 2D slices now has the same effect as applying several, stacked 2D convolutions to the 3D image stack.

However, the copied weights still need to be rescaled. Considering an individual voxel's value $\hat{i}_{v_j} = 1$ being computed by a 3D kernel that is convolved over a tensor only containing the value 1, the resulting voxel value would be $\hat{i}_{v_j} = k_h k_w k_d$. With

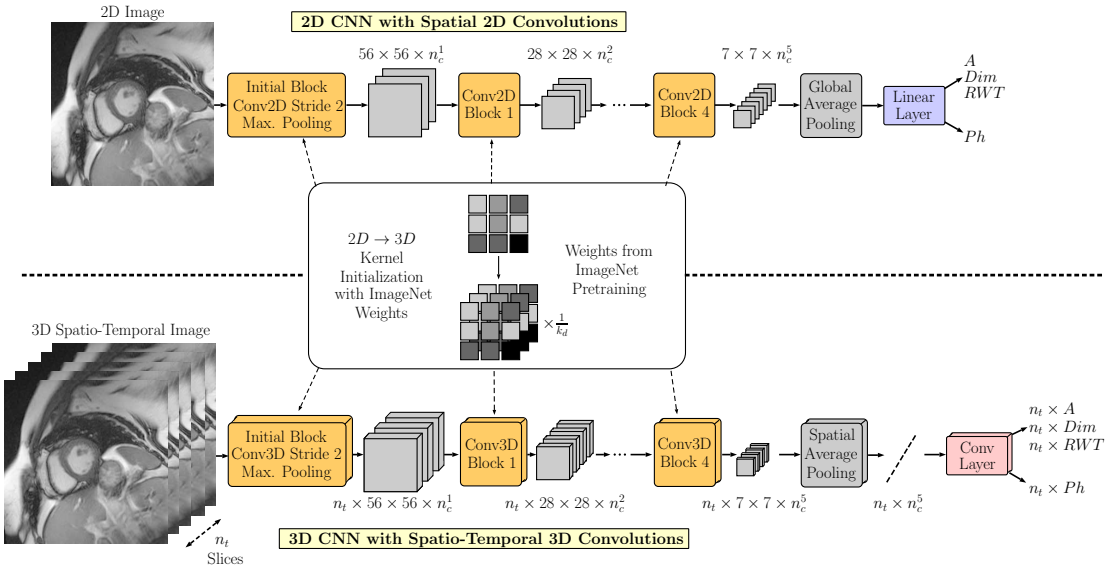


Fig. 4.10: Overview of our approach. We use both 2D CNNs with 2D slices (top) and 3D CNNs with temporally stacked slices (bottom). Both are initialized with pretrained weights from ImageNet. The initial and Conv2D/Conv3D blocks have a different structure based on the respective architecture.

2D processing and no inter-slice processing, the resulting value would be $\hat{i}_{v_j} = k_h k_w$. During the initial phases of training, this would lead to exploding values. To recover consistent value ranges, we therefore multiply the 3D kernels' weights by a factor $1/k_d$. We show that the proposed 3D extension approach in Figure 4.10 for the example application of LVQ. Here, the 2D image data consists of spatial 2D MRI slices, and the 3D spatio-temporal data consists of a temporal sequence of 2D MRI slices throughout a cardiac cycle. We also employ our method for transfer learning with weight initialization in the context of Siamese CNNs, which we introduce in the next section.

The 3D CNN input is of size $\mathrm{x} \in \mathbb{R}^{n_t \times n_h \times n_w}$ where n_h and n_w are the sizes of the spatial slice dimensions and n_t is the number of temporal slices. Throughout the entire network, we do not change the temporal dimension as we produce n_t predictions for n_t input slices in a single forward pass. For this purpose, we replace the linear output layer by a convolutional layer with kernel size 1, which is able to handle arbitrarily-sized inputs.

Table 4.1 shows different CNN architectures in their 2D and 3D configuration and the associated memory requirements and trainable parameters. Notice that our carefully handcrafted 3D CNNs introduced in the previous section came with 3 to 6 million parameters while the straightforward extension performed here leads to 10 to 20 times the number of parameters. Also, the memory requirements often exceed currently available hardware, where the most commonly used GPUs have a memory size of 11 GB. Due to the huge increase in memory and computational requirements, we only consider 3D variants of the smaller CNNs in our pool.

Summary. We propose a method for straightforward extension of 2D spatial CNNs to 3D spatio-temporal CNNs. As outlined in the previous section, this approach leads to sub-

Tab. 4.1: Overview of several different CNN architectures for the extension from 2D to 3D. We show the number of trainable parameters n_{params} in million as well as memory requirements (Memory) in gigabytes for a batch size of $N_b = 10$ and a temporal dimension of $n_t = 10$.

	2D		3D	
	n_{params} (M)	Memory (GB)	n_{params} (M)	Memory (GB)
DN-121	6.969	3.16	11.26	15.29
DN-161	26.51	5.78	39.47	27.88
DN-169	12.59	3.74	18.57	18.07
SE-RN-50	26.07	3.27	48.72	16.02
SE-RN-101	47.31	5.05	90.02	24.64
SE-RN-152	64.80	7.21	124.0	35.22
SE-RX-50	25.54	4.14	28.39	20.31
SE-RX-101	46.93	6.31	52.29	30.81
SENet	113.1	11.21	170.2	54.49

stantial increase in memory, computational requirements, and trainable parameters. We try to overcome the problem of increased trainable parameters with a multi-dimensional transfer learning approach where we copy pretrained, scaled 2D kernels into a 3D CNN.

4.1.4 Extending 3D CNNs to 4D

The next step we consider is the extension of 3D CNNs to 4D. Intuitively, this is similar to the extension from 2D to 3D, with the addition that all issues are aggravated. Computational costs and memory requirements are even higher. For example, extending a 2D CNN with n_{params} parameters to 4D by isotropically expanding the kernels would lead to an architecture with n_{params}^2 parameters. Thus, carefully designed solutions are required. Thus, similar to handcrafted 3D CNN architecture design, architecture hyperparameters $h_M^A \subset h_M$ are manually selected. Furthermore, current deep learning frameworks lack native support of deep learning tools such as 4D convolutional layers. In this work, we consider several different approaches for extending CNNs to 4D. We start with an extension where we employ normal 4D convolutions. Then, we build a more efficient variant using factorized convolutions. Finally, we take a different approach to the extension problem by considering multi-path architectures. These methods can be employed for 4D data, which typically consists of a temporal sequence of image volumes: $\mathrm{x} \in \mathbb{R}^{n_t \times n_h \times n_w \times n_d}$.

CNN4D. For this architecture, we replace 3D convolutions by their 4D counterpart. The mathematical extension of a discrete convolution to N_d dimensions is described by Equation 3.10 in Chapter 3. In a 4D convolutional layer, the kernel $K^l \in \mathbb{R}^{k_t \times k_h \times k_w \times k_d \times k_c}$ of layer l is applied to feature maps $\mathrm{x}^{l-1} \in \mathbb{R}^{n_t \times n_h \times n_w \times n_d \times n_c}$, excluding the batch dimension. n_t is the size of the temporal dimension, n_h, n_w, and n_d are the spatial extent of the feature map, and n_c is the channel dimension of the feature map. Currently, there are no native 4D convolution operations available in standard environments such as PyTorch and Tensorflow. To keep the 4D convolution as efficient

as possible, we implement a custom version in Tensorflow, which uses the native 3D convolution operation inside of two loops. The operation can be described as

$$(K^l * \mathrm{x}^{l-1}) = \sum_{i}^{k_t} \sum_{j}^{n_t} \mathrm{Conv3D}(K^l(i), \mathrm{x}^{l-1}(j)) \tag{4.9}$$

with correct padding and a stride of one assumed. Based on this layer, we build different implementations of 4D CNNs. This includes a ResNet-based RN4D architecture and a DenseNet-based DN4D architecture. Both architectures share a similar structure with an initial feature map size of $d_{c_{init}}$ that is doubled each time the spatial dimensions are reduced by a convolutional layer with stride 2. As n_t is usually small for most of our problems, the temporal dimension is not reduced by strides.

facCNN4D. A more efficient variant of RN4D or DN4D can be constructed by making use of factorized convolutions. In each ResBlock or DenseBlock, each convolution is split into two convolutions with spatial kernels $K_S^l \in \mathbb{R}^{1 \times k_h \times k_w \times k_d \times k_c}$ and temporal kernels $K_T^l \in \mathbb{R}^{k_t \times 1 \times 1 \times 1 \times k_c}$. This modification leads to a reduced number of parameters and decomposes spatial and temporal computations. Thus, the architecture is more efficient but does not have the same representational power as a CNN4D model as the decomposed convolutions can only represent separable 4D kernels.

MP-CNN4D. Instead of decomposing spatial and temporal processing in every single layer, as performed for facCNN4D, we can also decompose spatial and temporal processing globally. For MP-CNN4D, we first perform spatial processing for several layers while reducing all spatial dimensions. Then, the smaller 4D tensor is once again processed by a 4D CNN. In this way, costly 4D spatio-temporal processing is only performed after the spatial dimensions have been reduced already.

In detail, spatial processing is performed by using a multi-path 3D CNN where each path processes one 3D volume within the sequence individually. For efficiency, all 3D CNN paths share their weights. Intuitively, learning the same features for each volume should be effective as the volumes should share a lot of similarities. Afterward, higher-dimensional temporal dependencies can be captured by the 4D CNN.

Summary. Our three 4D CNN architecture concepts are shown in Figure 4.11 for a ResNet architecture. Overall, 4D CNNs are very challenging to design as their high-dimensional nature requires a large amount of memory and computational resources. Also, an extension by an additional dimension with normal convolutional layers would lead to very high numbers of parameters, risking overfitting. Therefore, we also consider two more efficient variants of 4D CNNs. The first variant employs decomposed spatial and temporal convolutions in each layer throughout the entire network. The second variant splits spatial and temporal processing globally by performing spatial processing first to obtain spatially smaller feature maps. Then, the smaller volume sequence can be processed by a 4D CNN for still capturing full 4D spatio-temporal context.

4.2 2.5D and 3.5D CNNs

Siamese CNNs. A lot of deep learning applications in biomedical research come with data that cannot be assigned to 2D or 3D data as one of the data dimensions is very small,

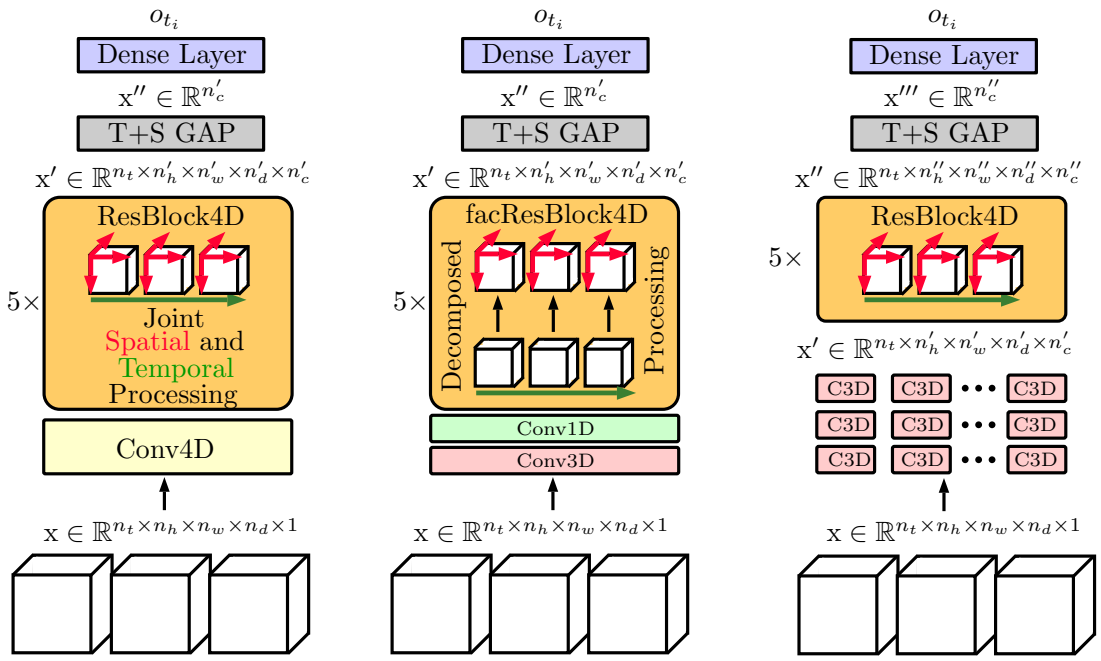


Fig. 4.11: The three 4D CNN concepts we propose for 4D spatio-temporal data. Here, we show the concepts in combination with ResBlocks. Each architecture receives a sequence of volumes as input. T+S GAP refers to GAP over the temporal and spatial dimensions. Red indicates spatial processing, green indicates temporal processing, and yellow indicates joint processing.

usually with a size of two. In this case, the data is referred to as 2.5D or 3.5D if 2D or 3D images from two states or two time points are used. Thus, input examples have a size of $\mathrm{x} \in \mathbb{R}^{2 \times n_h \times n_w \times n_d}$ for the case of 3D volumes. An example is the detection of disease progression between two images taken at two different time points. Intuitively, considering 2.5D data as 3D does not appear reasonable when using CNNs as convolving over a dimension of size two with a kernel size $k \geq 2$ simply results in a fully-connected structure. Instead, another intuitive approach is to treat the two states similar to colors. Here, the two states can be stacked in the CNN's input channel dimension. We will demonstrate that this approach can be seen as a special case of a class of Siamese CNNs that we propose.

Originally, Siamese CNNs were introduced in the natural image domain for tasks such as person tracking [277, 580]. The idea is to use two parallel CNNs where each path processes one image. At some point, the two processed data representations are fused, for example, in a loss function that expresses similarity between the two input images. There are a lot of different possibilities for modifying this concept and adapting it for different applications. Next, we introduce a Siamese CNN, where we consider different fusion strategies for the two paths, parameter sharing, and transfer learning for this particular type of architecture.

Our Siamese CNN architecture is shown in Figure 4.12. Siamese CNN architectures

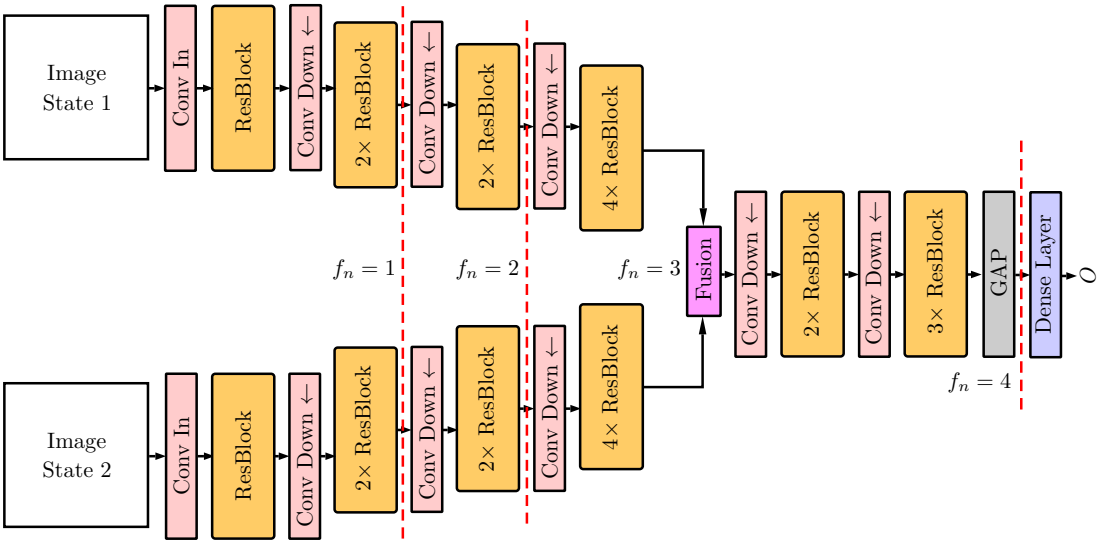


Fig. 4.12: The Siamese CNN architecture we propose. The model takes two images representing two different states as input. We consider both 2D and 3D images with 2D and 3D CNN variants. In the initial part, the two images are processed independently up to a fusion point f_n. At this point, the feature maps are aggregated and processed jointly. We consider different fusion points. Here, $f_n = 3$ is shown. At the output, GAP is applied to the remaining feature maps, and an FC-layer leads to the output. ResBlock refers to residual blocks.

take two images to be processed as input [277]. Then, the images are initially processed independently by the same set of learnable filters. At a fusion point, the feature maps of both images are aggregated and processed jointly by the remaining network layers [109, 580]. We consider the fusion point f_n as a hyperparameter that we study. Performing fusion early in the network relates to the assumption that the two images are very similar. The most extreme case is feature fusion at the input, by stacking the two states in the channel dimension. Conversely, late fusion is required if the two images are dissimilar and need to be processed individually first.

For fusion itself, we also consider several different operations. Straight-forward fusion techniques include pixel or voxel-wise summation and addition. Concatenation along the feature map dimension is another option where the combination of the two paths' features can be learned. The concatenation of the two paths is defined as follows. For path 1, consider a feature map tensor $f_m^1 \in \mathbb{R}^{N_b \times n_h \times n_w \times n_d \times n_c^1}$ where N_b is the batch size, n_h, n_w, n_d, are the feature maps' height, width, and depth and n_c^1 is the number of feature maps. The tensor f_m^2 is designed similarly. Thus, the concatenated tensor $f_m^3 = f_m^1 \| f_m^2$ has a shape of $f_m^3 \in \mathbb{R}^{N_b \times n_h \times n_w \times n_d \times n_c^1 + n_c^2}$. In our case, this doubles the number of feature maps, which is why we keep the number of feature maps constant in the following spatial reduction block in the network. This keeps the overall feature map sizes within the network at a reasonable level, despite the concatenation.

Regarding general network design, after the initial convolution in the network, we employ blocks derived from different architecture concepts such as ResNet or DenseNet. For spatial reduction within the network, we employ convolutions with a stride of 2.

For most tasks, we consider a purely contracting architecture that outputs a vector for classification or regression. Here, the spatial dimensions are repeatedly reduced up to a GAP layer, followed by an FC-layer. For segmentation tasks, we also consider Siamese architectures with an encoder-decoder structure. This is detailed further in the next section.

Siamese architectures can also be challenging in terms of computational and memory requirements, in particular, for 3.5D data. One way to overcome this issue is to follow the approach of carefully designing efficient architectures, as we outlined for the case of extending 2D CNNs to 3D, and for the design of 4D CNNs. Thus, architecture hyperparameters $h_M^A \subset h_M$ are chosen by the machine learning engineer. The other approach we proposed is to utilize standard architectures and overcome the immense increase in model size by transfer learning. This resembles our multi-dimensional transfer learning strategy where we reuse most architecture hyperparameters $h_M^A \subset h_M$ from the original architecture.

This approach can also be adopted for Siamese CNN design. For example, we can make use of a standard ResNet or DenseNet model and transform it into a Siamese architecture. For this purpose, we clone the initial part of the CNN, before the fusion point. After the fusion point, we use a single path of the rest of the original CNN. The CNN part before the fusion point does not need to be adjusted, besides cloning for obtaining two parallel paths. However, after the fusion point, if concatenation is employed, the original model changes slightly as twice the number of features need to be processed. A majority of the network's weights can be kept identical after concatenation by immediately reducing the number of feature maps back to the size of the original pretrained reference architecture.

Still, the convolutional layer that downsamples the feature map dimension after the concatenation needs to be adjusted for effective transfer learning. This layer performs downsampling along the feature map dimension with a kernel size of 1 along all spatial dimensions. Consider two spatial 2D feature maps before concatenation from the two path, each with shape $f_m^1 \in \mathbb{R}^{N_b \times n_h \times n_w \times n_c^1}$ and $f_m^2 \in \mathbb{R}^{N_b \times n_h \times n_w \times n_c^1}$. Thus, the downsampling convolution's weight tensor has the shape $w_s \in \mathbb{R}^{1 \times 1 \times 2n_c^1 \times n_c^2}$ where n_c^1 is the downsampled feature map size. We assign the original, pretrained weight tensor of shape $w_o \in \mathbb{R}^{1 \times 1 \times c_1 \times n_c^2}$ both to the sliced tensor of shape $w_s^1 \in \mathbb{R}^{1 \times 1 \times 1 \ldots n_c^1 \times n_c^2}$ and $w_s^1 \in \mathbb{R}^{1 \times 1 \times n_c^1 \ldots 2n_c^1 \times n_c^2}$. We show that this initialization does have a significant impact on performance.

Attention-Guided Interactions. The described architecture concept for regression and classification tasks is also applicable to segmentation problems with encoder-decoder architectures. Here, an intuitive approach is to use the point between encoder and decoder as the fusion point f_n. For the fusion point, we also consider different methods, including subtraction, addition, and concatenation. Another modification is the use of long-range connections, which are typical for segmentation architectures. Here, every long-range connection also needs to be augmented by a fusion strategy, similar to the normal fusion point.

In contrast to regression or classification problems, predicting a segmentation map requires accurate spatial correspondences between the two input images as spatially consistent outputs need to be generated. We hypothesize that an additional focus on relevant, corresponding regions could improve the learning problem. Therefore,

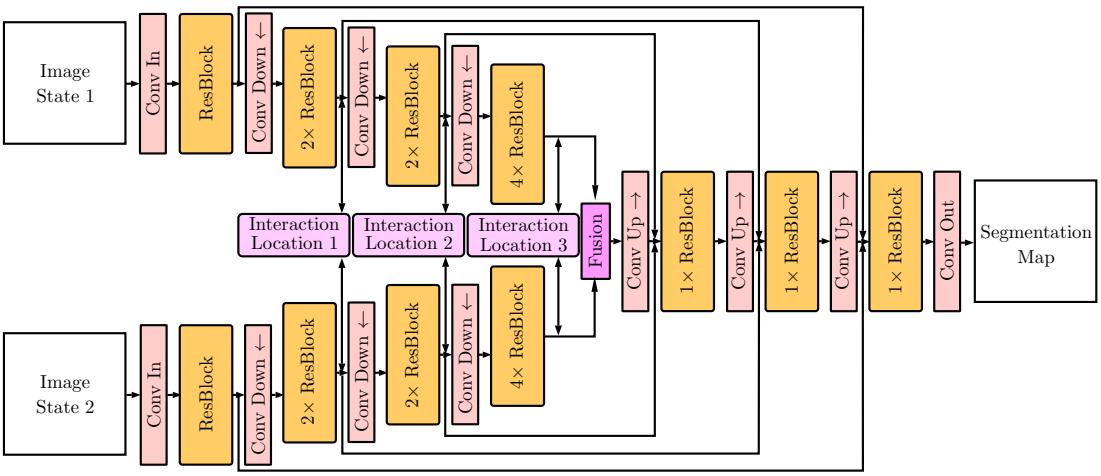


Fig. 4.13: The Siamese CNN architecture we propose for segmentation problems. The interaction modules are shown in Figure 4.14.

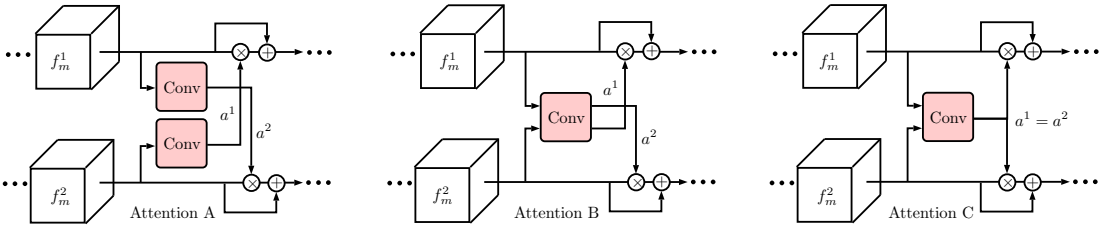


Fig. 4.14: The three types of attention-guided interactions we employ are shown. After the convolutions, a sigmoid activation function is applied.

we also incorporate targeted information exchange between the encoder paths, see Figure 4.13. For this purpose, we propose a trainable *attention-guided interaction* block, which controls information flow between the two paths. Inspired by squeeze-and-excitation [205, 418], and spatial attention [522], the attention module learns a map that suppresses spatial locations to guide the network's focus on salient regions. We consider blocks at three different locations in the network and propose three different variants for information exchange, see Figure 4.14.

For the first block, each path guides the other path's focus independently (attention method A). Consider features maps f_m^2 and f_m^1 of size $f_m \in \mathbb{R}^{n_h \times n_w \times n_d \times n_c}$ at layer l of each path. n_h, n_w, and n_d are the spatial feature maps sizes and n_c is the number of feature maps at layer l. We compute the attention map for the first path as

$$a^1 = \sigma(\text{conv}(f_m^1)) \tag{4.10}$$

where σ is a sigmoid activation function and $conv$ is a $1 \times 1 \times 1$ convolutional layer with learnable weights w_M^1. The map a^1 is of size $a^1 \in \mathbb{R}^{n_h \times n_w \times n_d \times n_c}$ and multiplied element-wise with f_m^1. Following the concept of residual attention [522], we finally add the modified feature map to the original feature map f_m^1. Thus, we obtain

$$\hat{f}_m^1 = a^1 \odot f_m^1 + f_m^1. \tag{4.11}$$

In this way, the information in the original feature maps is preserved while the attention maps provide additional focus to relevant regions. The attention map for path 2 and f_m^2 is computed symmetrically.

For the second block, we use both attention maps f_m^2 and f_m^1 for the computation of both attention maps (attention method B). Thus, we concatenate both feature tensors along the last dimension and compute

$$a^1 = \sigma(\text{conv}(f_m^1 \| f_m^2)) \tag{4.12}$$

with weights w_M^1. The attention map has the same dimensions as the ones computed for attention method A. Similar to method A, we use residual attention. We multiply the attention with the original feature map ${f_m}^1$ and add the original feature map to the result. The map a^2 is computed similarly with weights w_M^2.

Third, we consider a jointly learned attention map $a^1 = a^2$ (attention method C). We perform the same computation as for method B and share the weights $w_M^1 = w_M^2$. In this way, method C is more efficient in terms of the number of parameters, but the two paths receive attention towards the same regions.

Summary. Several application scenarios we consider throughout this work come with 2.5D and 3.5D image data. For processing this type of data, we propose Siamese CNN architectures, which we adopted from the natural image domain. These models process the two images individually up to a fusion point, where we consider both the position of the fusion point and the fusion method as important hyperparameters. We employ this architecture for regression, classification, and segmentation problems. For the latter, we also introduce attention-guided interaction blocks for improved information exchange between the two images' processing paths and an effective focus on relevant image regions.

4.3 Recurrent-Convolutional Models

CNN-GRU. Often, high-dimensional data is a combination of spatial and temporal dimensions. For the natural image domain, videos typically have a shape of $\mathrm{x} \in \mathbb{R}^{n_t \times n_h \times n_w \times n_c}$ where $n_c = 3$ is the RGB color channel. While convolutions can be used for any type of data dimension, recurrent architectures, that we introduced in Section 3.3.3, have also been shown to be particularly effective at processing temporal dimensions. The original design of RNNs is meant for time series that are sequences of feature vectors $\mathrm{x} \in \mathbb{R}^{n_t \times n_c}$ where n_c is the feature vector's length. Thus, they are not suitable for multi-dimensional data, where we have to deal with sequences of spatial images.

The conventional approach for this problem was to use feature extraction methods to obtain a feature sequence from individual images. This approach has been extended to the deep learning domain by using a CNN as the feature extractor [108]. Here, a CNN is trained on individual 2D images $\mathrm{x} \in \mathbb{R}^{n_h \times n_w \times n_c}$, for example, for a conventional image classification task. Then, the CNN is used for the target spatio-temporal sequence by performing forward passes through the CNN for the individual images in the sequence. Typically, the features after the CNN's GAP layer, before the classification layer, are extracted from the forward pass of each image. As a result, we obtain a sequence of

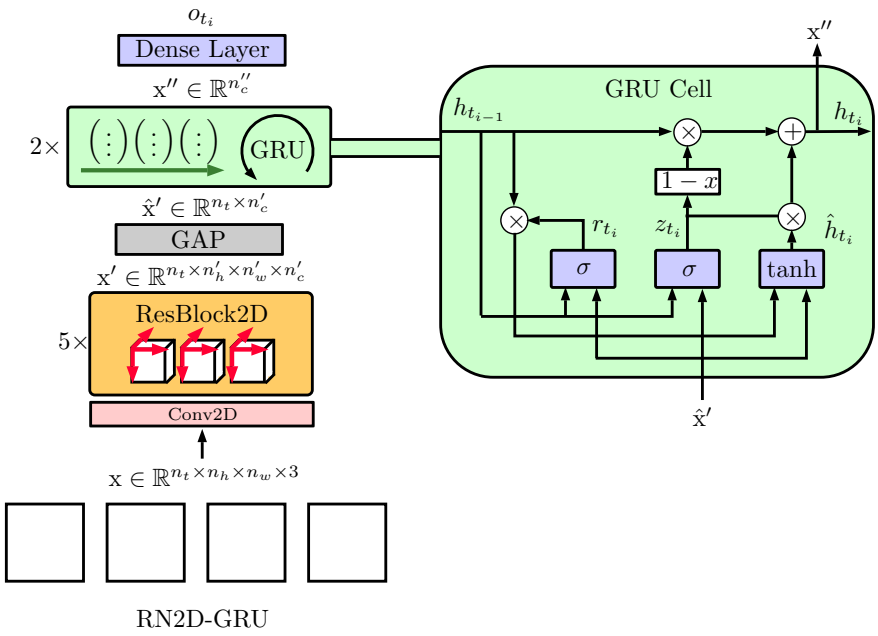


Fig. 4.15: The RN2D-GRU model we employ, which is inspired by typical CNN-RNN models for video processing in the natural image domain [380]. S GAP refers to spatial GAP. σ and $\tanh$ refer to dense neural network layers with sigmoid and tanh activation function, respectively.

feature vectors $\mathrm{x}' \in \mathbb{R}^{n_t \times n_c'}$, encoding the spatio-temporal sequence $\mathrm{x} \in \mathbb{R}^{n_t \times n_h \times n_w \times n_c}$. The sequence x' is now used to train a conventional RNN architecture for a target task such as action recognition or tracking.

An extension of this approach is to train a single deep learning model in an end-to-end fashion instead of pretraining the CNN for a different task [380]. The concept is shown in Figure 4.15. Here, the CNN and the RNN architecture are directly connected after the CNN's GAP layer. The full architecture takes the full spatio-temporal tensor $\mathrm{x} \in \mathbb{R}^{n_t \times n_h \times n_w \times n_c}$ as the input. Each of the n_t CNN paths processes one image $\mathrm{x} \in \mathbb{R}^{n_h \times n_w \times n_c}$. Typically, all CNN paths share parameters as we expect the individual images in the sequence to have similar spatial features and intend to focus inter-frame relationship learning on the temporal RNN part of the model. In practice, this can be implemented by using a single CNN that processes all images in the sequence in parallel through the batch dimension. Thus, with a batch dimension of size N_b, the single-path CNN processes multiple sequences $\mathrm{x} \in \mathbb{R}^{N_b \times n_t \times n_h \times n_w \times n_c}$ as a reshaped tensor $\hat{\mathrm{x}} \in \mathbb{R}^{N_b n_t \times n_h \times n_w \times n_c}$. Notice that this approach is similar to our proposed MP-CNN4D architecture in the previous section, except that we replaced the GAP layer and RNN with a full spatio-temporal convolutional processing block.

Both the CNN and the RNN part of this architecture concept can follow any design principle. For example, the CNN can be based on a ResNet, Inception-V3, or DenseNet, and the RNN can be a vanilla RNN, an LSTM, or a GRU. Throughout this thesis, we generally employ ResNet- or DenseNet-based CNNs for the spatial processing part and a GRU for the temporal processing part. For other multi-dimensional problems, RN2D-GRU can be converted to an RN1D-GRU or RN3D-GRU for sequences of 1D

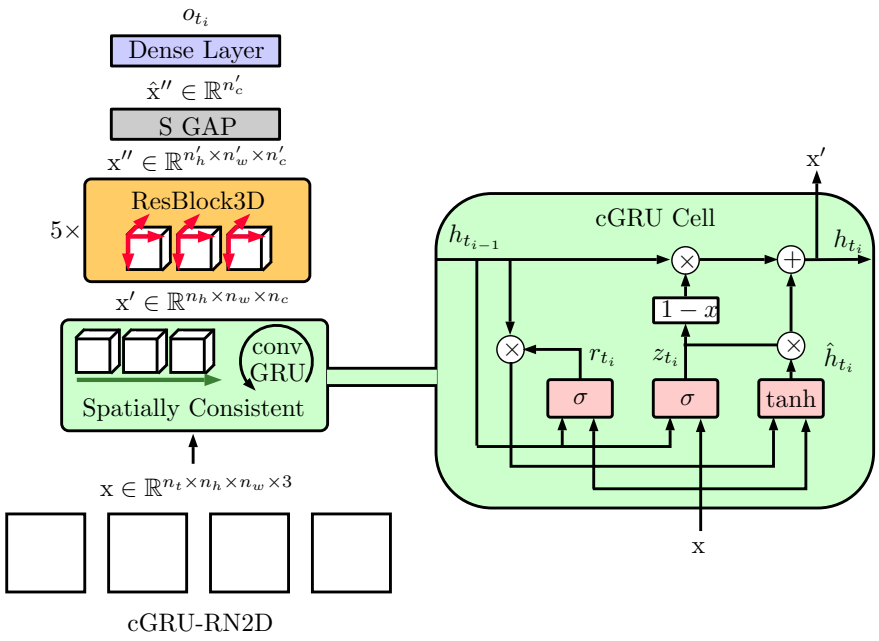


Fig. 4.16: The cGRU-RN2D model we propose. S GAP refers to spatial GAP. σ and $\tanh$ refer to convolutional neural network layers with sigmoid and tanh activation function, respectively. Note, that the use of convolutional layers instead of dense layers is a key difference to RN2D-GRU shown in Figure 4.15.

images or 3D volumes, respectively, when using a ResNet basis.

cGRU-CNN. As a next step, we extend the idea of using both CNNs and RNNs in an architecture. Most approaches for processing videos rely on the concept introduced previously, where a CNN is followed by an RNN. This is largely motivated by the idea to obtain abstract feature representations first and then to find temporal relationships between these abstract representations. While this is a reasonable approach for tasks such as action recognition, obtaining abstract representations of the spatial images first might not be optimal for other tasks. Consider a problem where temporal information is available, but not strictly necessary, for example, in the context of force estimation. Here, a temporal history of previous images largely serves the purpose of obtaining more consistent estimates from a sequence of images instead of a single estimate. Thus, capturing small, local variations in the full spatio-temporal sequence might be preferable over searching for context between abstract feature vectors.

This motivates the idea to reverse the order of processing steps for combined CNN and RNN architectures. We propose to let an RNN architecture perform temporal processing first, producing a single aggregated spatial representation for further processing by the CNN. Taking the example from the previous section where a video is being processed, the RNN architecture receives a sequence $\mathrm{x} \in \mathbb{R}^{n_t \times n_h \times n_w \times n_c}$ and produces an aggregated spatial representation $\mathrm{x}' \in \mathbb{R}^{n_h' \times n_w' \times n_c'}$. Notice that this requires the RNN architecture to be able to perform spatial processing in order to keep the spatial structure intact for subsequent CNN processing. This can be enabled by utilizing convolutional RNN modules which have been proposed for weather forecasting [556]. In that work, the author exchanged the conventional dense neural network layers, that act as the gates, by

convolutional layers for an LSTM model. We apply this approach to the more efficient GRU architecture. As a result, the computations performed by this cGRU are described by

$$\begin{aligned} z_{t_i} &= \sigma(K_z * h_{t_{i-1}} + L_z * RBN(\mathrm{x}_{t_i})) \\ r_{t_i} &= \sigma(K_r * h_{t_{i-1}} + L_r * RBN(\mathrm{x}_{t_i})) \\ c_{t_i} &= \tanh(K_c * (r_{t_i} h_{t_{i-1}}) + L_c * RBN(\mathrm{x}_{t_i})) \\ h_{t_i} &= z_{t_i} c_{t_i} + (1 - z_{t_i}) h_{t_{i-1}} \end{aligned} \tag{4.13}$$

where h_{t_i} is the hidden state, x_{t_i} is the input, K and L are filters, $*$ denotes a convolution, σ denotes the sigmoid activation function, and $RBN(.)$ denotes recurrent batch normalization [94]. In most applications, we also employ recurrent dropout [378] at the cell input with probability p_{di} and at the cell output with probability p_{do}. Combining this cGRU with a ResNet-based CNN results in our final architecture cGRU-RN2D, which is shown in Figure 4.16 for the example of 3D spatio-temporal data. Similar to RN2D-GRU, this architecture can be applied in a multi-dimensional context by converting RN2D into a RN1D or RN3D for sequences of 1D images or 3D volumes, respectively.

The idea of using convolutions in the processing gates of a recurrent unit can also be directly applied to our CNN-GRU model. To form a CNN-cGRU model, we swap the order of the GAP layer and the GRU layers. Thus, the cGRU layers receive the input $\mathrm{x}' \in \mathbb{R}^{n_t \times n'_h \times n'_w \times n'_c}$ and produce an output $\mathrm{x}'' \in \mathbb{R}^{n'_h \times n'_w \times n''_c}$. Then, the GAP layer is applied to obtain a feature vector $\hat{\mathrm{x}}'' \in \mathbb{R}^{n''_c}$ that is processed by the output layer for producing a prediction.

cGRU-CNN-U. In Section 4.2, we discussed 2.5D and 3.5D architectures in the context of segmentation problems where a segmentation map is derived from two input images. For problems such as MS lesion activity segmentation, an entire temporal sequence of input volumes can be available, instead of just two volumes representing two states. Taking care of sequence processing can also be performed with recurrent models, integrated into a U-Net-like CNN architecture.

Following the idea of fusion between encoder and decoder leads to the architecture depicted in Figure 4.17. Here, the model receives an input of size $\mathrm{x} \in \mathbb{R}^{n_t \times n_h \times n_w \times n_d \times n_c}$. Then, n_t encoder paths process the n_t volumes individually, similar to the RN2D-GRU model. Again, parameter sharing can be enabled by processing the individual image volumes as part of the batch dimension.

After the encoder, all n_t volumes are aggregated using a recurrent architecture. As the volumes are spatial in nature, we use cGRUs once again in order to preserve the spatial structure for the decoder. Thus, one or several cGRU layers receive the input $\mathrm{x}' \in \mathbb{R}^{n_t \times n'_h \times n'_w \times n'_d \times n'_c}$ and output a single spatial representation $\mathrm{x}'' \in \mathbb{R}^{n'_h \times n'_w \times n'_d \times n'_c}$. Then, x'' is processed by a normal decoder that upsamples the representation into the final segmentation prediction $\hat{\mathrm{y}} \in \mathbb{R}^{n_h \times n_w \times n_d}$.

Another aspect that is unique for segmentation architectures that needs to be considered is the presence of long-range connections in the architecture. Thus, a multi-time point encoder feature tensor needs to be connected to the single spatial representation in the decoder. Here, we opt for the same strategy as for the connection between encoder and decoder by using cGRUs for temporal fusion in each long-range connection before the decoder.

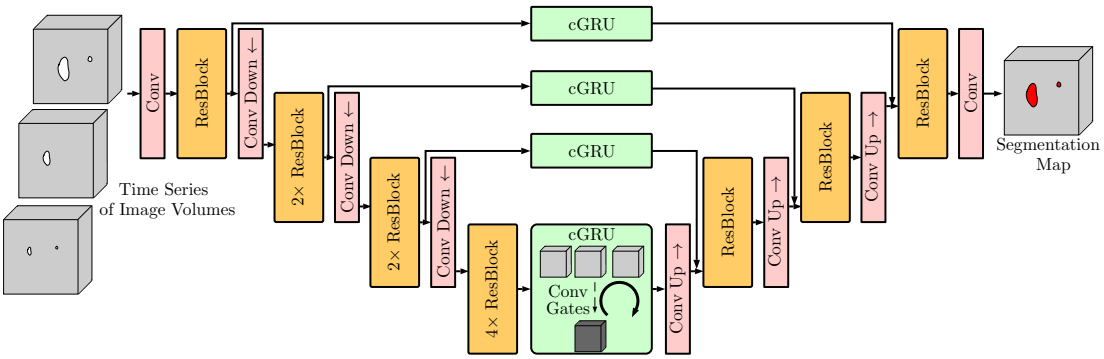


Fig. 4.17: The cGRU-CNN-U model we propose. The model takes a sequence of volumes as input and predicts a single volumetric segmentation map. All convolutional layers used 3D convolutions in this case.

Here, we described the cGRU-CNN-U for the case of a sequence of volumes. Thus, all convolutional layers employ 3D convolutions for spatial processing. In our applications we employ ResNet blocks within the architecture.

Summary. Multi-dimensional medical image data usually comes with several spatial dimensions and a temporal dimension. While convolutions can be universally used for any type of dimension, recurrent concepts such as LSTMs have been shown to be effective for temporal processing. Thus, multi-dimensional spatio-temporal data can be processed by joint CNN and recurrent architectures. For this type of architecture, we consider several different variants. First, we design a model that follows design principles from the natural image domain with a CNN, followed by recurrent GRU. Second, we proposed a cGRU-CNN concept where spatially consistent, temporal processing and aggregation is performed first, followed by spatial processing. Third, we apply this concept to segmentation problems and propose a cGRU-CNN-U concept where a cGRU aggregates a temporal sequence produced by an encoder, followed by a decoder that processes and predicts a single, spatial representation.

4.4 Summary

Overall, we propose a large number of deep learning methods that can be employed for different tasks where multi-dimensional aspects are relevant. We start with CNNs, the most common type of deep learning model for image processing. The majority of CNNs is designed for standard 2D image data. Thus, we first address the problem of moving from 2D models to 1D. In terms of computational and memory requirements, this step is not complicated, as 1D architectures are very lightweight. This opens up the option to implement an optimization procedure where we automatically search for the optimal CNN architecture. We study this idea in the context of image segmentation, where we search for a U-Net's building blocks using neural architecture search.

The next challenging step is to move from 2D standard architectures to 3D models for processing 3D image volumes. Here, the extension is not trivial as the higher dimensional convolutions require more computational power, and the increase in trainable parameters for capturing higher-dimensional context also increases the risk of overfitting. Our

first approach to this problem is careful architecture engineering, where we design 3D CNN extensions for four different architecture concepts we adopted from the 2D image domain. Our second approach to this problem assumes that the issue of computational requirements is solvable by better hardware. Thus, we address the remaining risk of overfitting with a multi-dimensional transfer learning strategy. Here, we directly use pretrained models from the 2D domain and isotropically expand all their 2D operations to 3D. This process is enabled by a weight rescaling strategy that ensures consistent feature ranges for the higher-dimensional computations. As a next step, we also extended 3D CNNs to 4D. This encompasses the same problems as for 2D to 3D extensions in an even more severe way. Thus, we take the route of architecture engineering and propose three different 4D CNNs variants that use full 4D convolutions and efficient alternatives such as factorized convolutions.

Not all image data can be assigned to, for example, 2D or 3D image data in a meaningful way if one of the data dimensions is very small. Often, input data only consists of two states captured by two images. For this type of 2.5D and 3.5D image data, we propose Siamese two-path architectures that we adapt from the natural image domain. For this type of architecture, we study concepts such as early and late feature fusion, different fusion strategies, and modifications that enable transfer learning. For 3.5D segmentation problems, we also propose attention-guided interaction modules between processing paths.

Last, we also consider recurrent architectures that are particularly suited for processing temporal data. A lot of multi-dimensional learning problems have a spatio-temporal data structure and thus, one temporal dimension that needs to be processed. Therefore, joint CNN and RNN architectures can be designed that perform spatial processing with convolutions and temporal processing with recurrent modules. In the natural image domain, CNN-RNN concepts are popular where a CNN produces abstract, spatial representations that are processed by an RNN. We extend this concept by performing temporal processing first with a convolutional RNN module that produces a single spatial representation for CNN processing. This follows the idea of aggregating a sequence, where all elements are very similar, into a single representation before extracting abstract features. We also extend this concept to segmentation problems, where we propose an encoder-decoder architecture with recurrent aggregation of volumes in a temporal sequence.

An overview of all deep learning architectures and some of their implementations in this thesis is given in Table 4.2. Next, we introduce the different application scenarios, where we apply the architecture concepts described in this chapter.

Tab. 4.2: An overview of all architectures we employ, example implementations, and their respective properties and application task. Tasks include regression or classification (R./C.) and segmentation (Seg.). The keyword custom indicates that the model is a custom implementation.

	Data Dim.	Implementation	Description	Task
Convolutional Neural Networks	1D	RN1D	Custom ResNet	R./C.
		RN1D-U	Custom ResNet-based U-Net	Seg.
		ENAS1D-U	Custom U-Net found by ENAS	Seg.
	2D	RN2D	Custom ResNet	R./C.
		IN2D	Custom Inception	R./C.
		RN2D-50	Adapted ResNet50 [193]	R./C.
		DN2D-121	Adapted DenseNet121 [208]	R./C.
		SENet2D-154	Adapted SENet154 [205]	R./C.
		RN2D-U	Custom ResNet-based U-Net	Seg.
		ENAS2D-U	Custom U-Net found by ENAS	Seg.
	3D	RN3D-A	Custom ResNet w/o bottlenecks	R./C.
		RN3D-B	Custom ResNet w/ bottlenecks	R./C.
		IN3D	Custom Inception	R./C.
		RX3D	Custom ResNext	R./C.
		DN3D-121	Adapted DenseNet121 [208]	R./C.
		SE-RN3D-101	Adapted SE-ResNet101 [205]	R./C.
		facRN3D	Custom ResNet w/ factorized convolutions	R./C.
		RN3D-U	Custom ResNet-based U-Net	Seg.
	4D	RN4D	Custom ResNet	R./C.
		DN4D	Custom DenseNet	R./C.
		facRN4D	Custom ResNet w/ factorized convolutions	R./C.
		MP-DN4D	Custom multi-path DenseNet	R./C.
Siamese Models	2.5D	TP-RN2D-50	Two-path CNN based on ResNet50 [193]	R./C.
		TP-DN2D-121	Two-path CNN based on DenseNet121 [208]	R./C.
	3.5D	TP-DN3D	Custom two-path DenseNet	R./C.
		MP-DN3D	Custom multi-path DenseNet	R./C.
		TP-RN3D-U	Custom two-path ResNet	Seg.
		MP-RN3D-U	Custom multi-path ResNet	Seg.
Recurrent-Convolutional Models	2D	RN1D-GRU	Custom ResNet-based CNN-GRU	R./C.
		RN1D-cGRU	Custom ResNet-based CNN-cGRU	R./C.
		GRU-RN1D	Custom ResNet-based GRU-CNN	R./C.
		cGRU-RN1D	Custom ResNet-based cGRU-CNN	R./C.
	3D	RN2D-GRU	Custom ResNet-based CNN-GRU	R./C.
		RN2D-cGRU	Custom ResNet-based CNN-cGRU	R./C.
		cGRU-RN2D	Custom ResNet-based cGRU-CNN	R./C.
	4D	RN3D-GRU	Custom ResNet-based CNN-GRU	R./C.
		RN3D-cGRU	Custom ResNet-based CNN-cGRU	R./C.
		cGRU-RN3D	Custom ResNet-based cGRU-CNN	R./C.

5 Application Scenarios and Previous Work

In this chapter, we introduce the application problems that we address throughout this thesis. We start by reviewing previous work and its development over time, where we focus on the use of data representations and machine learning methods. Then, we highlight aspects and open questions concerning data representations and deep learning that have not been sufficiently addressed in the literature so far.

First, we discuss the problem of vision-based force estimation. We address both image-based force estimation with higher-dimensional data representations as well as needle-based force estimation approaches with low-dimensional data. Second, we review work on OCT-based tissue segmentation and classification regarding the two major clinical applications of OCT, ophthalmology and intravascular imaging of the heart. Third, we address left-ventricle quantification, another deep learning problem that is focused on the heart as an anatomical region. Fourth, we examine the problems of pose and motion estimation with deep learning methods and higher-dimensional data. Fifth, we introduce the longitudinal disease tracking problem of multiple sclerosis lesion activity segmentation. Finally, we review work on multi-dimensional deep learning that is not associated with any of our application problems. We conclude the chapter by highlighting general trends and open questions across all applications and imaging modalities.

5.1 Vision-Based Force Estimation

5.1.1 2D and 3D Image-Based Force Estimation

Vision-based force estimation is primarily motivated by the idea of estimating forces without the need for a mechatronic force sensor, for example, for robot-assisted interventions. Robot-assisted minimally invasive surgery (MIS) has become increasingly popular as it addresses various shortcomings of conventional MIS [540]. Robotic systems allow for motion scaling, tremor compensation, and more degrees of freedom for tool movement, which improves precision and reduces physical trauma [263]. However, these systems often lack force feedback [105], which would be helpful in controlling the instrument-tissue interaction during surgery. Typically, haptic feedback is generated on the patient side with haptic sensors, such as force sensors [101]. The information is fed back to a haptic interface that delivers the information to the human operator, for example, as vibrotactile or kinesthetic feedback [333]. One of the key challenges of generating reliable haptic feedback is accurate sensing of the forces at the patient [361]. Lack of haptic feedback may lead to complications, increased completion time, or

severe injuries [369]. Although various approaches to realize force feedback have been proposed, the problem is still considered an open research challenge [37].

One approach is to directly incorporate force-sensing devices into the robotic setup [388]. The devices can be placed inside or outside of the patient. If the device is placed outside the patient, for example, in between tool and robot, only indirect measurement is possible. In addition to the forces at the tool tip, forces acting on the tool, for example, due to friction, are measured, which cannot be separated [133]. When placing the device closer to the tool-tissue interaction point, for example, inside the tool tip, problems such as sterilization and biocompatibility arise [457].

Due to these shortcomings, vision-based force estimation procedures have been proposed. Vision-based force estimation is the task of estimating forces that are acting on tissue, only based on images that are typically provided by RGB-D cameras. This problem was initially not considered as a spatio-temporal data processing problem as early methods relied on deformable template matching methods [179] for finite element modeling techniques [233]. The methods were largely motivated by applications to microassembly and biomanipulation of individual biological cells [310]. Similarly, following methods relied on mechanical deformation models [247, 249, 360]. Kim et al. considered the problem of haptic feedback in MIS [247]. Based on a-priori material knowledge and geometric estimates from camera images, the interaction forces between a tool and material were estimated. Noohi et al. considered the problem of estimating deformation from a monocular endoscopic camera without the use of a known, undeformed template [360]. Another research direction in vision-based force estimation is to use machine learning instead of explicit deformation modeling. Gremninger et al. proposed to learn forces based on handcrafted features [178]. Karimirad et al. applied this approach to biological cell microinjection using geometric image features and a fully-connected neural network [235]. Mozaffari et al. applied this idea to a laparoscopic surgery scenario and estimation of tool-tissue interactions force [351]. Here, geometric features were also used in conjunction with an adaptive neuro-fuzzy inference system.

More recently, the temporal dimension has been integrated into force estimation models by using handcrafted features derived from stereoscopic camera images in recurrent models [29]. This has been extended by the use of LSTM models [27, 28]. Using spatio-temporal data follows the idea of tracking tissue deformation over time, which provides a more reliable estimate than a single-shot measurement during surgery where the tissue is in continuous motion. Naeini et al. also employed a temporal approach for estimating grip forces by using dynamic vision sensors and time-delay neural networks [353]. Spatio-temporal approaches were extended by Marban et al. [321], where a 2D CNN extracts spatial features that are then fed into an LSTM. Gao et al. [147] use both a 2D CNN to learn spatial features from RGB images as well as PointNet [389] to process point clouds derived from depth images. Features from both networks are fed into an additional CNN for temporal processing. Thus, the approach takes 3D information into account by using depth, however, an explicit 4D problem is avoided. Kim et al. took a similar approach using 3D CNNs for spatio-temporal processing of RGB videos [246]. The authors employed factorized convolutions [472] for more efficient processing. Another approach utilized 2D CNNs followed by LSTMs, augmented by a sequential spatial and channel-wise attention mechanism [445]. The method applies attention modules, similar to squeeze-and-excitation attention [205],

while considering both the current and the previous frame for calculating the attention map. The method was applied to several phantoms that are not related to medical applications.

While most methods rely on RGB and depth cameras, Otte et al. proposed to use OCT as an imaging modality [367]. Here, classical feature extraction is performed from an undeformed reference image and an image showing deformation by a tool. These features are fed into a conventional machine learning model. A recent approach took a classification approach to force estimation with simulated 2D OCT images and fully-connected neural networks [335]. Spatio-temporal deep learning approaches have not been presented for OCT-based force estimation so far. In contrast to depth imaging, OCT can provide volumetric images with a resolution of a few micrometers, which allows for capturing the inner structure of tissue. The volumetric OCT image can also reflect tissue compression. Here, the question arises whether deep learning with 3D volumetric images provided by OCT is advantageous over learning from 2D depth maps, which have been used in the natural image domain. Besides 2D and 3D spatial data representations, the temporal dimension is still an open problem. In particular, the benefit of past OCT images for a current estimate is unclear. Furthermore, when considering a temporal sequence of 3D volumes, the challenging problem of 4D spatio-temporal deep learning arises. Also, using temporal information opens up the question of short-term force *prediction*. Assuming some non-random deformation and force patterns over time, forces should be predictable, given a history of data. Predicting forces could enable safety mechanisms for robot-assisted interventions [184, 188] as large force value increases could be detected earlier.

Summarized, vision-based force estimation was originally addressed with mechanical models. Early machine learning approaches made use of handcrafted features extract from images or videos. More recently, deep learning methods have been applied to different 2D spatial and 3D spatio-temporal image data representation using natural RGBD images. Also, OCT has been proposed as an alternative imaging modality that can also image subsurface tissue compression. For OCT, there has been no work on deep learning-based force estimation. Also, a concise comparison of multi-dimensional data representations is still missing, and deep learning with 4D data has not been addressed. Furthermore, force prediction has not been studied so far. In this thesis, based on our previous work [154, 155], we address these shortcomings of vision-based force estimation with OCT and deep learning.

5.1.2 Needle-Based Force Estimation

The problem of needle-based force estimation is also related to vision-based force estimation. As a lot of needle-based force estimation methods rely on simple depth measurements or 1D depth images, needle-based force estimation generally deals with low-dimensional data. Nevertheless, the problem comes with interesting problems regarding different lower-dimensional data representations.

For minimally invasive procedures such as biopsy, neurosurgery or brachytherapy, needle insertion is often utilized to minimize tissue damage [6]. To facilitate accurate needle placement, needle steering, image guidance, and force estimation [484] can be used. Accurate measurement of the forces affecting the needle tip is of particular

interest, for example, to keep track of the needle-tissue interaction and to detect potential tissue ruptures, or to generate haptic and visual feedback [362]. Therefore, various force-sensing solutions for needles have been proposed. A simple approach is to place a force sensor externally at the needle shaft, which would allow for the use of conventional force-torque sensors. However, during insertion, large frictional forces act on the needle shaft, which mask the actual tip forces. Therefore, forces acting on the needle shaft either need to be decoupled from the force sensor, or the force sensor needs to be placed at the needle tip [237]. This complicates building force sensors as they are usually constrained to a few millimeters in width [406], which is particularly difficult for mechatronic force sensors [190, 237]. For these reasons, fiber optical force sensors have been developed, which are often smaller, biocompatible, and MRI-compatible [38]. In particular, sensor concepts using Fabry-Pérot interferometry [38, 466] or Fiber Bragg Gratings [265, 395, 410] have been proposed. Su et al. built an opto-mechanical sensor based on a Fabry-Pérot interferometer where changes in force are reflected in an interference signal that is captured by a photodetector [466]. The sensor was calibrated, but no further practical experiments were performed. Beekmans et al. built on this concept and designed and calibrated several different needle-based force-sensing designs [38]. Fiber Bragg Gratings are another approach for optical force-sensing [395]. Here, Bragg gratings are introduced to a fiber, which cause partial reflections of light passing through the fiber. Changes to the fiber, such as an external force, cause a wavelength change that can be detected and correlated to the external force. Kumar et al. built a needle-based force sensor using this concept and validated it in several insertion experiments [265].

Summarized, all these methods have shown promising calibration results, however, manufacturing and signal processing can be difficult when different temperature ranges and lateral forces need to be taken into account. Also, the proposed fiber-optical methods typically rely on very low-dimensional data with a few features obtained from an optical signal and point estimates without a spatial and temporal component. Thus, using a higher-dimensional signal that is rich in information has not been explored so far. Furthermore, calibration methods are generally simple and potentially limited in their capability of capturing nonlinear behavior. Based on our previous work [161, 162], we address these aspects by building an OCT-based needle tip force estimation mechanism. OCT allows for moving from scalar signals and features to higher-dimensional 1D and 2D spatio-temporal data that we process with different deep learning models.

A summary of relevant literature on vision- and deep learning-based force estimation in the context of data dimensionality is provided in Table 5.1.

5.2 OCT-Based Tissue Classification

There are two major clinical application fields for OCT-based tissue classification. First, OCT is frequently used in ophthalmology and focused on the classification and segmentation of structures within the human eye. Second, IVOCT is used for intravascular imaging. Here, classifying diseased tissue is important for guiding treatment decisions.

Tab. 5.1: Overview of related work on vision-based force estimation with deep learning methods. The data types can be feature vectors (FV), RGB(D), or OCT images, including a temporal (T) component.

Reference	Application	DL Method	Data Type
Gremniger et al. (2003) [178]	Material deformation and micrograpping	FC-NN	FV
Karimirad et al. (2014) [235]	Biological cell microinjection	FC-NN	FV
Mozaffari et al. (2014) [351]	Laparoscopic surgery, phantom data	ANFIS	FV
Aviles et al. (2014) [29]	Robot-Assisted MIS, phantom data	RNN	FV-T
Aviles et al. (2015) [28]	Robotic-Assisted MIS, phantom data	LSTM	FV-T
Aviles et al. (2017) [27]	Robotic-Assisted MIS, in-vivo animal data	LSTM	FV-T
Naeini et al. (2019) [353]	Generic tactile sensing	TDNN	FV-T
Marban et al. (2019) [321]	Robotic-Assisted MIS, phantom data	CNN-LSTM	RGB-T & FV-T
Gao et al. (2018) [147]	Robotic-Assisted MIS, phantom data, and tissue data	CNN	RGBD-T
Kim et al. (2018) [246]	Generic tactile sensing, phantom data	CNN	RGB-T
Shin et al. (2019) [445]	Generic tactile sensing, phantom data	CNN-LSTM	RGB-T
Otte et al. (2016) [367]	Robotic-Assisted MIS, phantom data	FC-NN	FV (OCT)
Mendizabal et al. (2019) [335]	Robotic-Assisted MIS, phantom data	FC-NN	FV (OCT)

5.2.1 Ophthalmology

In ophthalmology, typical tissue classification tasks include detection of AMD, DR, or glaucoma. Most eye diseases affect the retina in some way, which is why retina layer segmentation is also a common problem that is solved as a preprocessing step, followed by a more detailed examination [393].

Disease Classification. The first methods for OCT-based disease classification in ophthalmology relied on handcrafted feature extraction and conventional machine learning methods. Farsiu et al. obtained features by first segmenting the retina semi-automatically [134]. Then, features such as layer thickness and volume were calculated and used in a generalized linear model for disease classification. Sugmk et al. took a similar approach by using layer segmentation first, followed by the detection of drusen material [468]. Then, a binary classifier was applied for differentiating diabetic macular edema and AMD. Deng et al. relied on Gabor filter banks that undergo a nonlinear transformation as their features [102]. Then, several histograms are computed and concatenated from the filtered regions using different scales and orientations. Finally, the features are fed into random forests, SVMs, and FC-NNs for classifying the OCT images into AMD and normal cases. Another approach used unsupervised clustering for obtaining few descriptive image patches in a feature extraction stage [511]. These patches are used to build patch occurrence histograms for all training images that are then used for supervised classification with a random forest. Wang et al. obtained features based on linear configuration patterns, extracted from multi-scale OCT images, building feature pyramids [531]. Afterward, correlation-based feature selection was applied, followed by different classification models. These models included sequential linear optimization, naive Bayes, SVMs, FC-NNs, and random forests. Other work was more focused on diabetic macular edema in the context of DR [546]. For example, Alsaih et al. performed feature extraction with histogram of oriented gradients and local binary pattern, followed by principal component analysis for dimensionality reduction [16]. The authors used linear and kernel SVMs, as well as random forests for classification. Lemaitre et al. took a similar approach, extracting features both from 3D OCT volumes and 2D patches [286].

More recently, deep learning methods have been applied to OCT-based retinal disease classification. Ravenscroft et al. bridged the gap between handcrafting features and deep learning-based feature learning by using a CNN for feature extraction [398]. Then, learned feature maps are transformed into feature vectors by using histograms. A typical classification stage using SVMs, FC-NNs, or random forests follows. Rasti et al. relied on an end-to-end approach when using CNNs for AMD classification [397]. To improve performance, the authors used CNNs with multiple input scale resolutions in an ensemble. A more recent approach by Rong et al. applied extensive data augmentation by generating multiple surrogate images using denoising and a masking process [408]. The performance was improved by averaging over multiple surrogate images. Serener et al. distinguished dry and wet AMD using different standard models from the natural image domain, such as ResNet and AlexNet [437].

Very recently, Wu et al. proposed a modified CNN for classification of several retinal diseases using an attention mechanism that should focus on critical regions within OCT slices [545]. Also, a preprocessing mechanism for noise reduction was employed before

CNN processing. Wang et al. proposed a CNN architecture for AMD classification where a DenseNet-like structure is employed [521]. Additionally, features from multiple scales within the network are pooled to a global feature representation for the final classification. Das et al. followed a similar approach and also found that using features from multiple scales improves AMD classification performance [98]. Qiu et al. applied the idea of self-supervised iterative refinement training to AMD classification [390]. Here, during the CNN's training stage, a part of the training data is relabeled by the model, followed by training on the newly obtained labels. Saha et al. took a different approach to AMD classification by learning typical clinical biomarkers with a CNN instead of the typical disease labels [421]. An et al. explored transfer learning in the context of classifying wet and dry AMD, where they found that a model pretrained on typical AMD detection is well suited for the more fine-grained task [17]. A new approach by Yoo et al. proposed to combine OCT with the modality color fundus imaging for improved AMD classification [571]. The authors used a CNN for each modality to extract a feature vector that is then fed into a random forest classifier.

Overall, the number of deep learning approaches for retinal disease classification has been growing, particularly in very recent years. Conventional approaches considered very low-dimensional context as they relied on a few features, extracted from A-Scans. With the emergence of deep learning, almost all recent approaches perform 2D B-Scan-based processing. However, there are no studies considering a multi-dimensional perspective or comparing different data representations. Also, there are many different handcrafted architectures, and often, it is unclear which architecture concept is preferable.

Retinal Layer Segmentation. Similar to early approaches for disease classification, earlier methods for retinal layer segmentation relied on conventional computer vision methods, often paired with classic machine learning methods. Koozekanani et al. addressed retinal boundary segmentation of 2D OCT images using edge detection and a Markov model [256]. Another approach relied on A-Scan-based processing, where the individual intensity profiles were thresholded to obtain a segmentation of four retinal layers [217]. Similarly, Shahidi et al. made use of intensity peaks within A-Scans to segment several retinal layers [440]. Lu et al. made use of additional preprocessing and noise-filtering before applying edge detection for layer segmentation [316]. Cabrera et al. made use of a deformable model instead, which can also be used for the segmentation of fluid-filled regions and lesion around the retina [137]. The idea of higher-dimensional data processing was also introduced for classical methods by Garvin et al. [151]. Here, a graph was constructed based on detected edges in a 3D OCT volume and an optimization process to enforce surface smoothness. This was also applied to particular difficult cases with serious pigment epithelial detachments [443]. A similar idea was pursued by Mishra et al., where the rough location of retinal layers is determined first, followed by a refining stage [342]. Yazdanpanah et al. propose an energy-minimizing active contour approach that should deal particularly well with OCT images corrupted by high levels of noise [567]. The use of classical machine learning methods for retinal layer segmentation was proposed by Lang et al. [268]. Here, a random forest classifier was trained to predict each pixel's retinal layer class. This was followed by a boundary refinement algorithm to output the final segmentation.

The emergence of deep learning methods in the medical image domain also led to applications for OCT-based retinal layer segmentation. One of the first approaches to

employed deep learning was presented by Fang et al. [131]. In this work, CNNs were fused with a graph search method. First, retinal boundaries were predicted by a 2D patch-based CNN predicting the layer class of the patches' center pixel. Second, a graph search constructed the final segmentation using the CNN's predicted probabilities. Ben-Cohen et al. also employed CNNs for retinal layer segmentation [40]. A key difference to Fang et al. is the prediction of an entire segmentation map instead of patch-wise processing. In particular, Ben-Cohen et al. show that predicting segmentation maps outperforms the patch-based approach. Roy et al. proposed a CNN with an encoder-decoder structure for segmented 2D OCT B-Scans into nine retinal layers [416]. The authors demonstrated significantly improved performance both over conventional segmentation methods and early deep learning-based methods. Schlegl et al. used a similar CNN architecture for segmenting macular fluid regions within 2D OCT images [427]. Shah et al. also employed CNNs for retinal layer segmentation, however, they proposed a different model output scheme [439]. Instead of predicting a segmentation map, the authors' model output is a vector of layer positions that produced more plausible outputs than a standard encoder-decoder model. A study by De Fauw et al. demonstrated the clinical applicability of deep learning-based OCT image analysis on a very large dataset [100]. First, a CNN is used for layer segmentation. Here, a 2D approach is employed where several neighboring 2D OCT slices are fed into the CNN, and the central 2D slice is segmented at the model output. Afterward, a classification model also performs a disease diagnosis and outputs a treatment recommendation. Another approach incorporated uncertainty estimates into a CNN model by making use of the concept of Bayesian Deep Learning [435]. In terms of implementation, the authors rely on Monte Carlo dropout [143] for uncertainty quantification.

Very recent methods for retinal layer segmentation have introduced more incremental improvements to the overall task. Orlando et al. focused on segmenting the photoreceptor layer in particular [366]. The authors also relied on Monte Carlo dropout for uncertainty estimation. Masood et al. compared several conventional segmentation approaches to deep learning for choroid layer segmentation [325]. While the deep learning model did not come with any methodological innovation, substantial performance improvement over conventional methods was demonstrated. Matovinovic et al. differentiate themselves from other work by making use of transfer learning for 2D OCT image segmentation [326]. The authors made use of a CNN pretrained on the ImageNet dataset as the model's encoder. The approach outperformed the use of an encoder with standard random weight initialization. Ngo et al. took a very different approach where they went back to classic feature extraction, followed by an FC-NN that regresses the retinal boundaries [357]. The authors argue that the approach is more data-efficient and robust to noise than conventional deep learning approaches. The authors show that they outperform earlier encoder-decoder models for retinal layer segmentation. Another recent approach performed joint prediction of segmentation maps and boundary regression for improved performance over several state-of-the-art methods [195].

All in all, for retinal layer segmentation, high-dimensional data processing was adopted early with classic methods as 2D, and 3D context was considered to extract smooth layer boundaries. However, deep learning approaches that followed did not make use of this concept. Most methods for retinal layer segmentation use 2D OCT slices as their model input with few exceptions utilizing 2.5D or 3D by incorporating

Tab. 5.2: Overview of related work on OCT-based retina disease classification and layer segmentation with deep learning methods. All methods employed 2D OCT B-Scans or few stacked neighboring slices (2.5D). Method distinction is largely based on model variations. Seg. refers to segmentation, and Reg. refers to regression.

Reference	Application	DL Method
Ravenscroft et al. (2017) [398]	AMD	CNN & RF/SVM
Rasti et al. (2017) [397]	AMD	CNN Ensemble
Rong et al. (2018) [408]	AMD	CNN & Surrogate Images
Serener et al. (2019) [437]	Wet/Dry AMD	Standard CNNs
Wu et al. (2020) [545]	AMD	CNN & Attention
Wang et al. (2019) [521]	AMD	CNN & Multi-Scale
Das et al. (2019) [98]	AMD	CNN & Multi-Scale
Qiu et al. (2019) [390]	AMD	CNN & Self-Supervision
Saha et al. (2019) [421]	AMD Biomarkers	CNN
An et al. (2019) [17]	Wet/Dry AMD	CNN & Transfer Learning
Yoo et al. (2019) [571]	AMD	2 CNNs & RF
Fang et al. (2017) [131]	Retina	CNN & Graph Search
Ben-Cohen et al. (2017) [40]	Retina	Enc-Dec CNN
Roy et al. (2017) [416]	Retina	Enc-Dec CNN
Schlegl et al. (2018) [427]	Macular Fluid	Enc-Dec CNN
Shah et al. (2018) [439]	Retina	CNN Boundary Reg.
De Fauw et al. (2018) [100]	Retina & AMD	Two-Stage CNN
Sedai et al. (2018) [435]	Retina	Bayesian CNN
Orlando et al. (2019) [366]	Photoreceptor	Bayesian CNN
Masood et al. (2019) [325]	Choroid Layer	CNN
Matovinovic et al. (2019) [326]	Retina	CNN & Transfer Learning
Ngo et al. (2019) [357]	Retina	FC-NN
He et al. (2019) [195]	Retina	CNN Seg. & Reg.

neighboring slices. A lot of effort is put into architecture design and the proposition of novel architectures. Similar to eye disease classification, different data representations are usually not compared, and the role of data dimensionality is not addressed. In this thesis, extending our previous work [166], we address retinal layer segmentation from a multi-dimensional perspective. In particular, we try to overcome handcrafting architectures, which has been extensively used both for retinal layer segmentation and disease classification by introducing an efficient NAS approach for ophthalmic OCT data.

A summary of Deep learning approaches for retinal layer segmentation is given in Table 5.2.

5.2.2 Cardiovascular Imaging

The second major clinical application of OCT is cardiovascular imaging for tissue classification, treatment planning, monitoring, and assessment of stent apposition. A single IVOCT volume acquisition procedure, as outlined in Chapter 2, leads to hundreds of up to thousands of cross-sectional images showing the artery and devices such as stents. The assessment of these images requires extensive training of the interventionalist. Also, the vast amount of images acquired in a single pullback cannot all be reviewed by the practitioner during clinical routine. Therefore, various approaches for automatic IVOCT data analysis have been proposed. For example, various approaches for automatic lumen segmentation for treatment planning have been proposed [319, 344, 348, 415, 493, 570]. Also, methods for stent detection and automated assessment of potential malapposition have been proposed [315, 493, 495, 518, 519, 533].

Another important task that could be automatized is tissue characterization for guiding treatment decisions. Similar to tissue characterization and classification in retinal OCT imaging, early approaches for IVOCT tissue analysis relied on classical image processing and computer vision. The general feasibility of inferring plaque characteristics from IVOCT data has been addressed by correlating the images to histology data [564]. IVOCT has been used to measure the thickness of fibrous caps in the arterial wall to assess the risk of ruptures and potential subsequent myocardial infarction. For example, Wang et al. used a semi-automatic approach to segment the fibrous cap with a dynamic programming approach [532]. Zahnd et al. employed a similar approach and demonstrated that cap thickness can be measured in a semi-automatic way [577]. More recently, Guo et al. extracted features such as local binary patterns from IVOCT imagesm which were used in an SVM for classifying the luminal and abluminal cap layers [180]. Furthermore, calcified plaque regions have been quantified as they can increase the risk of stenosis [340]. Mehanna et al. investigated OCT's capability of quantifying calcified plaques by registering cryo images to IVOCT volumes [331]. The author's found that IVOCT is able to depict plaque volume and borders accurately.

For automatically detecting plaques, many methods relied on the backscattering and attenuation coefficient of the OCT signal. For example, Xu et al. developed a physical model of the tissue's light-scattering properties for estimating the backscattering and attenuation coefficients [557]. They found notable differences for these features when estimated for fibrous, lipid and calcified plaque regions. Van et al. found similar results for estimating the attenuation coefficient of individual A-Scans in the IVOCT images [506]. Vermeer et al. proposed an improved scattering model for estimating the attenuation coefficient at each pixel in an A-Scan [512]. These approaches were extended by Gargesha et al., where the scattering parameters were estimated from small subvolumes instead of individual A-Scans [150]. Recently, Lui et al. provided a statistical analysis that demonstrated that scattering parameters and similar measures are useful features for distinguishing different plaque regions [308]. Other approaches have investigated automatic classification methods using texture and optical properties as features. For example, Ughi et al. extracted the attenuation and backscattering coefficient, as well as texture and geometric features, from IVOCT image pixels for classification with a random forest [496]. A similar approach was pursued by Abdolmanafi et al. [5]. Celi et al. perform tissue layer segmentation based on pixel intensities, followed

by a semi-automatic search for regions with fibrous, lipid, and calcified plaque [77]. Furthermore, the segmentation of calcified plaque regions has been proposed using k-means clustering and random forests [21].

More recently, deep learning methods have found application for tissue classification in IVOCT images. Early work on deep learning-based IVOCT plaque classification relied on a CNN as a feature extractor [4]. The authors performed extensive preprocessing, followed by patch-based classification of individual pixels in the IVOCT images using the CNN AlexNet. The 2D IVOCT images were processed in their polar representation. After end-to-end training, the authors extracted the learned features from the CNN, which were then used for training conventional classifiers such as SVMs and random forests. The authors also considered transfer learning with AlexNet [262] being pretrained on ImageNet, which they found to lead to improved performance. Comparison to conventional feature extraction, for example, focusing on texture, demonstrated that CNN-based features lead to better performance. This work was extended by considering multiple different deep learning architectures for feature extraction [3]. For improved performance, ensembling methods, and feature fusion from multiple sources were considered. Xu et al. took a similar approach to the task of vulnerable plaque detection [560]. Instead of polar images, the authors trained their CNN on the Cartesian representation. Also, the authors employed extensive data augmentation. In terms of architectures, the authors employed AlexNet, different versions of VGG, and Inception. He et al. directly relied on end-to-end CNN-based classification with a custom CNN design [194]. The authors used the Cartesian B-Scan representation as their model input. Another approach directly performed segmentation of calcified plaque regions using the encoder-decoder architecture SegNet [31] as their CNN model [365]. For more consistent outputs, the authors also applied a post-processing strategy where very small predicted plaque regions were removed. Kolluru et al. took a very different approach using a 1D CNN that processes and classifies individual A-Scans [255]. This approach lead to partially noisy results as the CNN is lacking context from neighboring A-Scans. The authors addressed this problem with conditional random fields for post-processing.

Most recent approaches for IVOCT-based plaque classification have moved to end-to-end deep learning approaches. Gharaibeh et al. considered both lumen segmentation and calcified plaque segmentation using the SegNet [31] CNN architecture [171]. The authors employed a specialized data augmentation strategy for the polar image representation, where crops were taken arbitrarily across the original M-Scans. The authors employed this approach in the context of finite element modeling for treatment planning and stent apposition in the artery [170]. Another segmentation approach employed both SegNet [31] and DeepLabV3+ [83] for segmentation of different plaque types [282]. The authors also employed conditional random fields for smoothing the predicted segmentation maps. In addition, the authors provided a comparison to previous A-Scan-based approaches using 1D CNNs and demonstrated that using 2D polar images improves the performance significantly. Similarly, Liu et al. employed an encoder-decoder architecture for segmenting several different tissue types in Cartesian IVOCT images [307].

Besides encoder-decoder models for segmentation, several approaches employed patch-based approaches, usually classifying the center pixel. Cheimariotis et al. relied on a pretrained AlexNet CNN model for patch-wise classification of polar IVOCT image

crops [79]. The crops were obtained from extensively pre-processed images, which also included lumen flattening. A similar approach was taken by Ren et al. using the VGG CNN model [401]. A major difference is the use of rectangular image crops instead of quadratic crops. Also, the authors proposed to fuse RGB and local binary pattern image representations in the CNN's input channel. An approach by Athanasiou et al. is based on pixel-wise classification with a CNN that processes IVOCT B-Scans in their Cartesian representation [22]. In contrast to other approaches, the authors first apply a pre-processing technique where normal tissue is detected. Then, abnormal tissue regions are classified by the CNN. Liu et al. largely employed an established pipeline using IVOCT B-Scans in their polar representation and CNNs for classification of different plaque types [305]. The authors demonstrated performance improvements by employing an ensemble of CNNs. Zhang et al. presented an approach where CNN-based features and handcrafted features are fused, similar to previous approaches [586]. However, the authors investigated the simultaneous use of IVOCT and intravascular ultrasound.

Lee et al. took the previous approach of using 1D CNNs for A-Scan processing combined with handcrafted morphological lumen features [283]. Both feature sources are combined with a random forest classifier. The authors found that the combination of features appears to improve performance. Furthermore, the authors employed an active learning approach to improve the quality of noisily labeled data. Another recent approach by Abdolmanafi et al. relied on the established approach of using a CNN for extraction and a random forest for classification [2]. A difference to previous work is the extended task of performing both plaque detection and, in case a pathological tissue region is found, plaque classification. An approach that fundamentally differs from all other approaches was present by Luo et al. [318]. Here, a deep reinforcement learning framework was proposed where an agent learns to track a plaque region across consecutive slices along an IVOCT pullback. Overall, deep learning has been extensively applied for IVOCT-based tissue classification. In particular, very recently, a multitude of approaches has emerged.

Summarized, IVOCT-based plaque classification started with low-dimensional features extracted from A-Scans, similar to the development in ophthalmology. Newer deep learning methods began to use full 1D A-Scan information to overcome manual feature engineering. Most recent deep learning methods employ either 2D polar or 2D Cartesian data representations, which are also most commonly used in clinical practice. While both 2D data representations have been studied, a systematic comparison between the two data representations is still lacking. In particular, there is no analysis of data representation-specific techniques such as data augmentation and the effectiveness of transfer learning with different representations. Thus, for IVOCT-based deep learning, a systematic analysis of IVOCT's most important two data representations is still missing. As a part of this thesis, following our previous work [157, 159, 160], we address these shortcomings with an in-depth analysis.

An overview of deep learning applications for IVOCT tissue classification is given in Table 5.3.

Tab. 5.3: Overview of related work on IVOCT-based plaque classification and detection with deep learning methods. Method distinction is largely based on model variations and data representations. The data representation can be polar or Cartesian (Cart.).

Reference	DL Method	Data Representation
Abdolmanafi et al. (2017) [4]	CNN & RF/SVM	2D Polar
Xu et al. 2017 [560]	CNN & SVM	2D Cart.
Abdolmanafi et al. (2018) [3]	CNN & RF/SVM	2D Polar
He et al. (2018) [194]	CNN	2D Cart.
Oliveira et al. (2018) [365]	CNN	2D Cart.
Kolluru et al. (2018) [255]	CNN	1D
Gharaibeh et al. (2019) [171]	CNN	2D Polar
Gharaibeh et al. (2019) [170]	CNN	2D Polar
Lee et al. (2019) [282]	CNN	2D Polar
Liu et al. (2019) [307]	CNN	2D Cart.
Cheimariotis et al. (2019) [79]	CNN	2D Polar
Ren et al. (2019) [401]	CNN	2D Polar
Athanasiou et al. (2019) [22]	CNN	2D Cart.
Liu et al. (2019) [305]	CNN	2D Polar
Zhang et al. (2019) [586]	CNN	2D Polar
Lee et al. (2020) [283]	CNN & RF	1D
Abdolmanafi et al. (2020) [2]	CNN & RF	2D Polar
Luo et al. (2019) [318]	CNN & RL	2D Cart.

5.3 OCT-Based Pose and Motion Estimation

Intraoperative applications are another field where OCT can be employed. Here, OCT systems can be used to provide 2D [120, 122, 270, 465], 3D [49, 66, 123] and also 4D [76, 513] visualizations for the surgeon. Besides visualization, OCT can also be employed as a sensory system to provide information on surgical tool poses or tissue motion. Pose and motion estimation are traditionally not solved with OCT systems, but conventional tracking and computer vision approaches. Pose or motion estimation of surgical and patients is a typical problem in many clinical scenarios, for example, minimally invasive surgery (MIS) [57], transcranial magnetic stimulation [404] or motion compensation for visualization systems [587].

For pose estimation and tracking, common commercially available optical and electromagnetic (EM) tracking systems reach an accuracy of $0.2\,\mathrm{mm}$ to $1\,\mathrm{mm}$ [260]. For optical tracking, a mean tracking error of $0.22\,\mathrm{mm}$ has been achieved for clinical setups [126]. EM tracking operates without a line of sight but generally reaches lower accuracy with a typical root mean square error of $1\,\mathrm{mm}$ [139]. While these tracking systems are frequently employed, their tracking accuracy or marker size can represent a limitation for some small-scale MIS scenarios such as ophthalmic surgery, cochleostomy or neurosurgery.

One class of approaches tries to overcome this problem using machine learning and computer vision techniques instead of large markers. In particular, CNNs have been used for pose estimation in the natural image domain with RGB-D images. Usually, the goal is to infer an object's pose from an image without using any tracking markers that are attached to the object. One of the first approaches was presented by Wohlhart et al., where a semantic descriptor was learned that separates image patches by object type and pose [541]. Object recognition and pose estimation are performed by a nearest neighbor search, which matches an image patch to a training sample based on their descriptors. The pose estimation is coarse and highly dependent on the density of training samples in the pose space. Krull et al. took an analysis-by-synthesis approach for 6D pose estimation in RGB-D images [264]. Rendered and observed image representations are fed as channels into a 2D CNN to predict an energy function value that is related to the target pose. Kehl et al. employ CNNs in an unsupervised fashion on RGB-D patches for feature learning and subsequent 6D pose estimation [241].

More recently, Zeng et al. addressed pose estimation by first segmenting the objects of interest [584]. Then, predefined model shapes are fit to the segmentation maps for 6D pose estimation. A self-supervised learning scheme is employed to avoid extensive manual data annotation. Xiang et al. propose an end-to-end learning approach where their model PoseCNN directly predicts objects' position and orientation within the image [549]. The authors design a multi-task CNN that performs semantic segmentation of several objects, predicts each object's center point, and regresses the object's orientation as a quaternion. The authors also proposed a specialized loss function for estimating the rotation of symmetric objects. Another approach addressed pose estimation by predicting object's bounding box corners using a modified version of the object detection framework YoloV3 [399] where the usual 2D image bounding boxes are replaced by 3D boxes that can be used for 6D pose estimation [487]. An approach by Wang et al. refined 6D pose estimation by rendering the known object and iteratively matching it

to the real image [295]. Wang et al. propose a method where RGB images and point clouds are fused in a deep learning model for pose estimation of known objects [520]. Depth and color information, are fused on a pixel level and following networks perform a per-pixel prediction of potential object poses and probabilities, followed by a selection of the most likely object poses. An iterative refinement procedure follows this process for improved pose estimates.

A problem related to pose estimation is the task of motion estimation. Tracking multiple subsequent poses can be described as motion. Motion is typically described by temporal changes between images, for example, with a motion vector or an entire motion field. Traditionally, motion estimation is addressed with methods such as optical flow [317] where pixel-wise motion between frames is estimated, template matching, for example, using phase correlation [601], or feature-based matching, for example, using image descriptors such as SIFT [314]. For these problems, deep learning has also been frequently employed in the natural image domain.

For example, learning optical flow has been addressed using a deep learning model called FlowNet [110]. Here, the authors process two subsequent frames with two parallel CNN paths. Features from the two paths are fused using a correlation layer. After a typical encoder path, a decoder path upsamples the network's feature maps, leading to a final prediction of optical flow. The method is trained in a supervised manner, using artificially generated data. FlowNet2.0 improved the proposed method using an improved training schedule and multiple stacked FlowNets for large motion [212]. To overcome the shortage of training data, Ren et al. proposed an unsupervised training scheme where the training loss is calculated by warping the moving frame onto the fixed frame, based on the predicted optical flow, using a photometric loss function [402]. Another approach extended this method using a spatial pyramid to capture multiple scales and large motion within the network [394].

Another typical motion estimation problem in computer vision is ego-motion estimation, for example, for simultaneous localization and mapping. This problem has also been addressed using CNNs. Costante et al. employ a CNN that takes a pre-computed dense optical flow representation, encoded as an RGB image, as its input and computes the 6D camera motion vector [95]. The model is trained in a supervised manner using the MSE loss. Zhou et al. extended this approach in an unsupervised manner [592]. First, a CNN is used to predict a depth image from a target frame. Then, a second CNN that uses three consecutive frames as its input predicts the relative pose change between the target frame and its neighboring frames. The depth map and the relative poses are used to map the neighboring frames onto the target frame. Then, similar to unsupervised optical flow estimation, the photometric loss is used for training. Yin et al. extend this approach by simultaneously estimating optical flow and ego camera motion from several consecutive frames [568].

The field of human pose and motion estimation is also extensively studied in computer vision. This research area is specifically targeted at the pose and motion of human body parts, which is not immediately related to the problem of object pose and motion estimation, as required for medical interventions. Therefore, this field is not discussed in this thesis. A recent, comprehensive overview is given by Sarafianos et al. [423] and Wang et al. [525].

Machine learning and computer vision-based techniques for pose and motion esti-

mation have also gained popularity in clinical applications [57]. In MIS environments, pose estimation is used for tracking of surgical tools or patients from endoscopic RGB videos. Allan et al. performed tracking and 3D pose estimation of surgical tools from videos using linear Kalman filters [13]. Recently, CNNs have been applied for the localization of tools in robot-assisted MIS surgery [424]. Here, the authors employ a model based on Faster RCNN [400] for the localization of surgical tools. The network makes use of both RGB images and pre-computed optical flow. Jin et al. followed a similar approach, also relying on Faster RCNN for its object detection pipeline from 2D RGB images [227]. Moreover, Garcia et al. employ fully convolutional networks for real-time segmentation and tracking of tools [149] from 2D RGB images. To overcome the long processing times of their CNN, the authors build a pipeline where the last available segmentation is rigidly registered to the current frame for providing continuous segmentation masks. Laina et al. propose a method for simultaneous segmentation and tool joint localization of surgical tools [266]. For the effective integration of the two approaches, the localization problem is formulated as a heatmap regression problem such that the network can share parameters for predicting a segmentation map and a regression map. Similarly, Du et al. address 2D instrument pose estimation by predicting a dense heatmap of tool joint positions within the image [115]. Their CNN model also uses 2D RGB images and covers multiple different tools.

While most approaches for surgical tool detection and pose estimation rely on 2D, frame-wise processing, tasks such as surgical workflow analysis or phase classification require temporal context for capturing motion. The CATARACTS challenge resulted in multiple approaches for tool annotation in surgical videos [11]. Here, instead of localizing tools, the goal is to detect the presence of a tool for potential workflow analysis. While most approaches relied on frame-wise detection, one approach took a spatio-temporal approach by extracting features from 2D images using a CNN, followed by temporal processing using RNNs. Explicit processing of full 3D spatio-temporal data has been performed by Colleoni et al. for surgical tool tracking [93]. The authors employed a 3D CNN for processing short RGB video sequences. Jin et al. also process 3D spatio-temporal data using a CNN followed by an LSTM for the task of surgical phase classification [228]. Chen et al. pursued a similar approach, augmented by unsupervised pretraining using generative adversarial networks [85]. Funke et al. also employ a CNN, followed by an LSTM [142]. A difference to other approaches is the use of several different self-supervised pretraining techniques before fine-tuning the network in a supervised fashion for surgical phase classification.

Motion estimation is also a relevant problem for image-guided interventions where respiratory motion occurs. Typically, an additional tracking method provides surrogate motion estimates which are correlated to the actual organ movements. Alternatively, image-based motion estimation can be performed where typical approaches use deformable registration methods [54], which is related to methods such as optical flow. Deformable medical image registration has been addressed using deep learning methods. For example, Yang et al. predict a dense deformation field using an encoder-decoder CNN that takes the moving and the fixed 3D MR image as the input [565]. The model is trained in a supervised fashion using a ground-truth from a different, slower registration algorithm. Also, a refinement process is applied after predicting the deformation field. Sentker et al. took a similar approach for registering 3D CT images [436]. The

authors also incorporated an uncertainty estimation using Monte Carlo dropout. Wu et al. proposed an unsupervised approach where a convolutional auto-encoder is used to obtain a low-dimensional feature representation of medical image volumes [544]. Then, deformable registration is performed using a feature-based registration method. Another approach learns unsupervised deformable image registration in an end-to-end fashion [514]. Here, the idea is to use the deformation field predicted by a CNN to warp the moving image onto the fixed image and then use a similarity metric as the model's loss function. This concept is related to unsupervised optical flow estimation, as proposed by Zhou et al. [592].

For the imaging modality OCT, pose and motion estimation have also been addressed in different application contexts. For laser cochleostomy, an OCT-based pose estimation framework has been proposed [590]. Artificial landmarks are carved into the patient's cochlea with a laser which are used for relative movement tracking. The high accuracy results imply the usability of OCT data for pose estimation and tracking. Moreover, tracking of a ROI has been performed with maximum intensity projections (MIPs) and handcrafted feature registration [276]. Again, this approach leverages 2D depth representations instead of full volumetric information. Irsch et al. considered the problem of OCT-based motion compensation and employed an algorithm for tissue motion estimation from 1D A-Scans [215]. Another approach investigated high-speed OCT imaging for tracking and performed OCT-based motion estimation using phase correlation [428]. Deep learning approaches for OCT-based motion estimation have been rare so far. Laves et al. investigated the feasibility of learning optical flow using 2D maximum-intensity projections [275]. The use of full 3D volume information was not addressed.

Summarized, deep learning-based pose estimation and motion estimation has been frequently addressed in the natural image domain. There are deep learning methods for predicting an object's 6D pose in a 2D RGB image. Also, motion estimation has been addressed in the context of deep learning-based optical flow estimation or for the prediction of a motion vector for ego-motion estimation. While considering temporal context, the methods typically make use of two or three neighboring frames in a video, which puts the approach into a 2.5D category in terms of data dimensionality. In the medical image domain, pose estimation is relevant in the context of surgical tool detection and pose estimation where CNNs are used with 2D RGB images. Motion is relevant for surgical workflow analysis, where 3D spatio-temporal methods are employed. The problem of respiratory motion estimation is addressed as a registration problem between two 2D or 3D images. For the imaging modality OCT, there are a few classic approaches for pose and motion estimation. A deep learning approach using 2.5D data representations has been proposed, however, there is no use of full 3D volumes or even 4D spatio-temporal data. In this thesis, based on our previous work [164], we address OCT-based pose estimation with a focus on 2D and 3D spatial data representations. Furthermore, we focus on OCT-based motion estimation and extend the problem to 3.5D and 4D data, while also considering motion forecasting [44, 156].

An overview of deep learning methods for pose and motion estimation in the medical domain is given in Table 5.4.

Tab. 5.4: Overview of related work on deep learning-based pose and motion estimation in medical applications. Method distinction is largely based on the application context, deep learning method, and data representation. OF refers to optical flow images.

Reference	Application	DL Method	Data Rep.
Sarikaya et al. (2017) [424]	Tool detection	Faster RCNN	RGB & OF
Jin et al. (2018) [227]	Tool detection	Faster RCNN	RGB
Garcia et al. (2016) [149]	Tool detection	CNN segmentation	RGB
Laina et al. (2017) [266]	Tool pose	CNN segmentation & heatmap regression	RGB
Du et al. (2018) [115]	Tool pose	CNN heatmap regression	RGB
Al et al. (2019) [11]	Workflow	CNN & RNN	RGB-T
Colleoni et al. (2019) [93]	Workflow	3D CNN	RGB-T
Jin et al. (2017) [228]	Workflow	CNN-LSTM	RGB-T
Chen et al. (2018) [85]	Workflow	CNN-LSTM & GAN	RGB-T
Funke et al. (2018) [142]	Workflow	CNN-LSTM & Pretraining	RGB-T
Yang et al. (2017) [565]	Registration	CNN & Supervised	3D MRI
Sentker et al. (2018) [436]	Registration	CNN & Supervised	3D CT
Wu et al. (2015) [544]	Registration	CNN & Unsupervised	3D MRI
de Vos et al. (2019) [514]	Registration	CNN & Unsupervised & End-to-End	3D MRI
Laves et al. (2019) [275]	Optical Flow	CNN & FlowNet	2.5D OCT

5.4 MRI-Based Left Ventricle Quantification

Left ventricle (LV) quantification from medical image data is often employed for assessment of cardiac function and for diagnosing diseases [234]. Relevant LV indices include the myocardium and cavity area, three LV cavity dimensions, six regional wall thickness (RWT) parameters, and the cardiac phase (systole and diastole). In clinical practice, additional measures such as ejection fraction are often calculated. Different imaging modalities can be used for quantification, including echocardiography, cine MRI, and cardiac CT. While echocardiography is frequently employed, it is limited in terms of image quality, and the result is operator-dependent [385]. Currently, cardiac MRI is considered the gold standard for LV quantification, and cardiac CT is considered as a viable alternative [253]. Independent of the imaging modality, LV indices are usually obtained by manual segmentation of the myocardium, which is time-consuming and associated with a high intra- and inter-observer variability [469]. Also, the problem is challenging due to the high variability of cardiac structure between patients and deformation during the cardiac cycle. Therefore, a lot of automatic LV segmentation and quantification methods have been proposed. In terms of data dimensionality, LV quantification can be considered a 2D image processing problem if individual image frames within the cardiac cycle are processed. Since function assessment requires observation across the entire cardiac cycle, the problem can also be considered to be 3D spatio-temporal. The problem can be extended to full 4D spatio-temporal data when using full volumetric data, which has been shown to be advantageous for echocardiography and leads to better volume estimates [269]. Similarly, 4D cardiac MRI has found clinical application [300].

For echocardiography, early methods for LV segmentation relied on semi-automatic methods to obtain a model of the left ventricle that is deformed over time [152]. Similarly, Caiani et al. proposed a method where initial, manual markings of the endocardial border are tracked over time [70]. Another method models the LV as a cubic hermite spline [453]. To account for deformation over time, the model can be translated, rotated, and scaled, and each control point can be varied to enable local deformation. A Kalman filter is used for tracking the LV deformation over time. More recently, deep learning approaches have been presented for LV segmentation from echocardiography images. For example, Chen et al. address the problem using fully convolutional 2D CNNs [82]. Specifically, they try to overcome the general problem of noisy borders and artifacts in ultrasound image segmentation with a cross-domain approach. The authors hypothesize that building a CNN that solves multiple ultrasound segmentation problems simultaneously should achieve better generalizable performance. The author's CNN consists of a domain-independent, initial processing path which is split into different paths for the different segmentation problems. The authors demonstrate that joint learning improves LV segmentation performance. Smistad et al. also investigate 2D CNNs for LV segmentation [455]. The authors try to tackle the problem of limited annotated data availability with a student-teacher approach. First, the authors obtain automated segmentations with a basic Kalman-based method, proposed in [453], for a larger dataset including images without expert annotations. The CNN is trained using these annotations and compared to the Kalman method, showing improved performance for the Hausdorff distance.

Oktay et al. propose anatomically constrained neural networks for several tasks,

including LV segmentation from echocardiography images [363]. Here, the idea is to incorporate anatomical priors such as shape into the learning process. First, an autoencoder is trained to learn a lower-dimensional representation of the label map. An additional CNN path ensures that the representation can be predicted from the original intensity images. Then, during training of the segmentation network, predicted segmentation masks are again encoded by the fixed encoder part of the autoencoder. The calculated representation is compared to a representation of the ground-truth mask using the L2 loss. This loss is also used for training the segmentation network. The authors found that this strategy improves LV segmentation performance. Jafari et al. focused on the temporal aspect of echocardiography data by employing a 3D spatio-temporal segmentation method [220]. The authors design a convolutional and recurrent model that first encodes individual frames into a lower-dimensional representation. Then, a bidirectional convolutional LSTM also performs temporal processing across slices. Finally, a decoder outputs individual segmentation masks for each image. The authors also consider temporal information by including an additional input path that processes pre-computed optical flow images between frames. In another work, Smistad et al. performed left ventricle segmentation using a fully convolutional architecture while focusing on the aspect of real-time processing [454]. The CNN processed the 2D images frame-by-frame. Recently, Azarmehr et al. compared several different CNNs for segmenting the LV from echocardiography images [30]. The authors found that the original U-Net architecture [409] without any modifications outperformed other newly proposed methods.

For cardiac MRI, initial LV segmentation and quantification methods relied on conventional image processing techniques such as deformable templates, active contour models, and level sets. Lee et al. proposed a segmentation method where region growing with iterative thresholding was employed for segmentation of the LV endocardium, followed by segmenting the epicardium using an active contour model [281]. Paragios employed a level set method for LV segmentation where local and global constraints, as well as temporal consistency, are introduced to the problem [370]. Kaus et al. build a deformable model for LV segmentation using prior knowledge from an annotated dataset [238]. Multiple similar approaches have been introduced, which are reviewed and discussed by Ngo et al. [358]. Ngo et al. also proposed a method where a traditional level set method is fused with a deep learning approach.

In recent years, deep learning methods have been employed frequently for MRI-based LV segmentation and quantification. For obtaining LV indices, one approach is to segment the myocardium with a CNN and calculate relevant metrics afterward. For example, Avendi et al. combined several 2D CNNs and deformable models [26]. The first step of their method is to detect an ROI around the heart, which is extracted from the entire cardiac MR image using a CNN. Next, the authors train a segmentation model to obtain the general shape of the myocardium. Then, an energy minimization strategy is used to adjust the predict segmentation's contour for improved performance. Finally, the LV area is derived from the obtained segmentation mask. Note that the method did not perform full LV quantification by calculating all relevant indices. Yang et al. also employ a two-step approach where a CNN first detects a bounding box around the heart [566]. Then, a U-Net model performs segmentation of the LV cavity. The authors also compare several different U-Net variations. Romaguera et al. directly segment the LV cavity

from MR images using a fully convolutional CNN, without a distinct decoder in their model [407]. They compare to several classic approaches for LV segmentation and present a slight performance improvement. Another approach performs LV segmentation in polar instead of Cartesian image space [480, 481]. First, the authors use a CNN to locate the center of the LV cavity. Then, the image is transformed into polar space, and a second CNN predicts the inner and outer border of the myocardium. Poudel et al. considered the 3D spatio-temporal learning problem by also considering neighboring slices [384].

Similar to the work by Jafari et al. for echocardiography [220], an encoder-decoder architecture, augmented by recurrent units, is employed. The authors demonstrate performance improvements over classic approaches and a slice-wise processing method. Mortazi et al. performed segmentation of multiple cardiac structures, including the LV, using multiple planar views of the heart [349]. The authors also compared MRI- and CT-based segmentation, finding similar performance. A very different approach was pursued by Mo et al. where an agent traverses the MR image to build up the contour of the LV cavity [345]. In an iterative process, patches are sampled from the MR image, which are processed by a CNN that predicts a velocity vector. This vector is used to sample the next patch for processing. Using the Poincaré map, the authors develop a stopping criteria for the trajectory generation process. The authors demonstrate performance improvement over conventional segmentation methods. A recent approach by Hu et al. combined deep learning with a conventional contour optimization algorithm based on dynamic programming [204]. First, a CNN predicts the rough contour of the endocardial and epicardial border. Then, several refinement steps follow to adjust the two borders. The authors also rely on a polar image representation for their post-processing procedure. A recent study by Tao et al. investigated the performance of CNN-based LV segmentation and LV indices calculation when using training data from different devices and different centers [483]. In general, performance improved significantly when using a multi-center and multi-device dataset for training.

As an alternative to LV segmentation and subsequent calculation of relevant indices, the values can be directly regressed from the 2D MR slices. This was proposed by Xu et el. [562] and found a wider application afterward. Here, the authors employed a 2D Auto-Encoder CNN for the extraction of relevant features. On top of the network, the authors stacked a small CNN for indices regression from the output of the Auto-Encoder. The authors compared their method to conventional feature extraction paired with random forests for direct indices regression, demonstrating a performance improvement. In an extension of their work, Xu et al. also considered temporal context by using a CNN as a feature extractor from individual slices, which was followed a recurrent neural network for aggregating temporal context [561, 563]. An additional auxiliary output was used to also predict the current cardiac phase. As a total of eleven LV indices have to be predicted for full quantification, the authors proposed a multi-task relationship learning scheme for capturing the correlation and dependence of the different indices. A similar approach was pursued by Li et al., where a multi-task relationship loss was employed [292]. Here, the authors did not explicitly incorporate temporal information and determined the cardiac cycle based on a polynomial fit to the predicted cavity area size. Jang et al. proposed to perform direct indices regression using 2D and 3D spatio-temporal CNNs [222]. In particular, the authors also design a CNN that uses alternating

2D and 3D convolutions for more efficient processing. They find a slight performance improvement when employing CNNs with 3D convolutions.

Other methods have combined segmentation and regression, for example, by regressing indices from a segmentation with an end-to-end model [528]. Here, the authors first pretrain a CNN for segmentation of the LV myocardium. Then, an additional CNN is plugged onto the model's output that processes the predicted segmentation maps and regresses the LV indices. The authors claim substantial improvement over previous direct regression methods. Xu et al. compared direct indices regression and calculation of indices from predicted segmentation masks [558]. The authors found that calculation from segmentation maps appears to be beneficial. When jointly predicting a segmentation map and LV indices, performance is higher than for regression only but lower than segmentation map-based calculation. Also, the authors find that incorporating temporal information using LSTMs slightly improves the performance. Khened et al. went one step further and built a full pipeline for cardiac disease classification [244]. First, the authors used a CNN for segmentation of cardiac structures including myocardium. Then, the authors extracted LV indices as a set of features from the segmentation maps. Finally, several different conventional classifiers are trained with these features to predict different heart diseases.

Summarized, there has been extensive research on left ventricle segmentation and quantification. Early approaches relied on classic image processing for segmenting the myocardium and calculating LV indices afterward. Recently, deep learning methods have become very popular for the problem. Some methods from the imaging modality echocardiography share similarities with approaches for cardiac MRI. In terms of data dimensionality, almost all deep learning approaches operate on 2D slices, although some conventional methods have proposed spatial 3D processing techniques. This is likely tied to a lack of public datasets with full 3D annotated volumes [562]. However, the temporal dimension plays an important role both for segmentation and direct quantification. Several approaches have demonstrated that using fused convolutional and recurrent models or 3D CNN architectures can improve indices estimation. However, a major challenge that remains is the shortage of annotated data. While most medical image analysis applications have adopted transfer learning for this problem, this aspect is still missing for LV quantification. In particular, solutions for transfer learning with 3D spatio-temporal architectures have not been proposed. In this thesis, extending our previous work [163], we address the problem of data shortage and transfer learning with multi-dimensional data. In particular, we focus on extending 2D spatial to 3D spatio-temporal CNNs while making use of the advantages of transfer learning.

An overview of deep learning methods for LV quantification and segmentation is provided in Table 5.5.

5.5 MRI-Based Multiple Sclerosis Lesion Activity Segmentation

Multiple sclerosis is an inflammatory disease of the central nervous system, which leads to disability, mostly in young adults. MS is characterized by lesions in the central

Tab. 5.5: Overview of related work on deep learning-based LV segmentation (Seg.) and quantification (Quant.). Methods are generally differentiated by application, method, and data representation employed. DM refers to a conventional deformable model approach. PM refers to the Poincaré map. DP refers to dynamic programming. Class. refers to the task of cardiac disease classification.

Reference	Application	DL Method	Data Rep.
Avendi et al. (2016) [26]	LV Seg.	CNN & DM	2D MRI
Yang et al. (2016) [566]	LV Seg.	2 CNNs	2D MRI
Romaguera et al. (2017) [407]	LV Seg.	CNN	2D MRI
Tan et al. (2018) [481]	LV Seg.	2 CNNs	2D MRI Polar
Poudel et al. (2016) [384]	LV Seg.	CNN-LSTM	2D+T MRI
Mortazi et al. (2017) [349]	LV Seg.	CNN	2.5D MRI
Mo et al. (2018) [345]	LV Seg.	CNN & PM	2D MRI
Hu et al. (2019) [204]	LV Seg.	CNN & DP	2D MRI Polar
Tao et al. (2019) [483]	LV Seg.	CNN	2D MRI
Xu et al. (2017) [562]	LV Quant.	CNN	2D MRI
Xu et al. (2018) [561]	LV Quant.	CNN-LSTM	2D+T MRI
Li et al. (2018) [292]	LV Quant.	CNN	2D MRI
Jang et al. (2018) [222]	LV Quant.	CNN	2D+T MRI
Wang et al. (2019) [528]	LV Seg. LV Quant.	CNN	2D MRI
Xu et al. (2018) [558]	LV Seg. LV Quant.	CNN-LSTM	2D+T MRI
Khened et al. (2019) [244]	LV Seg. LV Quant. Class.	CNN	2D MRI

nervous system. To track disease progression in the brain, MRI is often used. The FLAIR sequences show lesions as high-intensity regions, which allow for quantification of the disease progression [414]. To derive quantitative parameters like lesion number and volume, lesion segmentation is required. Obtaining segmentation maps is typically performed manually and represents the current gold standard [148]. As manual segmentation is time-consuming and error-prone [118], several semi- and fully-automated methods have been proposed for lesion segmentation from MRI scans.

Similar to other application fields, early approaches relied on conventional image processing methods. Van Leemput et al. proposed a probabilistic model for classifying individual voxels in MR images, learning a representation of healthy tissue using Gaussian Mixture Models (GMM) [504]. Then, lesion material is characterized by being outliers in this trained model. For more abstract parameters such as lesion load, the authors found a high correlation to lesion load, calculated from manual segmentation maps. However, a comparison of lesion maps between the automated method and expert annotations demonstrated substantial differences. A similar approach was pursued by Ait-Ali et al. where the GMM was obtained with a trimmed likelihood estimator [8]. Lesions and healthy tissue were distinguished by using the Mahalanobis distance based on the idea of distinguishing lesion voxels as outliers from the learned healthy representation. Other approaches proposed improvements to this strategy, for example, using Hidden Markov chains [62] and a mean shift algorithm [328]. Shiee et al. perform both MS lesion segmentation and general brain segmentation using an atlas-based method [444]. Schmidt et al. proposed a conventional approach that is still popular [430]. Here, thresholding is employed for obtaining initial lesion belief maps from several MR imaging modalities. Then, a region growing algorithm is used to obtain final segmentation maps. A similar approach was presented by Roura et al., where thresholding and several refinement steps were employed [413].

Besides unsupervised methods and modeling lesions as outliers, supervised approaches using conventional classifiers have been proposed. For example, Warfield et al. used a k-NN classifier paired with a registration algorithm [534]. Tissue classification is followed by matching the classified tissue regions to an Atlas of normal scans using elastic registration, repeated for several iterations. Zijdenbos et al. used a conventional fully-connected neural network for classifying MS lesions in a voxel-wise manner [598]. Input features consisted of different MR imaging modalities and additional brain tissue information (white matter, gray matter, or cerebrospinal fluid). Wu et al. also employed a k-NN classifier combined with a template-based segmentation approach [547]. Another approach by Akselrod et al. first performed segmentation of anatomical regions [9]. Then, a large number of handcrafted features is extracted from image volumes and segmentation maps for voxel-wise classification with ensembles of decision trees. A comprehensive overview of further techniques is provided by Garcia et al. [148].

Segmenting lesions in MRI scans can be considered a step within the full pipeline of MS treatment. For monitoring disease progression, lesion *activity* between two longitudinal MRI scans (baseline and follow-up) is the most important marker for inflammatory activity and disease progression in MS [372]. Lesion activity is defined as the appearance of new lesions and the enlargement of existing lesions [330]. This problem is particularly challenging as new lesions can be small, and changes are often

subtle. So far, most methods for MS lesion segmentation have only considered lesion segmentation for a single MRI volume. Thus, lesion activity is often derived from two independent segmentation maps, which is associated with high variability and inconsistencies [287]. Therefore, other approaches made use of information from the MRI volumes instead of lesion maps only. For example, image differences have been used to detect new lesions. Battaglini et al. took a subtraction between baseline and follow-up scan, masked by an initial segmentation of white matter tissue [34]. Then, thresholding is used to obtain an initial lesion mask that is improved further using shape, extent, and intensity constraints. Ganiler et al. relied on a very similar procedure, including thresholding of the difference images [146]. The authors identified several challenges and problems of image subtraction methods, including registration errors, inconsistent temporal properties, such as blood flow, and cerebrospinal fluid flow, image noise, and partial volume effects.

Other approaches have relied on deformation fields. For example, Rey et al. first performed rough alignment using a rigid registration, followed by a non-rigid registration for obtaining a 3D displacement field [403]. Based on the directions in the vector field, shrinking and enlarging lesions can be characterized. However, this approach is limited to tracking changes of existing lesions and is not suitable for detecting new ones. Cabezas et al. used both difference images and a deformation field for the detection of lesion activity [69]. An initial lesion map is obtained by subtraction and thresholding. The map is then refined using features obtained from the deformation field. The authors reported improvements over purely deformation field-based or thresholding-based approaches. A supervised learning approach was introduced by Sweeney et al. [474]. The authors extracted features from difference images and the follow-up scan, which are used to train a logistic regression model. Salem et al. extended the idea of difference and deformation field fusion with a supervised learning approach [422]. Features are extracted both from subtraction images and deformation fields, which are then fed into a logistic regression model for voxel-wise classification of lesion activity. A recent approach by Cheng et al. also considered local context around lesions between scans and multi-scale information [86]. In this way, the authors obtain additional, handcrafted features for training a logistic regression model for voxel classification.

Deep learning methods for the problem of MS lesion segmentation have primarily been considered for the task of individual scan segmentation. An early approach relied on restricted Boltzmann machines (RBM) for deep learning-based unsupervised pretraining [572]. The features obtained from this step are used for classification with a random forest. A large body of methods was presented in the context of the ISBI 2015 longitudinal lesion segmentation challenge [74]. While a majority of methods relied on the conventional methods introduced above, several deep learning approaches were presented. For example, Vaidya et al. relied on an ensemble of patch-based 3D CNNs using sub-sampling and sparse convolution operators for voxel-wise classification [500]. Also, Ghafoorian et al. used a patch-based CNN for voxel-wise classification, however, using 2D patches and 2D convolutions [168].

Brosch et al. relied on 3D CNNs with an encoder-decoder structure [64]. The authors used stacked RBMs for pretraining their encoder in a supervised manner. It is notable that this was one of the first approaches using dense prediction instead of voxel-wise prediction. Another method used a patch-based approach for voxel-wise segmentation

with 3D CNNs [502]. The authors proposed a cascaded approach where the output of one CNN is fed to another CNN for additional refinement. Birenbaum et al. used 2D CNNs and incorporated 3D context by using multiple orthogonal views [50, 51]. Also, the authors considered multiple time points with a multi-path architecture where features extracted from multiple time points are concatenated for voxel-wise prediction of a single scan's lesion map. Roy et al. employed a 3D CNN for voxel-wise classification while putting more emphasis on the different MR imaging modalities [419]. The authors design a multi-path architecture for initial, individual processing of each modality, followed by concatenation for feature fusion. An important deep learning method for brain lesion segmentation was proposed by Kamnitsas et al. [231]. Here, the authors considered both 3D CNNs for volumetric processing and 2D CNNs for slice-wise processing, demonstrating the advantage of using full volumes. Also, the authors incorporate multi-scale context in their network by using several processing paths. Predicted segmentation maps are additionally refined using conditional random fields.

Aslani et al. avoid full 3D convolutions by using a multi-branch approach with different orthogonal views and individual pathways for different MR imaging modalities [20]. The authors also consider multi-resolution context using skip connections between layers. Another recent approach by Nair et al. used an encoder-decoder 3D CNN for segmentation and focused on the aspect of uncertainty estimation [354]. The authors employ Monte Carlo dropout to obtain probabilistic predictions and use several measures of uncertainty, including sample variance, entropy, and mutual information. The authors find that high uncertainty is correlated with incorrect predictions, helping to improve performance for small lesions. Valverde et al. considered the problem of domain adaption of trained CNNs for MS lesion segmentation [501]. They propose a supervised adaptation strategy for different datasets, for example, using different scanners. Here, a pretrained CNN is adapted by retraining only the last fully-connected in a CNN for voxel-wise classification.

Some approaches have also considered unsupervised and semi-supervised deep learning approaches for MS lesion segmentation. For example, Baur et al. explored several auto-encoder 2D CNN approaches for segmenting anomalies such as MS lesions by learning a healthy representation of the brain [35]. The authors found that probabilistic, spatial, latent representations improve the segmentation task. Another approach by Atlason et al. used an auto-encoder 2D CNN with specialized output layers for unsupervised lesion segmentation [24, 25]. The authors trained the auto-encoder for normal image reconstruction, however, before the last layer, a softmax layer is added, which forces the network to partition the image into different maps. The authors observed that some of these maps correspond to high-intensity lesions such as MS lesions. Another method by Baur et al. combines unsupervised and supervised learning for MS lesion segmentation [36]. First, an auto-encoder is trained in an unsupervised manner for anomaly-based lesion detection. Then, another encoder-decoder CNN is trained for supervised lesion segmentation using both labeled data and unlabelled data with predictions obtained from the auto-encoder as an artificial ground-truth. Fenneteau et al. take a self-supervised approach for MS lesion segmentation [135]. First, the authors train a 3D CNN for an artificial localization task as a pretraining step. Here, the CNN regresses the x, y, and z location of an image crop within the entire image. In this way, representative features should be learned that help to improve performance for the actual task. Second, the

Tab. 5.6: Overview of related work on deep learning-based MS lesion segmentation. Methods are generally differentiated by method and data representation. Dense refers to dense prediction of lesion segmentation maps and voxel-wise refers to the patch-based approach where each model forward pass predicts one voxel's class.

Reference	DL Method	Data Rep.
Yoo et al. (2014) [572]	RBM (voxel-wise)	2D
Vaidya et al. (2015) [500]	CNN (voxel-wise)	3D
Ghafoorian et al. (2015) [168]	CNN (voxel-wise)	2D
Brosch et al. (2016) [64]	CNN (dense)	3D
Valverde et al. (2017) [502]	CNN (voxel-wise)	3D
Birenbaum et al. (2017) [51]	CNN (voxel-wise)	2.5D-T
Roy et al. (2018) [419]	CNN (voxel-wise)	3D
Kamnitsas et al. (2017) [231]	CNN (dense)	3D & 2D
Aslani et al (2019) [20]	CNN (dense)	2.5D
Nair et al. (2020) [354]	CNN (dense)	3D
Valverde et al. (2019) [501]	CNN (voxel-wise)	3D
Baur et al. (2018) [35]	CNN (dense) Unsupervised	2D
Atlason et al. (2019) [24]	CNN (dense) Unsupervised	2D
Baur et al. (2019) [36]	CNN (dense) Semi-supervised	2D
Fenneteau et al. (2020) [135]	CNN (voxel-wise) Self-supervised	3D

authors initialize the encoder of an encoder-decoder architecture with the pretrained weights from the self-supervision task. This second architecture is trained for normal MS lesion segmentation.

As a result, a large number of methods have been presented for MS lesion segmentation of individual scans. Early methods relied on classic image processing and conventional machine learning techniques. Similar to other applications, end-to-end deep learning methods have largely taken over the field. However, lesion activity segmentation has largely been addressed using conventional image processing methods or classical machine learning approaches with voxel-wise classification. Thus, in terms of data representations, the MS lesion segmentation problem has only been addressed as a 2D or 3D learning problem without consideration of temporal 3.5D or 4D context. In this thesis, we present the first approach for deep learning-based lesion activity segmentation [153, 158]. We consider both the 3.5D and full 4D learning problem.

An overview of all related deep learning-based MS lesion segmentation approaches is given in Table 5.6.

5.6 Deep Learning with other Multi-Dimensional Problems

While we cover a large number of OCT and MRI applications in the context of multi-dimensional deep learning, there are other problems for the two modalities with relevant deep learning approaches to consider. Also, for other imaging modalities such as CT and US, there are several multi-dimensional problems that share similarities with MRI and OCT.

For OCT, another important multi-dimensional problem is OCT angiography (OCTA). This imaging modality is used to visualize blood flow with a primary application in ophthalmology. Retinal blood flow in larger vessels can be detected and quantified using the Doppler shift, i.e., the phase shift between consecutively acquired A-Scans [529]. This has been applied to the task of detecting patients with diabetic retinopathy [530]. Wang et al. observed that patients with diabetes showed lower than normal blood flow in retinal vessels [530]. This has also been extended to blood flow quantification in microvessels using high-speed OCT [225]. The authors computed decorrelation angiography using eight consecutive 2D intensity B-Scans with decomposition into four spectral bands. The averaged, decorrelated angiography frames were averaged, and, based on slice-wise processing, a 3D angiography volume was obtained. A maximum intensity projection along the depth dimension provided an en face visualization of blood flow in the optic disc. Also, the authors demonstrated that blood flow calculated from the angiography images could serve as a marker for glaucoma detection. Moult et al. extended this idea to the problem of detecting AMD in patients [350]. Here, the authors also computed angiograms using the decorrelation of intensity B-Scans repeatedly acquired at the same spatial location. As a result, most conventional methods treated the problem of angiography as a 3D spatio-temporal problem by processing entire OCT volumes in a slice-wise fashion.

More recently, deep learning methods have been proposed in the context of OCT angiography. Guo et al. focused on the segmentation of vascular structures in en face OCT angiograms, which can provide information on the presence of diabetic retinopathy [181]. The authors avoided the spatio-temporal problem by calculating the angiography images using conventional methods. First, the superficial vascular complex in the retina was obtained from manual segmentation. Then, angiograms were calculated in the relevant areas using intensity images and decorrelation. A maximum intensity projection then provides a 2D en face view of the angiogram, which is segmented by a conventional 2D CNN. The authors employ an encoder-decoder architecture which also considers multi-scale context. The authors extended this approach by using additional inputs to the 2D CNN [182]. Besides the en face maximum intensity projection of the angiogram, the authors also use a maximum intensity projection of the normal OCT intensities and a depth image encoding the retinal thickness at each pixel location, obtained by retinal layer segmentation. Thus, the authors also consider additional spatial information in the problem but do not explicitly process the spatial depth dimension or the temporal dimension. Lauermann et al. proceeded similarly when addressing the problem of OCT angiogram quality assessment [274]. Deep learning-based temporal processing is not considered, as pre-computed angiography images are directly fed to a

CNN for classification.

OCT angiography was addressed very differently by Liu et al., who tried to estimate OCT angiograms from a time series of OCT images, thus explicitly formulating a spatio-temporal deep learning problem [309]. Ground-truth angiograms were automatically obtained by using an algorithm that considered both intensity-based decorrelation and phase difference. Then, a CNN was trained, which received four B-Scans taken at the same spatial location as the input and predicted an angiogram at that location. The four time points were processed by stacking in the CNN's channel dimension. Interestingly, the authors report an improved signal-to-noise ratio over the method that was used for generating the ground-truth. In a follow-up study, Jiang et al. investigated several different CNN methods for OCT angiogram generation [226]. This includes single- and multi-path models, an encoder-decoder CNN, and a generative adversarial network. Also, the authors tried adding additional phase information for improved angiogram prediction. The authors found that adding phase information significantly improves several measures for image quality assessment.

The problem of visualizing and assessing blood flow can also be tackled with MRI, usually using phase-contrast (PC) MRI [373] or arterial spin labeling [257]. Note that this type of imaging is particularly focused on blood flow itself, not blood flow as a surrogate for brain activity, as performed for functional MRI (fMRI). Cerebral blood flow has been visualized using MRI for generating 3D perfusion maps [104]. Flow maps were deemed useful for assessing blood flow and arterial stenosis. Blood flow measurements have also been employed for assessing cardiac function. Furthermore, Jerosch et al. demonstrated the feasibility of measuring blood flow from MR images [224]. While blood flow images can be processed and visualized in 2D or as 3D spatio-temporal images, 4D flow estimation, and visualization have been shown to improve the assessment procedure [505]. One problem with perfusion maps is the high presence of noise and a low signal-to-noise ratio. Therefore, several conventional methods for image denoising have been proposed. For example, Bibic et al. employed a wavelet-domain filtering approach, showcasing improved performance over conventional spatial denoising [48]. Liang et al. extended this approach by also employing non-local means filtering [296]. Also, a spatio-temporal approach was proposed using low-rank total variation [132]. Recently, deep learning approaches have been presented for this problem. For example, Kim et al. took an image-to-image translation approach [248]. Here, a CNN was trained to reconstruct a high-quality perfusion image, obtained from multiple measurements, based on a lower-quality image from fewer measurements. The authors demonstrated improved image quality compared to the perfusion images from fewer images. The authors employed a slice-wise 2D CNN for this problem. An extension was presented by Pinto et al [381]. Here, the authors augmented a 2D CNN approach with a signal model for improved denoising performance. An approach by Xie et al. proceeded similarly with a different 2D CNN architecture [551]. Overall, MRI processing related to blood flow is largely performed on 2D slices.

Additional multi-dimensional deep learning problems arise for the imaging modality fMRI. Here, blood flow is used to derive functional activity and creating a mapping of the brain [140]. As fMRI is able to capture functional activity, it can be used to detect and classify neurological brain disorders such as Alzheimer's disease (AD), autism spectrum disorder (ASD) and attention deficit hyperactivity disorder (ADHD). Similar to other

detection and classification problems, initial methods for these kinds of problems utilized conventional machine learning methods. Khazaee et al. constructed a connectivity matrix using a parcellation of the brain into multiple different functional regions to build a graph of brain functionality and extracted features from that graph. After using feature selection, a Naive Bayes classification model was trained for differentiating healthy, AD, and patients with mild cognitive impairment (MCI) [242]. This method was extended by using multivariate Granger causality analysis for building a directed graph representing functional brain regions [243]. Similar approaches have been pursued for the classification of ASD. For example, Iidaka et al. calculated correlation matrices from fMRI and used correlation features for training a probabilistic neural network model [211]. Correlation features were computed by taking several hundred different ROIs, averaging the spatial regions, and then calculating the correlation between the different times series. This formes a correlation matrix to be used as a feature source. Plitt took a similar approach and compared multiple different conventional machine learning methods for the task [382]. For ADHD, Park et al. also created a connectivity graph for deriving features from fMRI [371]. The features were then used for classification with an SVM. Deshpande et al. employed similar methods in combination with FC-NNs [103].

Recently, deep learning methods have taken over for classification tasks based on fMRI data. An early deep learning methods was proposed by Saraf et al. for AD classification [425]. The authors decomposed the 4D fMRI time series into 2D slices which were individually classified by a 2D CNN. Suk et al. used convolutional auto-encoders to extract discriminant features from fMRI data. [471]. Then, an HMM was used to provide an estimate of AD, modeled as a hidden state. A similar approach was presented by Zeng et al. [585]. Here, the authors first computed connectivity features, which were then used with an auto-encoder to obtain a compressed latent representation. The representation was then used for classification with a linear SVM. A method by Zou et al. combined both structural and functional MRI images with a spatial 3D CNNs [604]. The fMRI data was reduced to a 3D volume by computing voxel-wise features from the time series, which were stacked into the CNN's input channel. A similar approach was proposed by Qureshi et al., where 4D times series were aggregated into 3D volumes by removing noisy components from the time series [392]. Li et al. took a straightforward approach for aggregating 4D fMRI into a 3D volume by calculating mean and standard deviation across the time dimension and stacking both features into the model's channel dimension [293].

Dvornek et al. shifted the focus from aggregation into a spatial representation to temporal processing [117]. The authors took the time series extracted from multiple ROIs and, instead of computed correlation features as performed for conventional methods, directly processed the time series using LSTMs. Recently, in a preliminary study, we proposed full 4D deep learning for fMRI-based ASD classification [43]. We employed both spatial 3D CNNs processing a volume obtained by a statistical summary and 4D CNNs, as well as our novel cGRU-CNN3D architecture. We demonstrated that full 4D spatio-temporal processing led to the best performance. Another very recent method by Mao et al. combined multiple 4D deep learning methods for ADHD classification [320]. The authors used several parallel processing paths with a 3D CNN followed by LSTMs, a full spatio-temporal 4D CNN, and a 3D CNN followed by temporal pooling. The method was applied to AD classification. As a result, current deep learning methods for

fMRI data already make use of high-dimensional data processing.

The imaging modality CT is similar to MRI in terms of its data dimensionality. While image acquisition is usually performed for obtaining 3D image volumes, the processing is often also performed in 2D or using 2.5D projections. Also, temporal information can be involved, leading to up to 4D spatio-temporal data. Being one of the most widely used imaging modalities, CT comes with a vast amount of literature related to multi-dimensional deep learning problems. Here, we focus on the most relevant and prominent applications.

Typical tasks where CT imaging is employed include lung nodule detection, lung disease detection, and cardiac assessment. For long nodule detection, conventional methods have relied on frameworks using an algorithm for rough nodule detection, followed by false positive reduction. For example, Murphy et al. performed initial lung nodule detection by using the shape index and curvedness features [352]. Then, two stages of k-NN classification were used to reduce the false positive rate. Features were largely shape-based, including measures for nodule size, shape, dimensions, and sphericity. Other conventional approaches were similar, for example, Messay et al. used intensity thresholding and morphological operations for segmentation and detection of nodule candidates [336]. Then, a set of shape and intensity features is used to classify nodules using a Fisher linear discriminant classifier. For the classification of various lung diseases, conventional methods relied on 2D patch-based approaches, feature extraction from patches, and conventional machine learning methods. Uppaluril et al. computed gray level co-occurrence matrices for features, employed several texture features, and also used the geometric fractal dimension as a feature for the training of a Naive Bayes classifier [499]. Song et al. extend this approach by adding Gabor filter-based LBP features and HOG features to their pool of features for classification [460]. Classification is performed with a modified k-NN approach. One problem that can be tackled by cardiac CT is the detection of calcifications within coronary arteries. A classical approach by Isgum et al. used multi-atlas segmentation, thresholding, and 3D connected components to obtain candidates for calcifications around the heart [216]. Then, intensity and texture features were used for voxel-wise classification using multiple classifiers, including k-NN and an SVM. Another approach by Xie et al. used an initial segmentation to detect the heart and coronary arteries [555]. Then, filtering and thresholding were used to detect calcifications. Classic image processing and 4D CT has mostly been studied in the context of respiratory motion for radiation therapy [535] and modeling of the heart [338].

Similar to other medical imaging applications, deep learning methods recently gained traction for CT image processing. For long nodule classification, Setio et al. propose a deep learning-based system where nodule candidate detection is performed by a conventional method, followed by false positive reduction using several 2D CNNs [438]. The authors take a multi-view approach by selecting nine different planes, where each is processed by a different CNN, followed by prediction fusion. A more recent approach directly detects lung nodules from CT scans using CNNs [552]. The authors employ Faster R-CNN [400] for nodule detection in 2D CT slices. The nodule candidates are then processed by multiple 2D CNNs, each receiving a different 2D view. In another approach, full 3D volume processing was performed [597]. Here, the authors first employ an extension of the object detection framework Faster R-CNN to 3D for candidate nodule

detection. Then, a second, multi-path CNN is employed for nodule classification. Finally, the features learned from the second CNN are used to train gradient boosting machines to obtain the final classification.

For lung disease classification, Anthimopoulos et al. also rely on a conventional patch extraction system. Then, patches are classified using a conventional 2D CNN [18]. In a large-scale study, Walsh et al. used pre-segmented axial 2D CT slices for classifying multiple diseases [516]. The 2D slices were processed by a standard 2D CNN. More recent methods also employed 3D CNNs with 3D crops from full CT images [383]. The authors compared several 3D CNN variations but did not perform a comparison to 2D slice-wise approaches. For coronary calcification detection, an early deep learning approach relied on patch-wise classification where patches were detected using thresholding and morphological operations [290]. Then, several 2D CNN classified patches in terms of the presence of coronary calcifications. The different CNNs receive different orthogonal 2D views as the input. Lessmann et al. extended this approach by also using a 2D CNN for initial candidate selection in a full axial 2D CT slice [289]. More recent methods have also moved to full 3D CT image processing. For example, Ghanem et al. used 3D CT angiography images for detecting calcifications within a segmented coronary artery tree [169]. Very recently, deep learning-based 4D CT processing has also emerged, mostly, in the context of image-to-image translation and image reconstruction. Leemput et al. derived non-contrast CT images from 4D spatio-temporal perfusion CT images using fused recurrent-convolutional networks [284]. This is motivated by reduced radiation exposure for patients being treated in the context of acute stroke. Also, Clark et al. reconstruct high-quality 4D cardiac CT images from undersampled 4D CT data using a 4D encoder-decoder CNN [92]. Summarized, over the years, deep learning-based CT processing has moved towards higher-dimensional data processing, moving from slice-wise 2D processing to volumetric processing. While 4D CT is available for some problems, applications are still rare.

Given that CT images are widely employed, there are additional, more subtle methods improvements that have been proposed for different CT applications. An overview of deep learning methods for CT image data is given by Litjens et al. [301] and Halder et al. [185].

US is more closely related to OCT and offers similar data representations as OCT, ranging from 2D to 4D data representations. The multi-dimensional aspect of US is mostly relevant for applications including echocardiography, disease detection and classification, and fetal US imaging. A frequent application for disease detection is breast cancer. Similar to all other previously discussed imaging modalities and applications, early methods for automated breast cancer detection and classification relied on a conventional pipeline with feature extraction and classic machine learning models [172].

Over recent years, this was replaced by deep learning methods, mostly processing 2D US images with CNNs [306]. Han et al. first demonstrated the effectiveness of CNNs for breast lesion classification [186]. The authors made use of the Inception architecture for classifying lesions based on small 2D image crops around the lesion. The authors observed significant performance improvement over the use of classic features and SVMs. This approach was extended by Byra et al. where transfer learning from the ImageNet dataset was explored [68]. While transfer learning with CNN fine-tuning substantially improved performance, the authors also found that a color conversion strategy of the

2D US images to artificial color channels improved performance further. One way to incorporate higher-dimensional information into the problem is the use of shear-wave elastography. Here, temporal information in terms of velocity information can be used to derive 2D elastography maps. This 2D encoding of higher-dimensional data has also been used for cancer detection in breast US using 2D CNNs [595]. The use of full 3D US volumes is more common for other applications such as fetal US imaging. For example, Looney et al. performed placenta segmentation in 3D US volumes using a 3D CNN [313]. The authors relied on a multi-scale 3D CNN architecture that was previously employed for brain MRI. Another approach addressed the problem of abdomen segmentation in fetal US [432]. The authors performed initial segmentation using a multi-scale 3D CNN. Then, the segmentation maps were used in a traditional model-based segmentation algorithm to obtain improved segmentation borders. While 4D US can be acquired by obtaining 3D scans over time, deep learning methods explicitly processing this type of data are rarely found. Philip et al. used 4D US images of the heart, however, processing was applied on individual 3D volumes in the temporal sequence [379]. Thus, there are few US applications moving towards higher-dimensional data, but most methods still rely on 2D slice-wise processing.

Overall, US is widely applied in clinical practice due it safe, cheap, and fast imaging capabilities, resulting in a large body of applications where deep learning can be helpful. A more extensive overview of deep learning-based US applications is given by Liu et al. [306] and Huang et al. [209]. A survey with a focus on 3D US was conducted by Kozegar et al. [259].

Across multiple imaging modalities, there are a variety of deep learning applications with relation to multi-dimensional data. For OCT, angiography involves temporal context, which was explicitly considered in very recent work. For MRI and CT, the advantage of 3D volumetric over 2D slice-wise processing has become very evident over the last few years. MRI applications involving blood flow and brain function have also made steps towards 4D image processing with deep learning methods very recently. A similar trend can be observed for CT, however, 4D applications for this modality are focused on reconstruction and image-to-image translation. US deep learning applications show similar trends, however, 4D deep learning is rarely found for this imaging modality so far. For all modalities, we observe the same trend of higher-dimensional data processing with deep learning methods.

On overview of all relevant deep learning methods for multi-dimensional data is given in Table 5.7.

Tab. 5.7: Overview of related work on deep learning-based for other multi-dimensional problems that are not directly addressed in this thesis. Methods are generally differentiated by method, application, and data representation. Data representations include MIPs, FVs, and shear-wave elastography (SWE).

Reference	DL Method	Application	Data Rep.
Guo et al. (2018) [181]	CNN	Angiography	2D MIP
Guo et al. (2019) [182]	CNN	Angiography	2.5D MIP
Lauermann et al. (2019) [274]	CNN	Angiography	2D MIP
Liu et al. (2019) [309]	CNN	Angiography	2D-T OCT
Jiang et al. (2020) [226]	CNN	Angiography	2.5D-T OCT
Kim et al. (2018) [248]	CNN	Denoising	2D MRI
Pinto et al. (2018) [381]	CNN	Denoising	2D MRI
Xie et al. (2018) [551]	CNN	Denoising	2D MRI
Sarraf et al. (2016) [425]	CNN	AD	2D MRI
Suk et al. (2016) [471]	AE & HMM	AD	FV
Zeng et al. (2018) [585]	AE & SVM	AD	FV
Zou et al. (2017) [604]	CNN	ADHD	3D MRI
Qureshi et al. (2019) [392]	CNN	AD	3D MRI
Li et al. (2018) [293]	CNN	ASD	3D MRI
Dvornek et al. (2017) [117]	LSTM	ASD	FV-T
Bengs et al. (2019) [43]	cGRU-CNN	ASD	4D MRI
Mao et al. (2019) [320]	CNN	AD	4D MRI
Setio et al. (2016) [438]	CNN	Lung Nodules	2.5D CT
Xie et al. (2019) [552]	CNN	Lung Nodules	2.5D CT
Zhu et al. (2018) [597]	CNN	Lung Nodules	3D CT
Anthimopoulos et al. (2016) [18]	CNN	Lung Disease	2D CT
Walsh et al. (2018) [516]	CNN	Lung Disease	2D CT
Polat et al. (2019) [383]	CNN	Lung Disease	3D CT
Lessmann et al. (2017) [289]	CNN	Calcification	2.5D CT
Ghanemo et al. (2019) [169]	CNN	Calcification	3D CT
Leemput et al. (2019) [284]	RNN-CNN	Reconstruction	4D CT
Clark et al. (2019) [92]	CNN	Reconstruction	4D CT
Han et al. (2017) [186]	CNN	Breast	2D US
Byra et al. (2019) [68]	CNN	Breast	2.5D US
Zhou et al. (2018) [595]	CNN	Breast	2D SWE
Looney et al. (2017) [313]	CNN	Fetal	3D US
Schmidt et al. (2017) [432]	CNN	Fetal	3D US
Philip et al. (2019) [379]	CNN	Heart	3D US

5.7 Summary

In this chapter, we introduced the different application scenarios we consider throughout this thesis. We examined each application's development in the literature over time and highlighted open problems. This also allows us to examine general trends in medical image analysis.

Overall, the problem of multi-dimensional data is present across a large range of medical image analysis problems. Following the timeline, several patterns in method development can be observed that repeat across imaging modalities and applications. Very early image processing approaches did not employ predictive machine learning at all and focused on modeling the problem and employing conventional computer vision techniques. This encapsulates modeling of tissue and organ deformation [179] as well as classic segmentation using active contour models [567] or level-set methods [370]. These methods generally made use of both local context in 2D and 3D image neighborhoods as well as global context, for example, through anatomical priors [238]. Thus, very early methods already tried to make use of spatial, multi-dimensional information in medical image data.

The next major development that can be observed was the rise of feature extraction methods, paired with conventional machine learning methods such as SVMs and RFs. Here, multi-dimensional context generally moved to the background as common feature extraction methods focused on capturing pixel-wise or local features [460]. Also, segmentation problems usually relied on a pixel- or voxel-wise prediction without consideration of context at the model output [598]. Some approaches applied additional post-processing methods to recover context and obtain smooth prediction maps [148]. As a result, this class of approaches did not make extensive use of high- or multi-dimensional context.

With the emergence of deep learning, the traditional pipeline of feature extraction and machine learning was fused in a single, trainable model. Early approaches relied on similar image context as small patches were fed into CNNs for predicting the patch center pixel's class [131]. More context was incorporated by the introduction of dense prediction [409] where entire segmentation maps were predicted. Still, most methods processed images by using 2D representations, even when higher-dimensional temporal or spatial information was available [301]. Some approaches started to incorporate higher-dimensional context by using multiple 2D slices, for example, from a temporal sequence [309] or by extracting projections or slices from a full volume [438]. Recently, methods started to shift to processing full 3D data representations, for example, by using spatio-temporal sequences of 2D images [321] or full 3D volumes [231]. Overall, this trend highlights that using high-dimensional context is gaining traction as higher performance was achieved across applications.

Still, there are many open questions with respect to data dimensionality that remain unanswered. Although OCT is inherently multi-dimensional, there are hardly any studies and methods regarding the nature of higher-dimensional data processing. Comparisons between data representations and multi-dimensional deep learning methods are very rare. This spans across problems including force, pose, and motion estimation, as well as tissue classification and segmentation in ophthalmology and cardiology. While MRI-based processing has started to move from 2D to 3D spatial processing [231],

spatio-temporal context including both short-term time series and long-term longitudinal data is still rarely incorporated. Across all applications and modalities, 4D deep learning methods and studies remain a largely untapped field.

6 Experimental Results

In this chapter, we demonstrate the application of our methods we introduced in Chapter 4 to a variety of problems in the field of medical image analysis. For each application field that we introduced and reviewed in Chapter 5, we describe the application scenario's key challenges in terms of multi-dimensional data. Then, we outline the data acquisition procedure and dataset characteristics. We present quantitative and qualitative results, and finally, we discuss application-specific insights. We start with applications that come with lower-dimensional data and continue with increasing data dimensionality up to 4D data applications. First, we consider fiber-based force estimation with OCT, where we consider 1D and 2D data processing. Also, we study OCT-based retina segmentation with 1D A-Scans and 2D B-Scans. Second, we consider 2D and 2.5D data representations in the context of IVOCT-based plaque classification in coronary arteries. Third, we extend 2D applications to 3D for the tasks of left ventricle quantification from MRI slices and for OCT-based pose estimation for surgical interventions. Fourth, we consider motion estimation, multiple-sclerosis lesion activity segmentation, and vision-based force estimation where we employ 2D, 3D, 3.5D, and 4D data. A comprehensive overview of all our methods and the problems they are applied to is given in Figure 6.1.

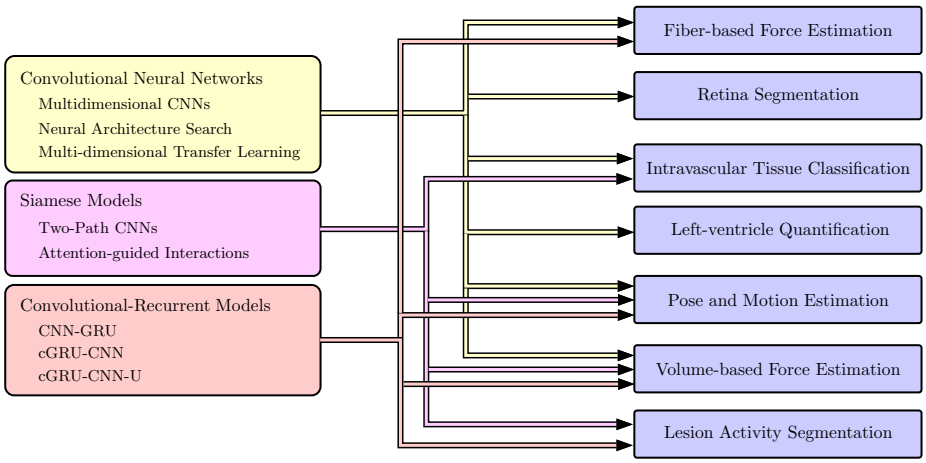


Fig. 6.1: An overview of applications in relation to our proposed methods.

6.1 1D and 2D Data: OCT Fiber-Based Force Estimation

As described in Section 5.1, needle tip force estimation is an important aspect of many surgical interventions. Integrating force sensors directly into a needle's tip is challenging due to size. A promising approach is to use fiber-optic setups as they only take up a small space, they are biocompatible, and MRI-compatible [38].

Here, we present a fiber-optic approach using the imaging modality OCT for force estimation at the needle tip. A single OCT fiber is embedded into a ferrule with an epoxy layer applied on top of it. A sharp metal tip is mounted on top of the epoxy, layer to facilitate tissue insertion. Axial forces acting on the needle tip lead to a deformation of the epoxy layer which is imaged by the OCT fiber. Thus, forces can be inferred from the OCT signal. In general, this needle design is easy to manufacture and flexible as no precise fiber placement is required, the needle tip's shape can be changed, and the epoxy layer's thickness and composition can be varied. Thus, softer epoxy resin could be used for application scenarios which require a high sensitivity such as microsurgery and stiffer epoxy resin could be used for large forces which occur, for example, during biopsy [38]. However, this approach comes with challenges for calibration and force estimation. In particular, a robust, non-linear model is required, which maps the deformations observed in the OCT images to forces. The signal can be understood as 2D spatio-temporal data with a spatial and a temporal dimension.

In the context of our research questions, we investigate two key aspects of this problem. First, we study whether using 1D spatial or 2D spatio-temporal data is advantageous for this type of problem. Second, we study which type of multi-dimensional deep learning concept is preferable. In particular, we employ 1D and 2D CNNs, CNN-GRU, CNN-cGRU, and the cGRU-CNN architecture we proposed in Section 4.3.

Methods and Datasets

Problem Definition. Our force sensing needle design uses OCT which produces series of 1D A-scans that need to be mapped to forces. Thus, we consider a 2D spatio-temporal learning problem with a set of n_t consecutive, cropped 1D A-scans $M_{t_i} = \{A_{t_{i-n_t+1}}, ..., A_{t_{i-1}}, A_{t_i}\}$ with $A_{t_i} \in \mathbb{R}^{n_d}$ where n_d denotes the A-Scan's cropped size. The resulting M-Scan $M_{t_i} \in \mathbb{R}^{n_t \times n_d}$ is used to estimate axial target forces $\mathrm{y}_{t_i} \in \mathbb{R}$. Thus, we try to find deep learning models $f_M : \mathbb{R}^{n_t \times n_d} \rightarrow \mathbb{R}$.

Needle Design and Experimental Setup. Our proposed needle tip force sensing mechanism and calibration setup are shown in Figure 6.2. The needle's base is a ferrule with a diameter of $1.25\,\mathrm{mm}$, which holds the OCT fiber. The fiber's and ferrule's ends are smoothed down to the same level. On top, we apply an epoxy resin layer with a height of $0.5\,\mathrm{mm}$ using Norland Optical Adhesive 63. On top of the layer, a cone-shaped brass tip is attached. The needle's OCT fiber is attached to a frequency domain OCT device (Thorlabs Telesto I). A force sensor (ATI Nano43) for ground-truth annotation is mounted between the needle and a linear stage that moves the needle along its axial direction.

For obtaining a training dataset, the tip is deformed with random magnitude and

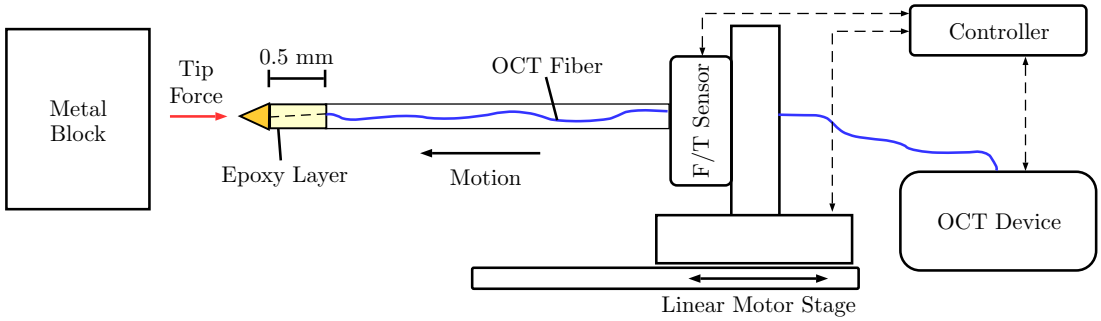


Fig. 6.2: Schematic drawing of the needle and the calibration setup. Not to scale. The needle contains an OCT fiber that images a deformable epoxy layer below the needle tip. Forces are measured by the force sensor at the base. The setup is moved with a linear stage. A shielding tube is decoupled from the needle and the force sensor and prevents shaft friction measurements for tissue insertion experiments.

velocity to create a large dataset with extensive force variations being covered. Next, we validate the needle in tissue insertion experiments, see Figure 6.3. Obtaining ground-truth tip forces is challenging for this case as the force sensor at the base measures both axial tip forces and friction forces acting on the shaft. Therefore, we use a shielding tube, which is decoupled from the needle and the force sensor. This allows for the measurement of axial tip forces for comparison to our needle tip sensing mechanism. Note that the shielding tube is a workaround for validation experiments but not for practical application as the stiff tube would increase trauma. We perform insertion experiments into a freshly resected human prostate.

Data Acquisition and Datasets. The OCT device we use is a frequency-domain-OCT that uses interferometry with near-infrared light to acquire 1D A-Scans with a rate of $5500\,\mathrm{Hz}$. The light's wavelength of $1325\,\mathrm{nm}$ allows for imaging of the inner structure of scattering materials with up to $1\,\mathrm{mm}$ depth. Considering the epoxy layer's thickness of $\approx 0.5\,\mathrm{mm}$, this allows for imaging of the entire layer up to the surface of the metal tip. As the metal cannot be penetrated by the light, we crop to the relevant signal part of the A-Scan. The force sensor for ground-truth annotation acquires data at $500\,\mathrm{Hz}$. Therefore, the OCT and force sensor data streams need to be synchronized and matched. We use the streams' timestamps for synchronization and nearest-neighbor interpolation to assign an A-scan to each force measurement. To construct a sequence, we add n_t previous A-scans to each A-scan with an assigned force value. Note that this leads to only a few A-Scans being labeled. As we use sequences of length n_t with only the first A-Scan A_{t_i} being labeled, the spatio-temporal models can still make use of the additional A-Scans obtained by the higher sampling rate. We show cropped A-Scans and the corresponding force ground-truth in Figure 6.4.

We acquire a calibration dataset containing approximately $90\,000$ sequences of A-scans, each labeled with a scalar, axial force. We use $80\,\%$ of the data for training and validation and $20\,\%$ for testing. There is no overlap between the sequences from the different sets. We tune hyperparameters based on validation performance. The hyperparameters include n_t, n_d, α_{lr}, p_{di}, p_{do} and network depth. As a baseline, we use

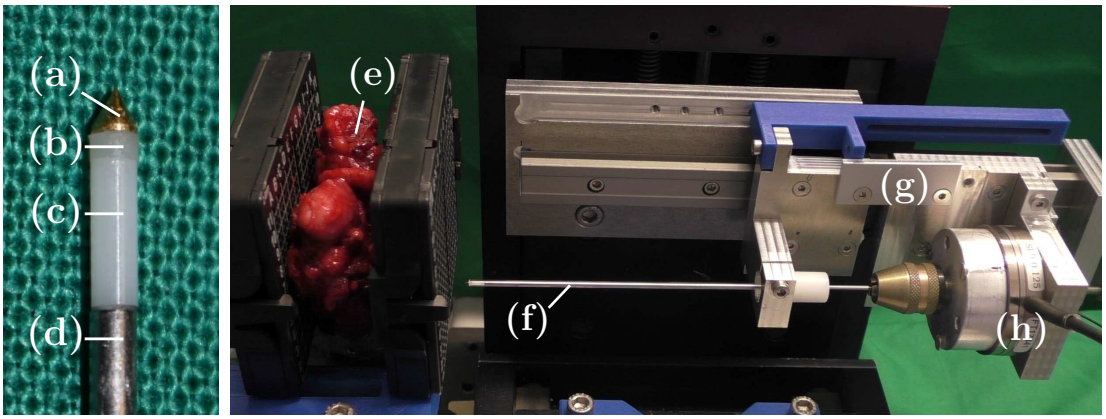


Fig. 6.3: Needle design (left) and a photograph of the experimental setup for the prostate insertion experiment (right). The brass tip (a) is attached to the epoxy layer (b), which is glued to the ferrule with the embedded OCT fiber (c). The ferrule is attached to the needle base (d) with a diameter of $1.25\,\mathrm{mm}$. For the prostate (e) insertion experiment, the needle is decoupled with a shielding glass tube (f). The linear stage (g) moves the needle, and the force sensor (h) acquires reference data.

$n_t = 50$, a crop size of $n_d = 70$ pixels, $p_{do} = 0.2$, $p_{di} = 0.1$ and $\alpha_{lr} = 10^{-4}$.

Deep Learning Models. We consider several deep learning models based on the architectures introduced in Chapter 4 to map M_{t_i} to F_{t_i}. Here, we describe the specific implementation details of the different architectures.

MIP-GPM is a simple reference model using classic feature extraction with a Gaussian process regression model (GPM) [396]. We extract the needle tip's high-intensity surface using 1D maximum MIP on each median-filtered A-scan A_{t_i}. The normalized pixel index of the MIP represents a simple feature that captures deformation. This scalar feature serves as a comparison to the feature learning approach of our deep learning models.

RN1D processes A-Scans A_{t_i} individually without considering a history of data that resembles a single-shot learning approach. The model is a ResNet-based spatial 1D CNN, as shown in Table 4.2. The network's first layer is a convolution with kernel size 3 and a spatial stride of 2 and 64 feature maps. Then, three processing stages follow with 3, 4, and 5 ResBlocks each. The first convolution in each ResBlock uses a spatial stride of 2 and increases the number of feature maps by a factor of 2. Each ResBlock is built on the bottleneck principle [193], containing three consecutive convolutional layers. The first layer reduces the number of feature maps by a factor of 4 with a kernel of size 1. Then, a layer with kernel size 3 processes the tensor, followed by a convolution that upsamples the number of feature maps again to the original size. Each convolutional layer consists of a convolution operation, batch normalization, and a ReLU activation function. After the CNN's last layer, GAP is applied, and the final feature vector is processed by a fully-connected layer with one output that represents the force prediction.

GRU processes the set of A-Scans M_{t_i}, without taking spatial structure into account as it consists of three GRU layers with standard matrix multiplications being performed inside the gates. The model follows the standard GRU structure, as introduced in Sec-

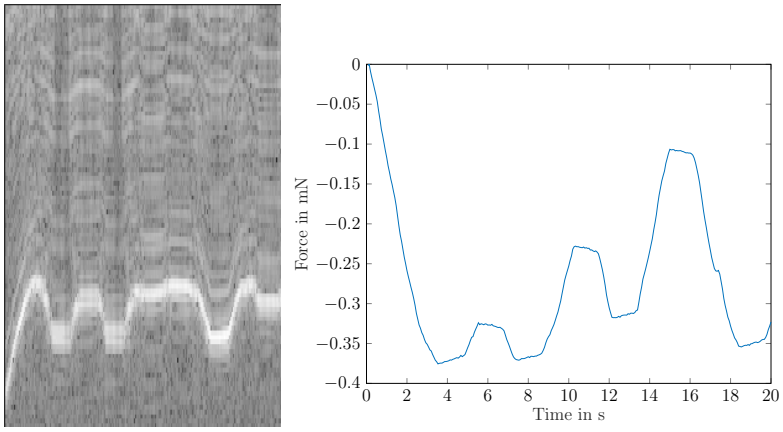


Fig. 6.4: We show cropped A-Scans and corresponding ground-truth forces over time for a training sequence.

tion 3.3.3. The first layer uses 128 feature maps, which is doubled in each following layer. We use recurrent batch normalization and recurrent dropout for additional regularization at the cell input with probability $p_{di} = 0.1$, and at the cell output with probability $p_{do} = 0.2$. After the GRU layers, GAP and the output layer, similar to RN1D, are applied.

RN2D is a 2D version of the RN1D model, processing the full spatio-temporal tensor M_{t_i}. The model is an implementation of a ResNet-based 2D CNN, see Table 4.2. All convolutional layers use 2D convolutions instead of 1D convolutions with isotropically extended kernels, compared to RN1D. Spatial and temporal downsampling is both performed with a stride of 2.

RN1D-GRU performs 2D spatio-temporal processing with spatial processing, followed by temporal processing. The model is an implementation of the CNN-GRU architecture we introduced in Section 4.3. The CNN part is the same ResNet-based model as described for RN1D. After the CNN's last layer and GAP layer, two GRU layers with 1028 feature maps sequentially process the temporal sequence of feature vectors. Finally, the output layer is applied to the GRU's last temporal output. The GRU layers also make use of recurrent batch normalization and recurrent dropout.

RN1D-cGRU is a ResNet-based implementation of CNN-cGRU, which we described in Section 4.3. We also use 1024 feature maps for the convolutional GRU units. Besides, the model is identical to the RN1D-GRU, except that the GAP layer is applied after the convolutional GRU unit. Thus, the convolutional GRU unit still processes a spatial representation instead of a feature vector.

cGRU-RN1D is a realization of the cGRU-CNN architecture we proposed in Section 4.3 that also processes the 2D spatio-temporal sequence M_{t_i}. We employ two convolutional GRU units with 64 feature maps each. Each convolutional GRU layer also uses recurrent dropout and recurrent batch normalization. The RN1D CNN is the same as for the RN1D model.

GRU-RN1D is a variant of cGRU-CNN with regular GRU units. Here, the A-Scans are directly treated as feature vectors. We use the same number of feature maps for the GRU units. The rest of the model is identical to cGRU-RN1D.

Tab. 6.1: Comparison of several architectures.

	MAE (mN)	rMAE (10^{-3})	PCC (%)	IT (ms)
cGRU-RN1D	$\mathbf{1.59 \pm 1.3}$	$\mathbf{19.9 \pm 17}$	**99.97**	10.3 ± 1.5
GRU	3.02 ± 3.7	37.7 ± 48	99.82	$\mathbf{2.5 \pm 0.4}$
RN1D	3.26 ± 3.9	39.3 ± 48	99.80	6.9 ± 1.3
RN1D-GRU	2.01 ± 3.2	24.7 ± 41	99.89	9.5 ± 1.4
RN1D-cGRU	2.03 ± 3.3	24.2 ± 43	99.90	11.1 ± 1.7
RN2D	2.11 ± 3.5	25.5 ± 44	99.87	8.6 ± 1.5
GRU-RN1D	11.79 ± 8.6	125 ± 100	99.48	10.1 ± 1.4
MIP-GPM	45.38 ± 38.7	482 ± 411	77.67	14.8 ± 2.2

Training and Evaluation. All networks are trained end-to-end. We use the Adam algorithm for optimization with a batch size of $N_b = 100$. Our implementation uses Tensorflow [1]. The initial learning rate is $\alpha_{lr} = 10^{-4}$. We halve the learning rate every 30 epochs and stop training after $N_e = 300$ epochs. We scale the target force values to a range of $[0, 1]$ for training and rescale them to their original range for metric calculation.

In terms of metrics, we report the MAE in mN with standard deviation, the rMAE with standard deviation, and PCC between predictions and targets. We test for a significant difference in the median of the models' absolute errors with the Wilcoxon signed-rank test and a significance level of $\alpha = 5\,\%$. Furthermore, we provide the inference time (IT) in ms of each model for a single forward pass, averaged over 100 repetitions.

Results

First, we compare our proposed cGRU-RN1D model to other spatio-temporal deep learning methods. The results are shown in Table 6.1. Overall, the models that perform spatio-temporal processing (cGRU-RN1D, RN1D-GRU, RN1D-cGRU, RN2D) clearly outperform RN1D and GRU. Overall, cGRU-RN1D performs best. Boxplots in Figure 6.5 show a more detailed analysis of the spatio-temporal deep learning models. The null hypothesis of an equal median for the absolute errors of cGRU-RN1D when compared to cGRU-RN1D, RN1D-GRU, RN1D-cGRU, and RN2D.

In terms of inference time, the spatio-temporal deep learning models can provide predictions with approximately $100\,\mathrm{Hz}$. The fastest spatio-temporal deep learning model is 2DCNN with an IT of $8.6\,\mathrm{ms}$, and the overall fastest model is GRU with an IT of $2.5\,\mathrm{ms}$. Note that these values are highly hardware (NVIDIA GTX 1080 Ti) and software (Tensorflow) dependent.

The previous results showed a clear performance increase for joint spatio-temporal processing. Therefore, we perform experiments to analyze the effect of the temporal dimension. In Figure 6.6, we show results for different n_t and the associated training durations with our cGRU-RN1D and RN1D-GRU model. Increasing n_t leads to improved performance with a lower MAE for both models. With increasing n_t, the overall training time also increases substantially. Across all values for n_t, the training time of cGRU-RN1D is lower than the time for RN1D-GRU.

Last, we present results for the needle insertion experiments shown in Figure 6.7.

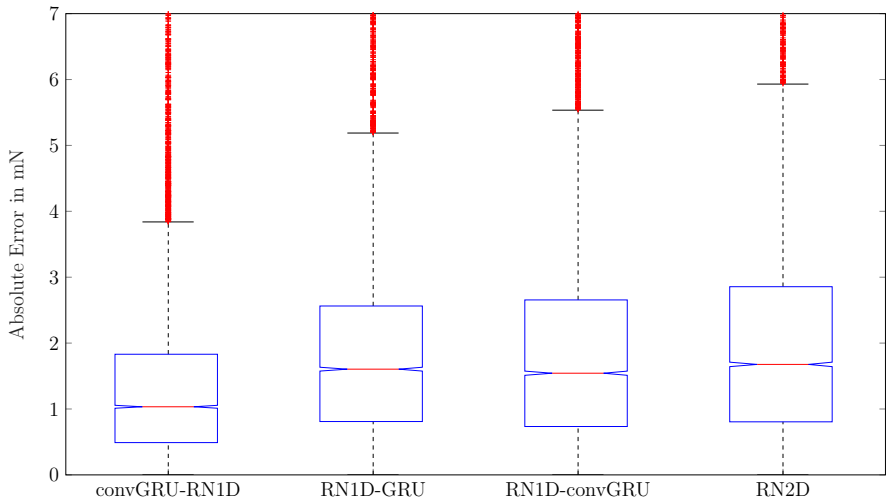


Fig. 6.5: Boxplots of the absolute errors for the top-performing spatio-temporal deep learning models. The red line marks the median, the boxes' bottom and top line mark the 25th and 75th percentiles, respectively. Red marks above the whiskers represent outliers. Notches around the median mark comparison intervals where non-overlapping intervals between boxplots indicate different medians at $5\,\%$ significance level. With respect to the frequency of outliers, consider that the test set contains 9000 examples.

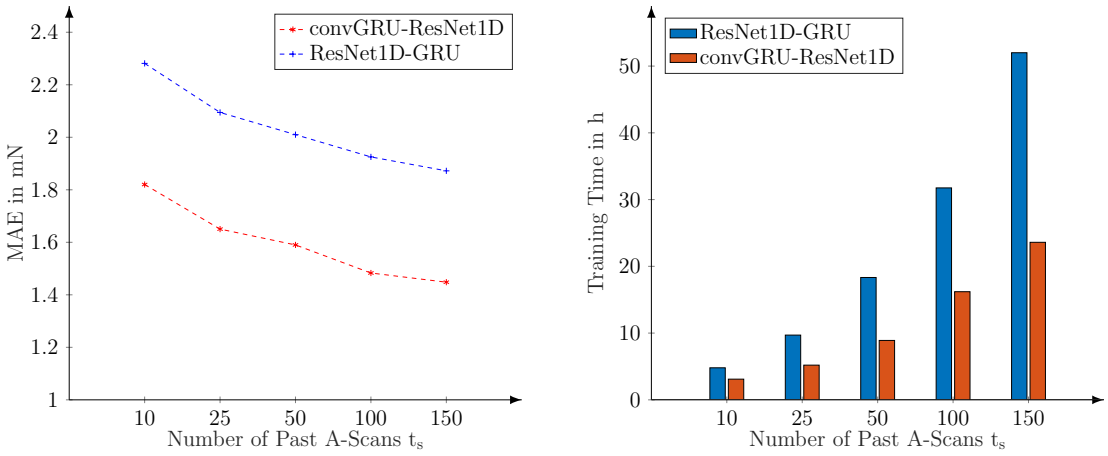


Fig. 6.6: Comparison between cGRU-RN1D and RN1D-GRU for different numbers of timesteps n_t.

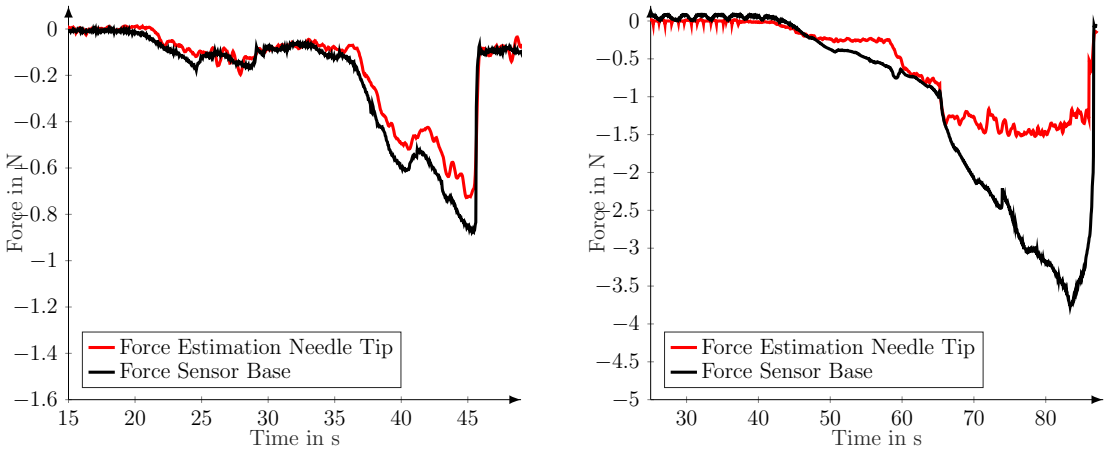


Fig. 6.7: Predicted and measured force values are shown for insertion with the shielding tube (left) and without (right). For the case without tube, differences between needle tip force estimation and force sensor are caused by friction. cGRU-RN1D was used for this experiment.

We performed one experiment with the shielding tube and without. When using the decoupled tube, the force sensor's measurements for ground-truth annotation closely match the values predicted by our model. Without the tube, friction forces induce a large difference between measurements and predictions.

Discussion

We present a new technique for needle tip force estimation using an OCT fiber embedded into a needle that images the deformation of an epoxy layer. The OCT fiber within the needle produces a series of A-Scans that can be treated as spatio-temporal data that consists of 1D images over time. To process this type of data, we employ our novel cGRU-RN1D architecture. The model performs both temporal and spatial processing and outperforms the pure temporal GRU and pure spatial RN1D with an MAE of $1.59 \pm 1.3\text{mN}$ compared to an MAE of $3.02 \pm 3.7\text{mN}$ and $3.26 \pm 3.9\text{mN}$, respectively. Also, we compared to the spatio-temporal models RN1D-GRU, RN1D-cGRU and RN2D, which are variants adopted from the natural image domain [32, 108, 556]. The three models are closer in terms of performance, but overall, cGRU-RN1D performs best. Notably, the differences in the median of the errors are significant, which is also highlighted by the boxplots showing the test set error distribution in Figure 6.5.

The key difference between all spatio-temporal deep learning models we compare is that cGRU-RN1D and GRU-RN1D first perform temporal processing, then spatial processing, RN1D-GRU and RN1D-cGRU first performs spatial, then temporal processing and RN2D performs concurrent processing. Overall, our proposed cGRU-RN1D model significantly outperforms all other variants. The lower performance of the previous spatio-temporal models ResNet1D-GRU [108] and RN1D-cGRU [32] indicates that temporal processing followed by spatial processing is preferable for the problem at hand. To highlight the necessity of convGRU units, we consider GRU-RN1D without convolutional gates. The MAE is significantly higher, which demonstrates the necessity

to preserve the spatial structure during temporal processing. In addition, we show that recurrent dropout and recurrent batch normalization can improve the spatio-temporal models' performance further. For reference, MIP-GPM shows that conventional feature extraction without extensive engineering cannot match deep learning models' performance for this problem.

Furthermore, we perform a more detailed analysis of our cGRU-RN1D model compared to the more common RN1D-GRU model. The results in Figure 6.6 show a decrease of the MAE when a longer history of A-Scans is considered. This highlights the value of exploiting temporal information for force estimation. However, this improvement is bought with a substantial increase in training time as the computational effort increases. For example, for cGRU-RN1D, using $n_t = 100$ instead of $n_t = 50$ previous measurements leads to a performance increase of $7\,\%$ and an increase in training time of $82\,\%$. Training time is an important aspect to consider for application as newly designed needles will require an initial calibration in terms of model training. Overall, both models benefit similarly from the additional temporal information, however, cGRU-RN1D trains faster. This is due to the convolutional GRU units, which have significantly fewer parameters than their GRU counterpart in RN1D-GRU.

Besides performance and training time, the models' inference time is important for application and real-time feedback of forces. Overall, the high-performing spatio-temporal deep learning models can process samples at $100\,\mathrm{Hz}$, which indicates real-time capability. Notably, RN2D is almost as fast as RN1D due to the fact that 2D convolutions are much more common and highly optimized in Tensorflow and CUDA. Thus, our cGRU-RN1D model's inference time could improve further with software optimization as a RN1D is also part of the model. Also, note that inference times are hardly affected by the number of previous measurements n_t. After initial processing, the recurrent part of the models stores previous information in its cells' states and only requires one additional sample to be processed at each step.

Last, we validated our needle design in a realistic insertion experiment with a freshly resected human prostate. Our results in Figure 6.7 show that the needle tip forces closely match the actual, decoupled, base measurements. While the decoupling is not perfect, we can show that our method accurately captures events such as ruptures. Without tip measurements or decoupling, large friction forces overshadow the actual tip forces. Overall, the experiments show that our method is usable for actual force estimation in soft tissue.

Summary

We investigate the use of 1D and 2D spatio-temporal data for the problem of needle-based force estimation. We build a needle design with a single OCT fiber that images an epoxy layer close to the needle's tip, capturing deformation that acts on the tip. The OCT signal is a time series of 1D depth images. For processing this type of data, we consider 1D CNNs, 2D CNNs, and various recurrent-convolutional models, including our novel cGRU-CNN concept. In the context of our two research questions, we find that using 2D data improves force estimation performance over 1D data processing. Also, our proposed cGRU-CNN concept performs best among multiple different recurrent and convolutional methods while maintaining reasonable inference and training times.

6.2 1D and 2D Data: OCT-Based Retina Segmentation

Beside vision-based force estimation, low-dimensional OCT is also relevant for diagnostic problems in ophthalmology, as introduced in Section 5.2. The extent of retina layers is important for assessing diseases such as AMD or DR and can be captured by OCT images. In principle, retina layers can be distinguished in individual 1D A-Scans or in 2D B-Scans where the latter potentially offers more context for more consistent boundary estimates. Thus, OCT-based retina segmentation can be addressed as a 1D or 2D spatial data processing problem.

In contrast to fiber-based force estimation, retina layer segmentation comes with established deep learning architectures, largely based on the U-Net principle. Thus, an important question is how these encoder-decoder architectures can be improved further. In recent research, one way to address this problem is the design of new processing blocks. Here, the macro-scale encoder-decoder structure is kept the same, while the convolutional layers are replaced by a more advanced principle such as ResNet blocks [575]. For the problem of retina layer segmentation, we take this approach one step further by employing our proposed ENAS framework for automatically learning the microarchitecture building blocks of encoder-decoder CNNs. This is one of the first applications of NAS frameworks in the medical image domain. Retina layer segmentation is particularly well suited for this exploration of NAS as we can try to exploit lower-dimensional, 1D data representations for learning architectures that potentially work well with 2D data.

As a result, concerning our two research questions, we study whether using 2D spatial data is advantageous over 1D spatial data. Also, we investigate whether using NAS allows for automatically finding better architectures in an efficient way. We explore whether architectures found with lower-dimensional data can be transferred to higher-dimensional data.

Methods and Datasets

Problem Definition. Our goal is to segment the different retina layers in OCT images. Therefore, we address a 1D and 2D spatial learning problem where we map either A-Scans $A_i \in \mathbb{R}^{n_d}$ or B-Scans $B_i \in \mathbb{R}^{n_a \times n_d}$ to segmented maps $\mathrm{y}_i \in \mathbb{R}^{n_d}$ or $\mathrm{y}_i \in \mathbb{R}^{n_a \times n_d}$. Thus, we try to find deep learning models $f_M : \mathbb{R}^{n_a \times n_d} \rightarrow \mathbb{R}^{n_a \times n_d}$. A B-Scan consists of n_a individual A-Scans.

Dataset. We use a publicly available OCT dataset with images from patients with mild AMD and normal subjects [134]. The mean age of healthy patients was 66.6 years and 74.6 years for patients with mild AMD. In total, we used images from 268 patients with 115 healthy subjects 153 subjects with mild AMD. All images were acquired with an SD-OCT device (Bioptigen, Research Triangle Park, NC). The total FOV covers $6.7\,\mathrm{mm} \times 6.7\,\mathrm{mm}$. For each patient, 100 B-Scans are acquired where each B-Scan B_i consists of 1000 A-Scans $A_i \in \mathbb{R}^{512}$. As not all scans are labeled entirely, we take a crop from each patient's scan such that the crop contains labeled regions only. This results in varying numbers of A-Scans per B-Scan and a varying number of B-Scans per patient.

Experts provided layer boundaries for the inner limiting membrane (ILM), retinal pigment epithelium drusen complex (RPEDC), and Bruch's membrane (BM). We generate

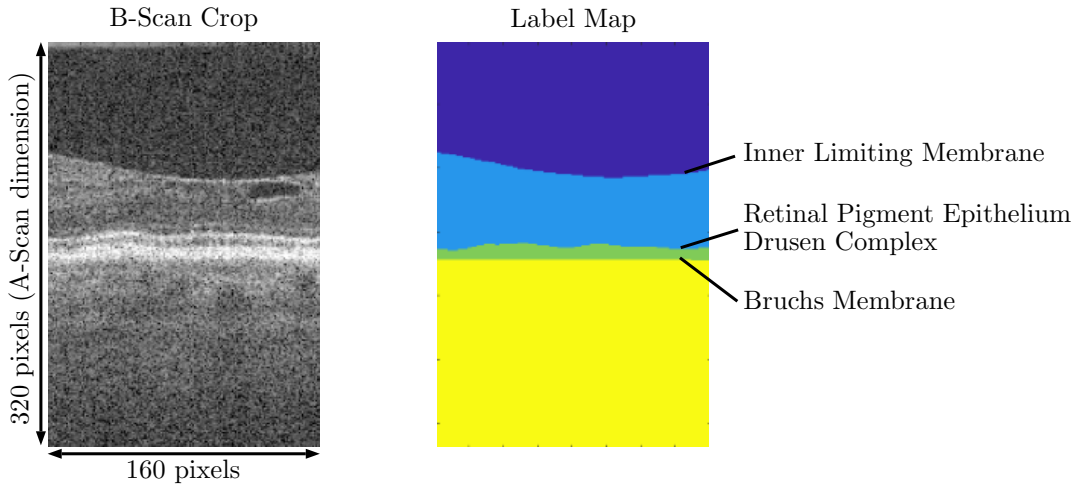


Fig. 6.8: Example images and labels for retina layer segmentation.

pixel-wise annotations by assigning classes to tissue layers in between boundaries. Thus, ILM to RPEDC is class 1, RPEDC to BM is class 2, and BM to the end is class 3. The image space above the ILM is treated as background. Note that directly learning the boundaries can be beneficial for this problem [416]. We chose a pixel-wise encoding to have a representative medical segmentation task that can be addressed with a standard U-Net. Example cropped image data and ground-truth labels are shown in Figure 6.8.

Deep Learning Models. As described in Section 4.1.1, we start with a baseline U-Net architecture for segmentation. The model consists of four processing stages with different spatial resolutions. At the start of each stage, the spatial resolution is reduced by half with a stride of 2 and the number of feature maps is doubled. Each stage consists of two module blocks, except for the first and last stages. The initial feature map size is $n_c = 32$. For the baseline model, we use standard ResNet blocks with two consecutive convolution layers, batch normalization, ReLU activation functions, and a skip connection. Similar to U-Net, we make use of long-range connections between encoder and decoder. In contrast to U-Net, we use feature summation instead of concatenation. We employ this model both for the segmentation of 1D A-Scans (RN1D-U) and 2D B-Scans (RN2D-U). We use 1D convolutional layers and 2D convolutional layers, respectively. All other operations remain the same.

Next, we employ our proposed ENAS U-Net concept that we introduced in Section 4.1.1. Here, we try to learn the microstructure of the baseline architecture described above in terms of alternatives for the ResNet blocks. As described in Section 4.1.1, we set up the search space of learnable architecture hyperparameters $h_M^A \subset h_M$ similar to the original description given by Pham et al. [377] with connected cells. Both the operations within the cells and the connections are learnable. To keep the overall search space small, we consider either $N_{Cells} = 2$ or $N_{Cells} = 3$ cells within each block. Also, we consider the options to either learn one type of block for the entire model or one type of block for the encoder and one type of block for the decoder. For ENAS training, all possible operations are defined in each cell and maintained throughout the training process. If the controller defines a different architecture, the set of connections within the cell are rerouted. Operations such as convolutional layers remain in the architecture

with their current weights, even if they are disconnected from other operations for a particular setting defined by the controller.

The controller is a single-layer LSTM with 64 feature maps. Attention-based highway connections [600] augment the recurrent connections and produce a combination of the previous LSTM cell state and a new, transformed state. The controller's output layer is a softmax layer that represents the probability of the current cell's input or operation. A specific connection or operation is sampled by treating the softmax vector as a multinomial probability distribution.

Training and Evaluation. We consider a training set $\mathcal{X}_{train}$ of 150 volumes (model training), a reward set $\mathcal{X}_{reward}$ of 56 volumes (controller training), a validation set $\mathcal{X}_{val}$ of 2 volumes, and a test set $\mathcal{X}_{test}$ of 60 volumes. We follow ENAS with interleaved training of the model and the controller. After training for $N_e = 200$ epochs, we sample $N_{sample} = 20$ architecture configurations from the controller and evaluate them on the validation set. Then, we select the best-performing configuration and retrain the model from scratch on the training set. Finally, we evaluate the model's performance on the test set. For the baseline model, we train on the training set for $N_e = 200$ epochs and evaluate on the test set afterward.

All models are trained using the Adam algorithm with the dice loss function given in Equation 3.50. During training, we take random crops of size $A_i \in \mathbb{R}^{320}$ or $B_i \in \mathbb{R}^{320\times160}$ for 1D and 2D models respectively. This serves the purposes of inducing more variation during training and adding a regularizing effect. Furthermore, we perform data augmentation by randomly flipping B-Scans along the B-Scan dimension. For evaluation and metric calculation, we first obtain predictions for every B-Scan for each patient. Along the A-Scan dimension, we take two overlapping crops whose predictions are averaged in the overlapping region. For 1D models, we obtain a prediction for each A-Scan in every B-Scan. For 2D models, we take five overlapping crops across the B-Scan dimensions where predictions are averaged in the overlapping regions. This is an implementation of the multi-crop evaluation strategy described in Section 3.3.5. Finally, we calculate the metrics for each patient in the evaluation set. Then, the metric values for all patients are averaged into our final evaluation metric. As metrics, we consider the dice coefficient, pixel-wise accuracy, sensitivity, and specificity.

The controller is also trained with the Adam algorithm. Its loss function consists of the predicted dice metric that was obtained for the reward set, as described above. Also, the controller's current sample entropy is added to the reward, weighted by a factor of 10^{-4}. To prevent early convergence, a temperature of 5.0 and a $\tanh$ constant of 2.5 is used for the logits before applying the softmax, see Bello et al. [39]. A moving average baseline of the reward is used to reduce variance during controller training [377].

Results

Quantitative results for the comparison of 1D ENAS and 2D ENAS are shown in 6.2. Comparing the models processing 1D and 2D data, the use of higher-dimensional data leads to a substantial increase in performance. When considering the models learned with ENAS, both the 1D and 2D architectures learned with ENAS on 1D data outperform the ResNet baseline in terms of all metrics. For the 2D case, when also performing the ENAS search on 2D data, performance improves marginally. In terms of search time,

Tab. 6.2: We show the dice score, sensitivity (Sens.), specificity (Spec.), given in percent for the different models. For the ENAS-based models, we present search time for a search performed on an NVIDIA GTX 1080 Ti. We use $N_{Cells} = 2$ and shared blocks for encoder and decoder for this experiment.

	Dice	Sens.	Spec.	Search Time
RN1D-U	91.6 ± 10	91.9 ± 4	96.4 ± 6	–
ENAS1D-U h_M^{1D}	$\mathbf{92.8 \pm 9}$	$\mathbf{93.6 \pm 6}$	$\mathbf{97.8 \pm 4}$	1 h 30 min
RN2D-U	96.2 ± 5	96.7 ± 7	98.2 ± 4	–
ENAS2D-U h_M^{1D}	97.1 ± 4	$\mathbf{97.5 \pm 6}$	97.9 ± 3	1 h 30 min
ENAS2D-U h_M^{1D}	$\mathbf{97.3 \pm 4}$	97.3 ± 6	$\mathbf{98.4 \pm 5}$	12 h 0 min

Tab. 6.3: We show the dice score, sensitivity (Sens.), specificity (Spec.), given in percent for different ENAS configurations. We compare different numbers of cells N_{Cells} and shared blocks for encoder and decoder (S) as well as individual, unshared blocks (NS).

	Config	Dice	Sens.	Spec.
ENAS1D-U h_M^{1D}	$N_{Cells} = 2\ S$	92.8 ± 9	93.6 ± 6	97.8 ± 4
ENAS1D-U h_M^{1D}	$N_{Cells} = 3\ S$	92.6 ± 8	92.9 ± 5	98.3 ± 4
ENAS1D-U h_M^{1D}	$N_{Cells} = 2\ NS$	93.1 ± 9	93.8 ± 5	98.0 ± 2
ENAS1D-U h_M^{1D}	$N_{Cells} = 3\ NS$	$\mathbf{93.2 \pm 7}$	$\mathbf{94.1 \pm 7}$	$\mathbf{98.3 \pm 3}$
ENAS2D-U h_M^{1D}	$N_{Cells} = 2\ S$	97.1 ± 4	97.5 ± 6	97.9 ± 3
ENAS2D-U h_M^{1D}	$N_{Cells} = 3\ S$	97.3 ± 4	97.6 ± 7	97.5 ± 4
ENAS2D-U h_M^{1D}	$N_{Cells} = 2\ NS$	$\mathbf{97.4 \pm 3}$	97.9 ± 5	$\mathbf{98.2 \pm 4}$
ENAS2D-U h_M^{1D}	$N_{Cells} = 3\ NS$	$\mathbf{97.4 \pm 4}$	$\mathbf{98.4 \pm 4}$	98.1 ± 3

the search on 1D data is substantially shorter than searching on 2D data.

Next, we consider ablation experiments where we study the properties of ENAS U-Net in more detail. The results are shown in Table 6.3. We compare the use of 2 cells and 3 cells for each block as well as individual block types for the encoder and decoder, respectively. We can observe that there is a minor improvement if we increase the search space by allowing different blocks for encoder and decoder as well as allowing 3 instead of 2 cells per block.

Finally, we consider qualitative results in terms of the learned model blocks, see Figure 6.9. We can observe that different types of blocks emerge to be the highest-performing configuration for the different search configurations. Notably, there is a lot of variability in terms of the operations chosen by the controller and only a few repetitions.

Discussion

We propose ENAS U-Net, an adaptation of the ENAS framework [377] for U-Net-like architectures. In particular, we try to reduce search time by performing architecture search on lower-dimensional data representations before applying the learned architectures to higher-dimensional data. This follows the hypothesis that architectures are, to some degree, transferable across dimensions.

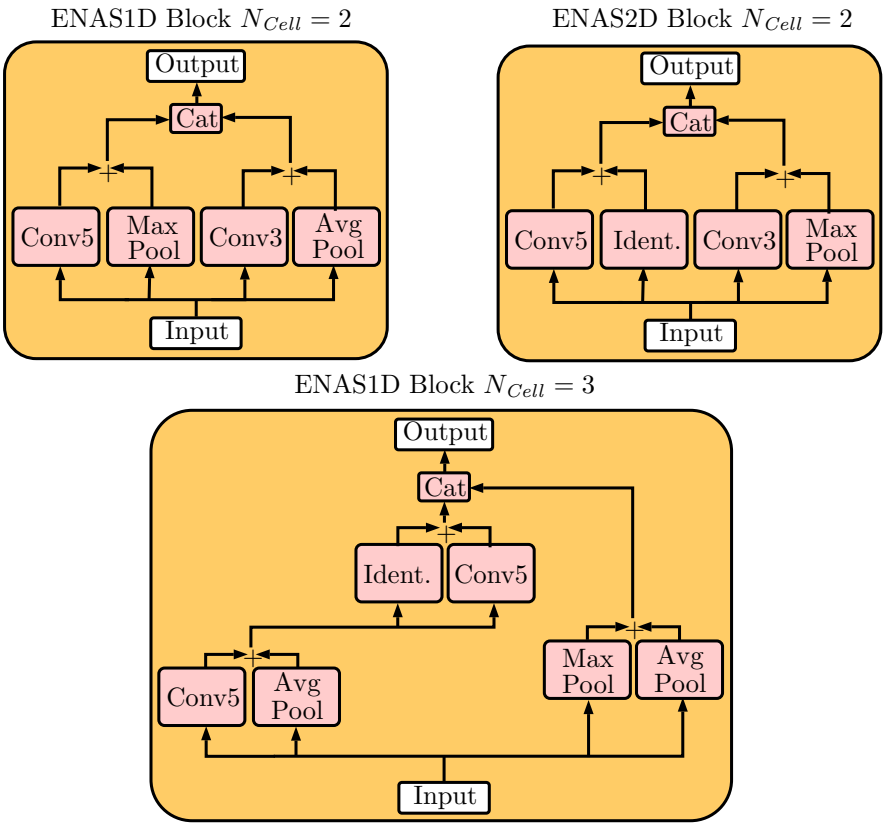


Fig. 6.9: Example blocks that were learned by ENAS U-Net.

Considering the performance difference between A-Scan-based 1D and B-Scan-based 2D processing with U-Net architectures, we observe a substantial performance increase. This suggests that the additional context is helpful for learning accurate retina layer segmentation.

When comparing architectures that were derived from ENAS to the standard ResNet, there is also a performance increase across multiple metrics. This is in line with previous research on NAS, where better architectures could be found with reinforcement-based search strategies [39, 377, 603]. Notably, the increase is achieved without altering fundamental, and potentially more impactful U-Net properties such as the encoder-decoder structure or the long-range connections. As a next step, these properties could be included in the search space, which was successful for segmentation in the natural image domain with DeepLab-based architectures [302].

Performing a search on 1D data substantially decreases the search time by $87.5\,\%$ compared to a search on 2D data while performance differences are marginal. This is particularly interesting as the OCT data is not isotropic, and the spatial dimensions are quite different. This indicates that learning on low-dimensional, less resource-demanding data representations is a viable approach for NAS. Thus, an extension to other problems such as brain segmentation might be feasible, for example, by performing NAS on axial slices before applying the discovered architectures on 3D volume data.

Also, we consider several variations of the ENAS search space, see Table 6.3. While there are increases in performance for larger search spaces with more learnable cells or different blocks for the encoder and decoder, the overall performance gain is limited. This indicates that more substantial changes to the search space might be necessary, for example, including the macro-level architecture in terms of the different processing stages or long-range connections.

Summary

Summarized, we propose an efficient approach for NAS that is suitable for multi-dimensional medical image data. In detail, we transform the ENAS framework for applications with U-Net-like architectures and evaluate the approach for retina layer segmentation. We demonstrate that searching for an architecture on low-dimensional 1D data transfers well to higher-dimensional 2D data. An architecture discovered on 1D data performs similar to one discovered on 2D data while substantially reducing search time. In terms of our two research questions, we find that using 2D data is advantageous over using 1D data for retina layer segmentation. Also, automatically designed architectures for 1D data transfer well to 2D data.

6.3 2D Data: OCT-Based Intravascular Tissue Classification

The second major clinical application of OCT that we introduced in Section 5.2 is intravascular tissue classification. Here, the goal is to detect and characterize plaque deposits within the arterial walls of patients' coronary arteries. IVOCT images are acquired very differently compared to ophthalmology. As described in Section 2.1.1, a rotating probe continuously acquires A-Scans that can be mapped into different image data representations. Usually, A-Scans from a 360° rotation are mapped into a 2D B-Scan. By directly stacking the A-Scans, we obtain a polar B-Scan representation $B_i^P \in \mathbb{R}^{n_d \times n_\theta}$ where n_d is the size of the A-Scans' depth dimension, and n_θ is the number of rotation angles. Using a circular transform, we can obtain the Cartesian representation $B_i^C \in \mathbb{R}^{n_h \times n_w}$. While clinicians prefer the Cartesian representation, it is largely unclear which data representation is preferable for deep learning-based image processing, as outlined in Section 5.2. Also, for image classification tasks with 2D spatial image data, transfer learning from classifications tasks from the natural image domain is a viable approach. Here, a major open question is whether learning from the different 2D IVOCT data representations can also be improved by transfer learning and which type of transfer strategy is optimal for that.

Therefore, we provide a thorough investigation of IVOCT-based deep learning using a newly built dataset. First, we study the use of polar and cartesian image representations for deep feature learning. In particular, we use data augmentation techniques tailored to each representation. To improve model performance, we employ transfer learning from the ImageNet dataset using the established models ResNet and DenseNet. As natural images are vastly different from IVOCT data, we provide an in-depth analysis of transfer learning, showing performance for different transfer strategies. Additionally, we explore a type of multi-dimensional transfer learning, where we build a new two-path architecture out of pretrained parts from other models that leverage features from both image representations. Overall, we consider binary labels "plaque" and "no plaque", as well as a further differentiation of "calcified plaque" from "fibrous/lipid plaque". This choice is also clinically motivated as calcified plaque deposits often require different tools for treatment, such as rotablation.

Summarized, in terms of our two research questions, we investigate which 2D spatial data representation is advantageous for deep learning-based IVOCT data processing. Furthermore, we study different types of architecture concepts, transfer learning strategies, and our new two-path model, enabled by multi-dimensional transfer learning, for this problem.

Methods and Datasets

Problem Definition. We address the problem of detecting and characterizing plaque deposits in IVOCT images. Based on 1D A-Scans $A_i \in \mathbb{R}^{n_d}$ we consider both polar 2D B-Scans $B_i^P \in \mathbb{R}^{n_d \times n_\theta}$ and Cartesian 2D B-Scans $B_i^C \in \mathbb{R}^{n_h \times n_w}$. The targets $\mathrm{y}_i \in \mathbb{R}^{N_c}$ consist of N_c classes. Thus, we try to find deep learning models $f_M : \mathbb{R}^{n_d \times n_\theta} \rightarrow \mathbb{R}^{N_c}$ and $f_M : \mathbb{R}^{n_h \times n_w} \rightarrow \mathbb{R}^{N_c}$.

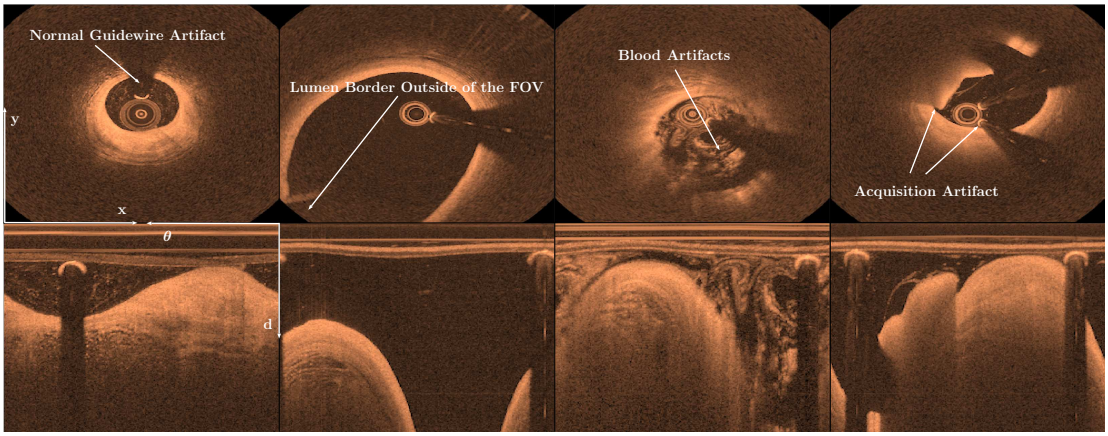


Fig. 6.10: Example IVOCT images, highlighting the problematic nature of the modality. The bottom row shows polar representations, the upper row shows the corresponding transformed Cartesian images.

Dataset. We build a new dataset based on clinical IVOCT images that were acquired with a St. Jude Medical Ilumien OPTIS. The device is connected to a Dragonfly OPTIS imaging catheter, which is inserted into the patient's coronary vessels. Inside the vessel, images are acquired by rotating the OCT probe inside the catheter and pulling it backward. In this way, a continuous M-Scan consisting of 1D A-Scans is created. The A-Scans from each $360°$ turn are arranged in a B-Scan, which is the polar representation shown in Figure 6.10. The polar image $B_i^p \in \mathbb{R}^{n_d \times n_\theta}$ can be transformed into Cartesian space with the coordinate transformation $x = d\cos(\theta)$ and $y = d\sin(\theta)$, where x and y are Cartesian coordinates, and d and θ are polar coordinates. Applying interpolation in between the transformed pixels results in the Cartesian representation $B_i^C \in \mathbb{R}^{n_h \times n_w}$, which resembles a cross-sectional view of the artery, see Figure 6.10.

For ground-truth annotation, three trained experts with daily experience in IVOCT-assisted interventions determine the type of plaque that is present in a B-Scan. The experts provide labels on the B-Scan level, and they assign the labels "no plaque", "calcified plaque", or "lipid/fibrous plaque" to each B-Scan presented to them. All experts were partially provided the same images and different images for labeling. For the same images, the final label is determined based on a consensus between the experts. Different opinions on the plaque type were resolved by asking the experts for a repetition of their evaluation. Providing different images to experts allows for a larger but potentially noisily labeled dataset. We sample our test set from the consensus set to ensure a meaningful evaluation. Before resolving conflicting evaluations, we found an agreement of $87\,\%$ for binary classification and $68\,\%$ for multi-class classification.

In total, the dataset consists of 4000 images from 49 patients. We split off an independent test set of 742 images from 9 patients. We report our results based on this test set. We perform three-fold cross-validation on the training set in order to select all the relevant hyperparameters mentioned above. We use the ground-truth labels in two variants. For general results on plaque detection, we use the binary labels "plaque" and "no plaque". We refer to this dataset as the binary dataset. Furthermore, we consider a multi-type plaque dataset where we divide the class "plaque" further into "calcified

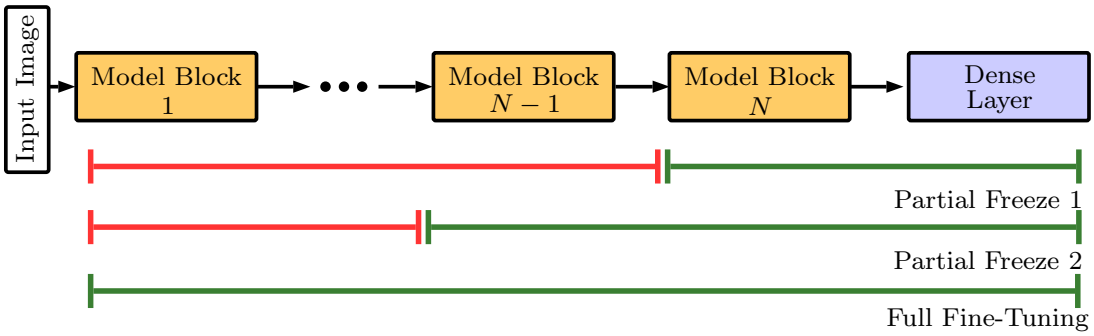


Fig. 6.11: The transfer learning strategies we employ are shown. We either freeze a part of the weights that were pretrained on ImageNet, or we fine-tune the entire network.

plaque" and "lipid/fibrous plaque". This dataset serves the purpose of showing the feasibility of further plaque differentiation with clinical relevance.

For the binary dataset, plaque and non-plaque images are approximately balanced. For the multi-type dataset, roughly $10\,\%$ are labeled as "calcified plaque", $40\,\%$ are labeled as "fibrous/lipid plaque" and $50\,\%$ are labeled as "no plaque". None of the images contains a stent as this would distort results. Our goal is to detect untreated plaque deposits, and a deep learning model might learn to detect stents instead of the underlying disease.

Preprocessing and Data Augmentation. The original polar images obtained from the OCT device have a resolution of $B_i^P \in \mathbb{R}^{496\times960}$ pixels. Thus, the transformed cartesian images have a resolution of $B_i^C \in \mathbb{R}^{1920\times1920}$ pixels. For training, we resize both to a resolution of $B_i \in \mathbb{R}^{300\times300}$, which matches the range used with standard architectures in the natural image domain [193, 208, 262]. Note, that this leads to the Cartesian images having half the depth resolution of the polar images.

For data augmentation, an intuitive choice is to use random rotations for the Cartesian images due to their circular structure. We apply random rotations with $\alpha_{rot} \in [0°, 360°]$ to Cartesian images. Moreover, we apply random flipping along the horizontal and vertical directions. For polar images, rotations are not meaningful. Instead, we shift the polar images randomly along the θ dimension with $s_\theta \in [0px, 300px]$. If A-Scans are shifted out of the image along the positive θ direction, they are added back on the other side. In this way, we achieve a transformation that matches a rotation in Cartesian space. Additionally, we apply random flipping along the θ dimensions. During training, for both image representations, we apply random cropping to an image size of $\mathrm{x} \in \mathbb{R}^{270\times270}$. For evaluation, we do not use any rotations or flipping, and we use a center crop.

Deep Learning Models. We employ two state-of-the-art architectures from the natural image domain, namely ResNet50V2 (RN2D-50) [193] and DenseNet121 (DN2D-121) [208]. The architecture concepts were introduced in Section 4. The specific models are standard versions of the architecture concept for multi-class image classification.

As our dataset is small compared to, for example, ImageNet [262], we make use of transfer learning. We consider different levels of weight freezing, both for the RN2D-50 and DN2D-121 model. The retraining points are indicated in Fig. 6.11. For example, for partial freeze 1 ($p_r = 1$), we freeze all weights on the left of $p_r = 1$ and retrain the

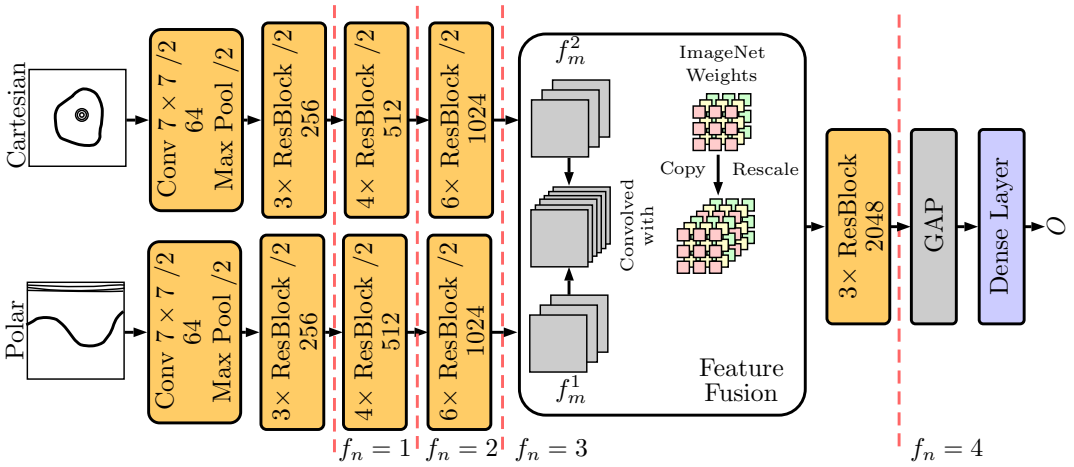


Fig. 6.12: The two-path architecture for simultaneous use of polar and Cartesian representations, shown for the RN2D-50 model. We initialize all weights with the pretrained values from ImageNet training. The point $f_n = i$ denotes feature fusion after the i^{th} ResNet block. Here, $f_n = 3$ is shown. For fusion, we employ our specialized weight adjustment strategy we introduced in Section 4.2.

rest. Thus, we consider two partial retraining scenarios, training from scratch, and full retraining for both models. In all cases, we remove the last fully-connected layer and replace it with a layer that has a matching number of outputs for our binary or multi-class labels.

Furthermore, we consider a fusion of polar and Cartesian images in a single architecture in order to investigate whether joint features from the two representations provide an overall benefit for the problem. For this purpose, our Siamese architecture concept introduced in Section 4.2 is employed. The implementation of this model based on ResNet (TP-RN2D-50) is shown in Figure 6.12. Similarly, we construct the architecture TP-DN2D-121. Each path receives the polar or Cartesian representation as input. We apply the same data augmentation techniques to each representation, as for individual models. The point f_n describes the concatenation point within the network, where the image features are fused.

For transfer learning with the Siamese models, we employ the initialization strategy described in Section 4.2. Here, pretrained weights are copied into the new convolutional layers after the concatenation point. Also, the weights are rescaled by a factor of 0.5 to keep feature magnitudes consistent after applying the operation. The implementation of this concept is also shown in Figure 6.12.

Training and Evaluation. For model training, we minimize the cross-entropy loss as given in Equation 3.43. For the multi-class cases, we weight the loss for "calcified plaque" higher in order to counter the class imbalance. For training, we use the Adam algorithm with a starting learning rate of $\alpha_{lr} = 10^{-4}$. To find the optimal schedule, we reduce the learning rate by a factor of two when the validation error saturates. We use a batch size of $N_b = 30$ for single-path models and $N_b = 20$ for two-path models. In total, we train each model for $N_e = 300$ epochs. We find relevant hyperparameters with a restricted grid search with a coarse grid using the F_1-score from three-fold cross-

Tab. 6.4: Results for binary plaque classification are shown. All models are pretrained on ImageNet, and all weights are fine-tuned. Metrics are given in percent.

		Acc.	Sens.	Spec.	F1-Score
Data Aug.	DN2D-121 Cart.	89.2	**87.4**	90.7	88.5
	DN2D-121 Polar.	87.1	85.2	88.3	86.5
	RN2D-50 Cart.	**90.3**	86.1	**93.7**	**88.8**
	RN2D-50 Polar	87.2	88.8	85.9	86.1
No Data Aug.	DN2D-121 Cart.	75.5	69.3	80.7	71.4
	DN2D-121 Polar.	82.1	80.2	83.1	81.8
	RN2D-50 Cart.	74.0	77.6	71.9	73.7
	RN2D-50 Polar	81.4	81.0	85.4	81.3

validation. These hyperparameters include the initial learning rate, dropout rate, input image size, and crop size. Relevant results are reported for the independent test set. We implement our models using Tensorflow [1].

For the evaluation of binary classification models, we report accuracy (Acc.), sensitivity (Sens.), specificity (Spec.), and the F1-score. For multi-class classification, we report the per-class weighted accuracy for each class and the F1-score with all classes. Note that in contrast to our other application scenarios, we do not provide a standard deviation as we evaluate on a single test set. The per-class weighted accuracy is calculated by dividing the number of true positive examples by the number of all examples for that particular class.

Experiment Overview. First, we present results for binary classification, both for the RN2D-50 and DN2D-121 model, as well as for polar and Cartesian images. We show results with and without our data augmentation strategy. All models are pretrained on ImageNet.

Second, we investigate transfer learning by showing results without pretraining, and two partial retraining scenarios, both for DN2D-121 and RN2D-50 using Cartesian images.

Third, we provide results for our two-path architecture, using both image representations. Results for TP-RN2D-50 and TP-DN2D-121 are presented. We show results for different fusion points f_n, investigating the optimal abstraction level for feature fusion. We present results for our weight initialization strategy at the fusion point.

Last, we consider multi-class classification for both CNN models. Again, we consider both image representations and the use of data augmentation. All models are pretrained on ImageNet.

Results

For binary plaque classification, results are shown in Table 6.4. The most evident difference is visible for data augmentation strategies. Also, performance improvement through data augmentation is notably higher for Cartesian images. Furthermore, we can observe that Cartesian images lead to better performance than polar images. The difference between the RN2D-50 and DN2D-121 model is relatively small. Additionally, we show qualitative results for plaque detection along a pullback. The visualization is

shown in Figure 6.13. Although some slices are misclassified, the method provides a good overall overview of the presence of plaque along the pullback.

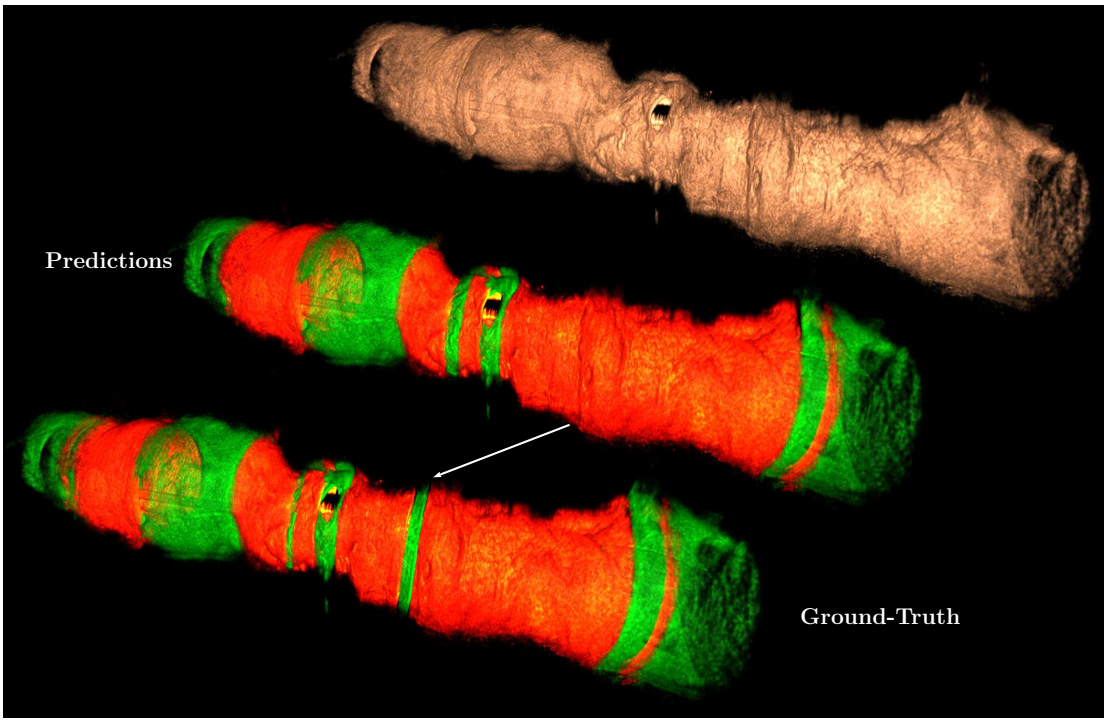


Fig. 6.13: We show qualitative results with a rendered pullback. Top, the original pullback is shown. Mid, predicted plaque regions are shown along the pullback. Bottom, the ground-truth labels are shown along the pullback. The color red indicates regions with plaque, and the color green indicates regions without plaque. The arrow indicates visibly incorrect predictions. Rendering was performed by stacking Cartesian slices along the catheter's center.

Furthermore, we investigate different transfer learning strategies. Table 6.5 shows the results for different levels of retraining. We consider training from scratch, full fine-tuning, and two different partial freezing scenarios. Overall, fine-tuning all weights within the network performs best. This is followed by partially freezing some weights. The more weights are frozen, the more performance deteriorates. Performance is worst when no pretraining is performed at all. We also report results for training from scratch with both Cartesian and polar images. The difference between full retraining and training from scratch is larger for Cartesian images.

Moreover, we consider the two-path model for the simultaneous processing of both data representations. The results for the approach are shown in Table 6.6. We show results for different fusion points and our weight initialization strategy at the fusion point. The model with an intermediate concatenation point $f_n = 3$ performs best, being close to $f_n = 4$. Moving the point closer to the input of the network deteriorates performance. Moreover, the proposed weight initialization strategy considerably improves performance.

Last, we show multi-class plaque classification results in Table 6.7. Overall, the classification is worse than for the binary classification case. Still, the classification of

Tab. 6.5: We show results for different transfer learning scenarios using Cartesian images unless indicated otherwise. Full refers to full fine-tuning of all weights. No Pre. refers to training with random weight initialization instead of pretraining. p_r denotes the point of weight freezing in the network. Metrics are given in percent.

		Acc.	Sens.	Spec.	F1-Score
DN2D-121	Full	89.2	**87.4**	90.7	88.5
	No Pre.	73.7	76.1	71.8	72.1
	$p_r = 1$	86.1	84.2	89.1	86.6
	$p_r = 2$	84.8	81.7	89.2	84.0
	No Pre. (Polar)	75.8	77.0	74.3	75.1
RN2D-50	**Full**	**90.3**	86.1	**93.7**	**88.8**
	No Pre.	75.8	73.3	77.0	71.6
	$p_r = 1$	88.2	87.0	89.3	86.8
	$p_r = 2$	85.1	78.5	89.4	82.3
	No Pre. (Polar)	77.4	76.5	80.0	75.7

Tab. 6.6: We show results for the two-path models TP-DN2D-121 and TP-RN2D-50. f_n denotes the fusion point in the network. No Init. refers to no use of our weight initialization strategy. All other weights are still initialized with their pretrained values. We demonstrate it with the best performing model with concatenation point $f_n = 3$. Metrics are given in percent.

		Acc.	Sens.	Spec.	F1-Score
TP-DN2D-121	No Init.	85.3	83.6	87.5	84.6
	$f_n = 2$	87.1	84.3	89.5	86.4
	$f_n = 3$	91.0	89.2	91.9	90.8
	$f_n = 4$	90.3	88.1	92.3	89.5
TP-RN2D-50	No Init.	87.1	85.0	88.3	86.2
	$f_n = 2$	86.7	87.1	85.2	85.6
	$\boldsymbol{f_n = 3}$	**91.7**	**90.9**	**92.4**	**91.3**
	$f_n = 4$	90.4	89.9	90.6	90.1

Tab. 6.7: We show results for multi-class plaque classification. The classes are "calcified plaque" (c_1), "fibrous/lipid plaque" (c_2) and "no plaque" (c_3). All models are pretrained on ImageNet and all weights are retrained. W.A. refers to the weighted per-class accuracy. Metrics are given in percent.

		W.A. c_1	W.A. c_2	W.A. c_3.	F1-Score
Data Aug.	**DN2D-121 Cart.**	78.0	**84.8**	**89.7**	**83.3**
	DN2D-121 Polar.	75.5	80.3	86.7	79.4
	RN2D-50 Cart.	**79.4**	82.2	87.3	82.9
	RN2D-50 Polar	76.2	79.9	85.6	80.5
No Data Aug.	DN2D-121 Cart.	64.6	70.2	75.5	70.8
	DN2D-121 Polar.	69.1	73.7	80.1	75.7
	RN2D-50 Cart.	63.0	68.9	75.3	69.4
	RN2D-50 Polar	68.7	69.1	77.0	73.3

calcified plaque achieves a slightly lower but similar performance as for the detection of other plaque types. The general trends in terms of performance are similar to binary classification. The use of Cartesian images improves performance slightly for both models. No use of data augmentation significantly reduces classification performance.

Discussion

We provide an extensive investigation of deep learning-based IVOCT data processing in the context of different data representations. This encompasses comparisons between using Cartesian and polar representations, representation-specific data augmentation, transfer learning strategies, different classification problems, and a novel approach for using both data representations simultaneously with a multi-dimensional transfer learning strategy.

Using our dataset, we first investigate the use of polar and Cartesian image representations for deep feature learning. For binary classification, we find that the best Cartesian model achieves an accuracy of 90.3 compared to 87.2 for the best polar model. Overall, models using Cartesian images appear to perform better, see Table 6.4. This is a surprising result as polar images should be richer in tissue property information since we downsampled both images to the same size of 300×300. Due to the circular transformation, the depth scans in the Cartesian images only possess half the resolution compared to polar images. This indicates that the relevant features for plaque detection are easier to exploit from Cartesian images for deep learning models. It should be noted that practitioners also generally use Cartesian images for assessment.

We used extensive data augmentation, both for Cartesian and polar images. In particular, rotations for Cartesian images and shifting for polar images are well suited for IVOCT data. Considering the results in Table 6.4, Cartesian-based models benefit substantially more from data augmentation. Accuracy improves by $16\,\%$ for Cartesian images and $6\,\%$ for polar images. This shows that the superior performance of Cartesian images is largely tied to data augmentation.

We provide an extensive evaluation of transfer learning for IVOCT-based deep learning. The results in Table 6.5 show that there is a substantial difference in performance

between training from scratch and fine-tuning. For both models, a performance gain of approximately $15\,\%$ is achieved. Previous transfer learning approaches for other medical imaging modalities achieved similar performance improvements [447]. For IVOCT, it is still notable that pretraining on natural images works well, considering that the modalities are vastly different. For further investigation, we considered partially freezing early layers of the architectures, which is motivated by the fact that earlier layers tend to learn more generic features [573]. While our results show that performance is still high compared to training from scratch, the accuracy deteriorates the more weights are frozen during training. This indicates that IVOCT features are very different from natural image features, and thus, full retraining should be performed. Moreover, we compared how transfer learning affects performance for polar and Cartesian representations. Pretraining appears to be more effective with Cartesian images, which might be related to the fact that the source domain images are also Cartesian. Despite the significant difference between natural and IVOCT images, the shared Cartesian structure appears to be beneficial for transfer learning.

Besides representation-specific data augmentation, we also considered a fusion of polar and Cartesian image features for additional insights and performance gain. Overall, the two-path models with late feature fusion slightly increase performance over the best single-path models by $1.4\,\%$ for TP-RN2D-50 and $1.8\,\%$ for TP-DN2D-121, see Table 6.4 and Table 6.6. First, this underlines that deep learning models are able to extract different features from polar and Cartesian representations, as the performance increases when using both. Also, the learned features appear to differ earlier in the network, as early feature concatenation decreases performance. Second, we show an effective way of reusing pretrained weights for a modified architecture. As discussed before, transfer learning significantly boosts performance, and thus pretraining is mandatory for high-performance models. Our simple yet effective initialization strategy at the fusion point shows that task-specific architecture design is still feasible when being tied to pretrained models. It should be noted that this concept works consistently across both the RN2D-50 and DN2D-121 model, although their structure and feature propagation mechanisms are different.

Besides binary plaque detection, we consider a subdivision into calcified and fibrous/lipid plaque types. While fibrous and lipid plaques are relevant for risk assessment in terms of potential ruptures and thrombosis [532], the detection of calcified regions is relevant in terms of tool selection for treatment. Often, the use of rotablation or a cutting balloon is required for calcified regions. Our results in Table 6.7 show that calcified plaques can be distinguished from others. However, the overall F1-scores are significantly lower than for binary classification despite both approaches being based on the same image data. This indicates uncertainty about specific plaque types. Considering our labeling strategy, this indicates that plaque differentiation is subjective and dependent on the labeling expert. This is not surprising in the sense that plaque development over time is continuous. For example, the decision whether a plaque deposit is still to be considered fibrous or already calcified is not always clear and might change even within the same plaque region. Our results highlight this well-known issue [486], and suggest extended research towards both experts' labeling and multi-class learning approaches.

Summary

We study two different data representations for spatial 2D IVOCT images in the context of deep learning methods. With respect to our research question on data representations, we find that using the Cartesian representation is generally preferable for the problem. However, the advantage of the representation is largely tied to effective data augmentation strategies. These observations are consistent across binary and multi-class problem formulations. Regarding our research question on deep learning models, we find that different architecture concepts work well for the task. Also, using transfer learning is useful, particularly for Cartesian images. We show a new application of our Siamese architecture concept and demonstrate the effectiveness of a new initialization strategy that is a variant of our multi-dimensional transfer learning method.

6.4 2D and 3D Data: MRI-Based Left Ventricle Quantification

Another problem that is related to the state and function of the heart is LV quantification, which we introduced in Section 5.4. Here, the goal is to derive geometric parameters of the heart's left ventricle throughout a cardiac cycle. These parameters are essential indicators for diseases and function of the heart. Often, the imaging modality MRI is used for this problem. A temporal sequence of 2D spatial slices is acquired that captures the entire cardiac cycle. For each frame, geometric parameters need to be derived. Thus, learning from individual 2D images could work well. However, considering temporal context in terms of 3D spatio-temporal data might allow for more consistent estimates and thus improved performance. This raises the question of whether 2D spatial or 3D spatio-temporal data is preferable for the problem and which type of models should be employed for each data representation. Another aspect is the general approach for estimating geometric parameters. While the typical approach is to estimate parameters from segmentation masks of the LV, direct regression methods without intermediate segmentation have gained popularity.

We address these open problems in the context of the LVQuan19 Challenge. The challenge comes with additional aspects to consider. In contrast to a majority of previous work, the associated dataset is significantly smaller (56 patients), and the MRI images are hardly preprocessed with a high spatial resolution but without any region of interest cropping. Thus, an algorithm needs to deal with small dataset size, and make use of the high image resolution while focusing on the relevant region in the image.

To address all these challenges, we take previous approaches into account while putting a strong focus on using pretrained CNNs and transfer learning. Models trained for the common problem of image classification can be adapted for regression by replacing the output layer. Thus, we perform direct LV indices regression using various pretrained CNNs. For temporal processing, we employ spatio-temporal 3D CNNs. The use of these 3D CNNs is enabled by our multi-dimensional transfer learning strategy we introduced in Section 4.1.3 that allows for the transfer of 2D pretrained models to 3D. Also, we address high spatial image resolution paired with uncertain ROIs by using a multi-crop evaluation strategy that covers the entire image. To incorporate spatial segmentation information into predefined models, we propose an architecture-independent regularization by adding a decoder to the model. Finally, we integrate all our models with a new ensembling approach, where we automatically select the best-performing ensemble for each regression and classification task.

Summarized, with regard to our research questions, we investigate whether 2D or 3D spatio-temporal are advantageous for this problem. In terms of deep learning models, we apply our multi-dimensional transfer learning strategy for using 2D models in the 3D domain. Furthermore, we explore whether 2D and 3D models benefit from the additional spatial context in terms of a segmentation regularization strategy and whether combining 2D and 3D models by ensembling can improve performance.

(a) 2D Image (b) Segmentation (c) LV Indices

Fig. 6.14: Example images, ground-truth indices and segmentation masks for left ventricle quantification.

Methods and Datasets

Problem Definition. The problem at hand is to derive LV indices and the cardiac phase from cardiac MR images. We use a single 2D slice $B_{t_i} \in \mathbb{R}^{n_h \times n_w}$ to learn a target vector $\mathrm{y}_{t_i} \in \mathbb{R}^{N_r}$ containing N_r entries with both LV indices and the cardiac phase. Also, we use a spatio-temporal sequence of slices $V_{t_i} = \{B_{t_{i-n_t+1}}, ..., B_{t_i}\}$ with $V_{t_i} \in \mathbb{R}^{n_t \times n_h \times n_w}$ to derive a target matrix $\mathrm{Y}_{t_i} \in \mathbb{R}^{n_t \times N_r}$ that describes the LV indices and cardiac phase across a number of slices n_t. Thus, we try to find deep learning models $f_M : \mathbb{R}^{n_t \times n_h \times n_w} \rightarrow \mathbb{R}^{n_t \times N_r}$.

Dataset and Preprocessing. The LVQuan19 training dataset consists of short-axis MRI data from 56 patients. For each patient, 20 slices representing one cardiac cycle are provided. The resolution of the MRI slices is either 256×256 or 512×512 with a pixel spacing between $0.6836 \frac{\mathrm{mm}}{\mathrm{pixel}}$ and $1.7188 \frac{\mathrm{mm}}{\mathrm{pixel}}$. For each slice, segmentation masks of the myocardium and cavity area, all 11 LV indices and the cardiac phase are provided. The LVQuan19 Challenge goal is the estimation of all 11 indices and the cardiac phase, and the segmentation masks can be used optionally. Example image data, ground-truth indices, and the segmentation map are shown in Figure 6.14.

First, we resize all images to have a pixel spacing of $1 \frac{\mathrm{mm}}{\mathrm{pixel}}$. Second, we take a center crop of size 300×300, which is the smallest image size of all resized slices. We clip intensities between the 1st and 99th percentile. Afterward, we perform image normalization by subtracting the mean and dividing by the standard deviation for each slice. Then, the intensities are scaled between 0 and 1. We convert the target indices to mm. Then, we scale all regression targets to a range of 0-1 for training. For evaluation, the targets are scaled back to their original range. We split the dataset into 5 CV folds.

Deep Learning Models. The key idea of our approach is to use pretrained 2D CNN architectures for indices regression. Thus, we consider a pool of pretrained architectures including DenseNet [208], ResNet [193], ResNeXt [553], and Squeeze-and-Excitation Networks [205], as introduced in Section 4.1.3.

Each CNN was pretrained on ImageNet for classification of natural images into 1000 classes. We replace the model's output layer to match the number of outputs for our problem. We consider regression only with 11 outputs, classification (cardiac phase) with 2 softmaxed outputs, or both simultaneously with 13 outputs. The pretrained model

expects a 3-channel input image, which we handle with two approaches. Either, we use a single slice, copied to all three channels, or we include the two neighboring slices.

We also consider spatio-temporal 3D CNN approaches. We hypothesize that temporal context might improve indices regression. Furthermore, cardiac phase classification requires temporal context, and including more slices might be helpful. We employ the same type of models as for 2D processing, except that all convolutional kernels are isotropically extended by a third dimension. To overcome the significant increase in trainable parameters and the associated risk of overfitting, we employ our multi-dimensional transfer learning strategy introduced in Section 4.1.3. The 3D CNN input is of size $\mathrm{x} \in \mathbb{R}^{n_t \times n_h \times n_w}$, where n_h and n_w are the spatial slice dimensions and $n_t \in [1, 20]$ is the number of selected slices. Throughout the entire network, we do not change the slice dimension, and we produce n_t predictions for n_t input slices in a single forward pass. For this purpose, we replace the linear output layer by a convolutional layer with kernel size 1, which is able to handle arbitrarily-sized inputs. Due to the increase in memory requirements, we only consider 3D variants of the smaller CNNs in our pool, which includes DN3D-121, DN3D-161, and SE-RN3D-101.

In addition, we implement a segmentation-based regularization (SR). Here, we add an additional decoder to the architecture, before its GAP layer. The decoder upsamples the spatial dimension of the feature maps in several steps until the original input image size is reached. At each step, we apply a convolution with kernel size 1, which halves the current feature map dimension, followed by nearest neighbor upsampling with a factor of 2. Then, a standard ResNet block is applied. In total, there are 5 upsampling stages. At the output, we predict a softmaxed probability map, which is used to calculate a cross-entropy loss with the ground-truth masks. During training, the loss is propagated through the entire network, forcing the core architecture to learn spatial features for segmentation, and indices regression simultaneously. We do not use the predicted segmentation explicitly for indices calculation, and it only serves as a regularizer for the network. We employ this regularization strategy both for 2D and 3D CNNs.

Data Augmentation. Due to the small dataset size, we employ extensive data augmentation. We use random rotation with $\theta \in [0°, 360°]$ which we found more effective than simple $90°$ rotations. Furthermore, we employ random scaling by resizing the slices by a factor of $s_c \in [0.8, 1.2]$ with appropriate cropping and zero padding, if required. The targets are scaled accordingly (quadratic for areas). We chose a small batch size of $N_b = 8$ to induce more variation during training. We did not see any improvement for dropout, L^1, or L^2 regularization.

The pretrained models' standard input size is 224×224 while our preprocessed images are of size 300×300. Therefore, we crop patches from random image locations during training. This should have a regularizing effect as the CNN gets more robust towards different relative LV locations in the images. For 2D CNNs, we randomly select a cropped slice from N_b different patients to construct a batch of size N_b. For 3D CNNs, we randomly select a sequence of n_t slices from N_b patients for each batch. For evaluation, we employ a multi-crop evaluation approach to cover the entire image. We crop from 4 predefined locations in each slice and average the result. For 2D CNNs, this is repeated for every slice for each patient. For 3D CNNs, we crop sequences of n_t slices from every possible location $n_t \in [1, 20]$. Then, we average over all overlapping regions to obtain a final prediction for each slice. To handle the start and end of the sequence,

we use cyclic repetition.

Training and Evaluation. In total, our loss function consists of the MSE for regression, the CE loss for phase classification, and the CE loss for segmentation. The loss functions were introduced in Section 3.3.6. The two CE losses are not present in every model. If they are included, phase classification is weighted by $\lambda_P = 0.05$ and the segmentation loss is weighted by $\lambda_S = 0.1$. The final loss is the sum of all individual loss functions. For optimization, we employ Adam with a learning rate of $\alpha_{lr} = 1 \times 10^{-4}$. We train each CV model for $N_e = 150$ epochs, where a single epoch consists of 10 random crops from each patient in the training set. Overall, we train multiple models with different hyperparameter configurations h_M^i. The configuration options include the network architecture (for example, DN2D-121), the input dimension (2D or 3D), $n_t \in \{3, 5, 7, 10\}$ for 3D CNNs, weight initialization (random or ImageNet), segmentation-based regularization (on or off), and prediction targets (areas, dimensions, RWTs, phase).

We consider the MAE with standard deviation in mm (mm^2 for areas) and PCC in % for regression and the error rate (ER) for classification. Configurations include no pretraining (No Pre.), no weight scaling for multi-dimensional transfer learning (No WS), SR, joint indices regression, and phase classification (Joint) and phase classification only (Class.). We test for statistically significant differences in the absolute error with the Wilcoxon signed-rank test and a significance level of α = 5%.

Ensembling. Instead of deciding for a single model we search for the optimal ensemble. Let $H_M = \{h_M^i, \ldots, h_M^n\}$ be the set of all configurations we consider, and let $CV = \{\mathcal{V}^1, \ldots, \mathcal{V}^m\}$ be the set of all CV splits. Then we obtain predictions $\hat{\mathrm{y}}_{ij} = f_M(h_M^i, \mathcal{V}^j)$ for all combinations of i and j after training a model with h_M^i on $CV \setminus \mathcal{V}^j$. We obtain the predictions per configuration through concatenation as $\hat{\mathrm{y}}_i = \cup_j \hat{\mathrm{y}}_{ij}$. Subsequently we perform an exhaustive search to identify the set $H_M* \subseteq H_M$ such that $\hat{\mathrm{y}}* = \frac{1}{|H_M*|} \sum_{k \in H_M*} \hat{\mathrm{y}}_k$ minimizes the error between $\hat{\mathrm{y}}*$ and ground-truth y. For the challenge test set we obtain predictions with all models in the optimal subset $S*$ which are subsequently averaged into a final prediction. We search for the optimal subset for each task (areas, dimensions, RWTs, phase) individually to maximize performance. To keep the search time bounded, H_M only includes our top 20 configurations, ranked by individual CV performance.

Our proposed strategy potentially leads to implicit overfitting of the subset choice in terms of CV performance. Therefore, results reported for the CV sets might overestimate the performance gain of our ensembling strategy. We overcome this problem by introducing an additional test split for ensemble evaluation only. Here, we split each CV fold into two folds. We use the first portion of these new sets to find the optimal subset with our strategy. Then, we use the second portion of sets to evaluate the strategy.

Results

Results for different configurations with DN2D-121 and DN3D-121 are shown in Table 6.8. For DN2D-121 with 2D slice inputs, the pretrained model with segmentation regularization performs best. The difference in the median of the absolute errors of DN2D-121 and DN2D-121 SR is significant for areas and dimensions. Combining regression and classification leads to a lower performance than training separate models.

Tab. 6.8: All results for different configurations with DN2D-121. The MAE is given with standard deviation in mm (mm^2 for areas) and the PCC is given in %.

	Areas		**Dimensions**		**RWTs**		**Phase**
	MAE	PCC	MAE	PCC	MAE	PCC	ER
DN2D-121 No Pre.	199 ± 129	93.5	2.98 ± 1.8	94.5	1.55 ± 0.9	77.0	-
DN2D-121	139 ± 74	97.2	2.38 ± 1.3	96.7	1.33 ± 0.7	83.5	-
DN2D-121 SR	$\mathbf{133 \pm 76}$	**97.4**	$\mathbf{2.08 \pm 1.2}$	**97.5**	$\mathbf{1.30 \pm 0.7}$	**84.7**	-
DN2D-121 Joint	161 ± 89	96.0	2.54 ± 1.3	96.3	1.39 ± 0.7	82.3	9.0
DN2D-121 Class.	-	-	-	-	-	-	**8.4**
DN3D-121 No Pre.	180 ± 165	94.0	2.77 ± 2.4	94.3	1.48 ± 0.8	79.8	-
DN3D-121 No WS	171 ± 117	95.5	2.53 ± 1.5	95.2	1.40 ± 0.8	81.4	-
DN3D-121	133 ± 82	97.1	2.26 ± 1.3	96.8	1.32 ± 0.7	83.8	-
DN3D-121 SR	$\mathbf{126 \pm 71}$	**97.5**	$\mathbf{2.14 \pm 1.2}$	**97.2**	$\mathbf{1.30 \pm 0.8}$	**84.4**	-
DN3D-121 Joint	146 ± 74	96.6	2.33 ± 1.2	96.9	1.43 ± 0.8	81.2	7.9
DN3D-121 Class.	-	-	-	-	-	-	**7.5**

For DN3D-121 ($n_t = 5$), the overall performance improves slightly with respect to its 2D counterparts. Again, the difference in the median of the absolute errors of the 2D and 3D model is significant for areas and dimensions.

Regarding our multi-dimensional transfer learning strategy, we can observe that it improves performance significantly, both compared to training from scratch and to transfer learning without weight scaling. In Figure 6.15, we can observe that the model with weight scaling convergences best.

Considering different architectures in Table 6.9, improved performance can be observed for larger models. The best performing model is RX2D-101-64d. With respect to ensembling, taking the average over all our models does not perform better than the best single model. We improve performance by using our optimal subset strategy. When evaluating on a different test split (*), the performance difference is still large between averaging and our strategy. The difference in the median of the absolute errors for averaging and our ensembling strategy is statically significant for all three indices. Our final ensembles mostly contain the models RX2D-101-64d SR, DN2D SR variants, and DN3D-121 SR. On the test set of the LVQuan19 Challenge, the performance of the ensemble is substantially lower. For reference, we include the results from DMTRL [561]. Note that these results are not directly comparable, as a different number of patients and different image resolutions were used.

Discussion

We address LV quantification from cardiac MR images with a focus on 2D and 3D image data with retrained models. We consider a variety of deep learning models that have been successful for the classification of 2D images. We find a substantial increase in performance with pretrained weights, for example, the MAE for area estimation improves from $199\,\mathrm{mm}^2$ to $134\,\mathrm{mm}^2$. This is likely tied to the new and small dataset,

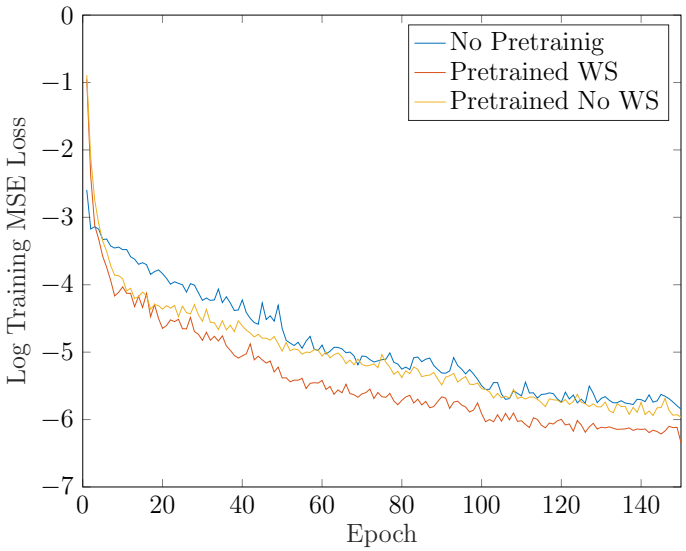


Fig. 6.15: We show the convergence of DN3D-121 for different transfer learning strategies in the context of our multi-dimensional transfer learning strategy. This includes the use of no pretrained weights, straightforward copying of pretrained weights, and our scaling rule after copying weights.

which contains only roughly a third of the number of patients compared to most previous studies [528, 561]. At the same time, the MRI images have a higher resolution, which is closer to the standard input resolution of models trained on ImageNet. Both likely lead to a substantial advantage of utilizing pretrained models.

To incorporate previous segmentation-based approaches, we propose a segmentation regularization by adding an architecture-independent decoder close to the model output. The additional segmentation loss forces the model core to learn both features for direct indices regression and myocardium segmentation. Our results indicate that including the segmentation is advantageous as we observe a statistically significant increase in performance both for areas and dimensions. This matches insight from previous work, where using both a segmentation mask and LV indices lead to improved results [528,558].

Furthermore, we address temporal context by extending the existing 2D CNN models to 3D. To enable 3D CNN usage with very limited data, we use our multi-dimensional transfer learning strategy, where we copy pretrained 2D weights to 3D. Again, we find a substantial increase in performance by relying on pretraining, see Table 6.8. Notably, the weighting scaling step appears to be important to enable this transfer from 2D to 3D, see Figure 6.15. Without weight scaling, convergence appears to be limited and closer to no use of pretrained weights.

Overall, there is a significant improvement for dimension and area regression for 3D CNNs over 2D CNNs with an MAE of $139\,\mathrm{mm}^2$ compared to $133\,\mathrm{mm}^2$ for the areas. For the cardiac phase, it is notable that the 2D approach with neighboring slices performs reasonably well and only slightly worse than 3D CNNs. Overall, enabled by our initialization strategy, spatio-temporal 3D CNNs improve performance over spatial 2D CNNs for LV quantification.

Tab. 6.9: All results for different architectures and ensembling. The MAE is given with standard deviation in mm (mm^2 for areas) and the PCC is given in $\%$. Results marked with a star (*) are evaluated on a different test split.

	Areas		**Dimensions**		**RWTs**		**Phase**
	MAE	PCC	MAE	PCC	MAE	PCC	ER
DN2D-169 SR	122 ± 72	97.6	1.99 ± 1.2	97.6	1.27 ± 0.7	85.3	-
DN2D-161 SR	127 ± 81	97.1	2.00 ± 1.1	97.8	1.30 ± 0.7	84.3	-
SE-RN2D-101 SR	126 ± 70	97.4	2.01 ± 1.1	97.7	1.32 ± 0.8	84.4	-
SE-RN2D-152 SR	124 ± 81	97.4	2.07 ± 1.2	97.5	1.29 ± 0.7	84.9	-
RX2D-101-64d SR	$\mathbf{118 \pm 72}$	**97.6**	$\mathbf{1.99 \pm 1.0}$	**97.8**	$\mathbf{1.25 \pm 0.7}$	**85.9**	-
SE-RX2D-101 SR	135 ± 80	97.1	2.23 ± 1.2	97.3	1.33 ± 0.8	83.9	-
SENet2D-154 SR	129 ± 81	97.3	2.12 ± 1.3	97.3	1.31 ± 0.8	84.6	-
Ensemble Average	118 ± 76	97.7	1.96 ± 1.1	97.8	1.26 ± 0.6	85.6	7.7
Ensemble Optimal	$\mathbf{111 \pm 76}$	**97.9**	$\mathbf{1.84 \pm 0.9}$	**98.0**	$\mathbf{1.22 \pm 0.6}$	**86.4**	**6.7**
Ensemble Average*	117 ± 75	97.8	1.96 ± 1.0	97.8	1.23 ± 0.6	86.0	8.3
Ensemble Optimal*	111 ± 75	97.9	1.85 ± 1.0	98.0	1.19 ± 0.6	87.1	7.0
Ensemble Testset	371	92.5	3.02	95.7	2.53	82.6	11.5
DMTRL [561]	180 ± 118	94.5	2.51 ± 1.6	92.5	1.39 ± 0.7	76.8	8.2

Next, we consider different baseline architectures for our approach. Using larger architectures with more layers and more feature maps tends to improve performance over the DN2D-121 baseline. In particular, it is notable that the highest performance increase among different models is substantially larger than the performance increase of moving from 2D to 3D. RX2D-101-64d improves the MAE by $16\,\mathrm{mm}^2$ over DN2D-121 compared to a $6\,\mathrm{mm}^2$ decrease caused by using 3D convolutions. In the best case, one would extend the best 2D model to 3D for performance maximization, however, this is limited by GPU memory and not feasible for the larger, higher-performing 2D models. Summarized, using high-performing 2D architectures can be very beneficial for LV quantification.

Last, we combine all our models with a new ensembling technique, where the best performing models were automatically selected based on cross-validation performance. The method improves performance over simply averaging predictions across all models. Also, we used separate test splits to ensure that the optimal subset selection does not implicitly overfit to the CV sets. Even for this evaluation scenario, our ensembling method performs better than averaging with statistically significant performance differences. Interestingly, the selection method included both 2D and 3D CNNs, which indicates that both spatial 2D and spatio-temporal 3D information is important for LV indices regression. On the LVQuan19 test set, our method performs substantially worse than in our CV experiments. This indicates that the test set is very challenging and potentially differs from the training set. Similar observations were made for the last year's challenge [558]. Thus, generalizable LV quantification remains a challenging task.

Summary

We study the use of 2D spatial and 3D spatio-temporal data representations for deep learning-based left ventricle quantification. Regarding our research question on data representations, we find that using 3D spatio-temporal data slightly improves performance over individual 2D spatial slices. With respect to our research question on deep learning methods, pretrained models are very effective for this task, both for 2D and 3D models. In particular, our multi-dimensional transfer learning strategy with weight scaling enables the use of pretrained 3D models. However, we observe that using high-performing 2D models provides a more significant benefit than extending models from 2D to 3D. Still, when constructing an optimal ensemble, 2D and 3D models synergize well and lead to better performance than any single model. We also observe that adding additional spatial context with a segmentation regularization strategy can improve performance slightly.

6.5 2D and 3D Data: OCT-Based Pose Estimation

Another way of extending 2D CNNs to 3D is to engineer new, efficient 3D CNNs that are loosely based on concepts that have been introduced for the 2D image domain. This approach is particularly suitable for problems where very efficient CNN architectures are required. In contrast, the straightforward extension from 2D to 3D that we studied in the previous section always leads to huge architectures.

In the field of deep learning-based computer-assisted interventions, CNNs often have to be applied in real-time. One example is the problem of pose estimation, which we introduced in Section 5.3. Here, the pose of a patient or a surgical tool needs to be tracked continuously throughout an intervention. Given that many surgical interventions such as neurosurgery or ophthalmic surgery require a high level of precision, CNN models for pose estimation need to be both highly accurate and fast.

Regarding the problem of real-time pose estimation with high accuracy, OCT is well-suited due to its high spatial and temporal resolution. As discussed in Section 5.3, there are early approaches for OCT-based pose estimation that used 2D OCT data representations. Given the requirement of highly accurate estimates, using full 3D OCT volumes might be advantageous. At the same time, the increase in computational requirements for higher-dimensional models makes architecture design for this problem very challenging.

Therefore, we study OCT-based pose estimation in the context of different data representations and CNNs architectures. For this purpose, we propose a specialized framework for data acquisition and model application. We use arbitrary small objects and turn them into a marker for pose estimation or tracking. We employ an experimental setup with a robot that allows for automatic acquisition of OCT volumes with the marker and its respective pose provided by the robot. Thus, we have a supervised learning problem where a pose vector needs to be regressed from an OCT volume. In terms of application, the goal is to attach the marker to another object that should be tracked.

First, we investigate the use of volumetric data for deep learning-based pose estimation. We explore how directly leveraging volume information with 3D CNNs compares to the typical use of 2D depth representations. Regarding the choice of volume data as our image representation, we also analyze how 3D CNNs make use of the additional depth information. OCT is a modality that can provide deep, subsurface information. However, this depends on materials and whether they can be penetrated by infrared light. We investigate how subsurface information benefits 3D CNN learning by comparing markers with and without an identifiable inner structure. We provide quantitative results and qualitative saliency maps to show how 3D CNNs exploit volume information for pose estimation. Furthermore, we test our marker's performance when the OCT image is occluded, which provides insights on robustness.

Also, we address the aspect of architecture design. As pose estimation requires real-time application, we employ our efficient CNN extensions to the 3D domain, which includes the ResNet, Inception, and ResNeXt design principle, see Section 4.1.2. Additionally, we leverage long-range connections for efficient feature reuse throughout the CNN networks. We study how different design principles affect OCT-based pose estimation performance.

As a result, with respect to our first research question on data representations, we

study the use of 2D depth representations and full 3D volumes. Also, we investigate the effect of subsurface features in full volumes. With respect to our second research question on deep learning models, we study different design principles of CNNs when they are extended to 3D.

Methods and Datasets

Problem definition. Formally, we consider the problem of estimating a target pose $\mathrm{y}_i \in \mathbb{R}^{N_r}$ that consists of position and orientation with N_r total regression targets. We use full volumes $C_i \in \mathbb{R}^{n_h \times n_w \times n_d}$ as well as depth representations obtained by a projection $P : \mathbb{R}^{n_h \times n_w \times n_d} \rightarrow \mathbb{R}^{n_h \times n_w}$. This results in depth representations $D_i \in \mathbb{R}^{n_h \times n_w}$. Thus, we try to find deep learning models $f_M : \mathbb{R}^{n_h \times n_w} \rightarrow \mathbb{R}^{N_r}$ and $f_M : \mathbb{R}^{n_h \times n_w \times n_d} \rightarrow \mathbb{R}^{N_r}$.

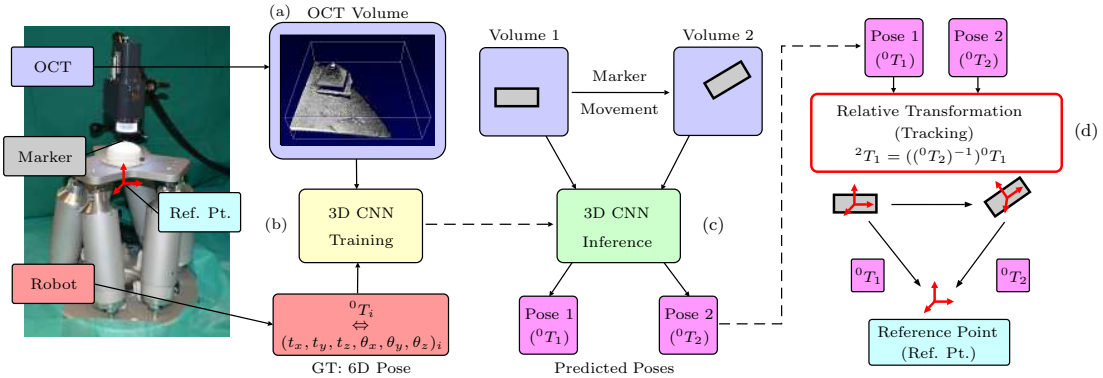


Fig. 6.16: We show a sketch of our approach. During the initial experiment, the hexapod robot moves the marker to predefined, randomly generated poses within the OCT's FOV. The poses are expressed with respect to the hexapod's reference point. In each pose, a volume is obtained while the robot is standing still (a). Next, the data samples are used for training a 3D CNN (b). Finally, the trained model (c) predicts a pose from an OCT volume. Two poses are used for tracking by obtaining the relative transformation (d) between the two marker poses.

Data Generation and General Setup. We employ a setup to automatically generate a set of image and pose data for learning. The setup consists of a hexapod robot, a SD-OCT device with a stand, and a phantom to be used as a marker, see Figure 6.16. The hexapod moves the marker inside the OCT's FOV and stops at predefined poses. The position part of the 6D poses is generated by randomly sampling positions in a 3D bounding box that covers the OCT's FOV size. Orientations are created by randomly generating rotation angles within an interval. All components are uniformly sampled from their respective space. The hexapod moves to a pose, stops, and an OCT volume is acquired. The volume is combined with the current pose to form a labeled data sample. This procedure is repeated several thousand times to create a dataset for training. As a result, our 3D CNNs receive an OCT volume containing the marker as their input and are trained in order to predict the pose with respect to the hexapod's reference point. Tracking is achieved by letting the CNN predict the marker's pose in two different volumes. Then, the relative

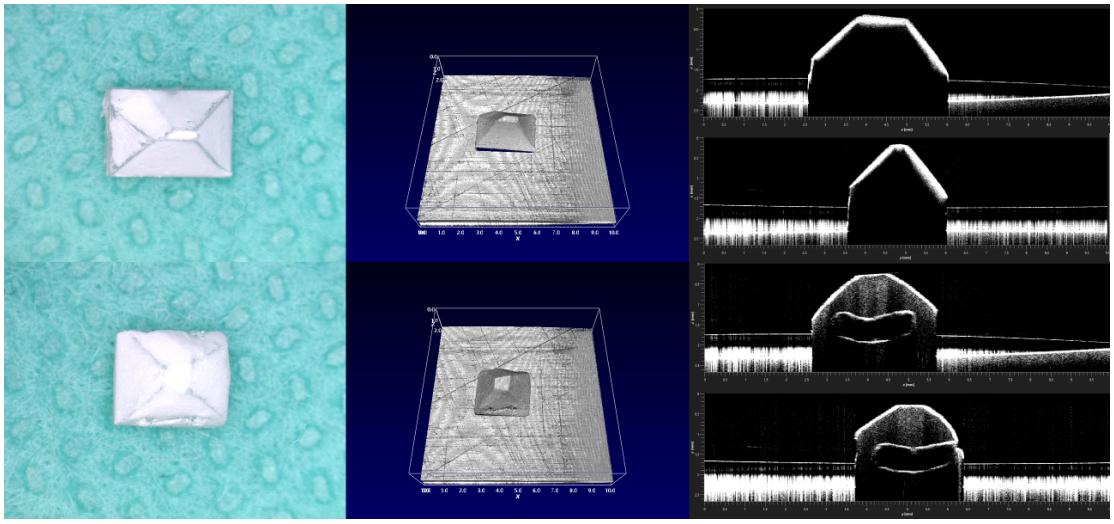


Fig. 6.17: We show the two markers we employ. Each row shows different image representations for each marker. From left to right: Digital microscopy image, rendered volume, B-Scan slices along the x and y direction. The key difference is the inner structure of the second marker that is only visible in OCT volumes.

transformation can be easily obtained by matrix multiplication. This is depicted in the right part of Figure 6.16.

Markers. Marker A was milled from a block of polyoxymethylene (POM) with an asymmetric prism shape and a size of approximately $3.75\,\mathrm{mm} \times 2.4\,\mathrm{mm} \times 2\,\mathrm{mm}.$, see Figure 6.17. The material reflects the infrared light very well, which is why mostly its surface is visible in an OCT volume, not its interior. The second marker B was 3D printed with Formlabs Resin to obtain an inner structure. The marker has a size of $3.2\,\mathrm{mm} \times 2.68\,\mathrm{mm} \times 1.9\,\mathrm{mm}.$

OCT Device. The imaging device is a Thorlabs Telesto I SD-OCT system with a center wavelength of $1325\,\mathrm{nm}$ for 3D volume acquisition. Its lateral resolution is $15\,\mu\mathrm{m}$ and its depth resolution is $7.5\,\mu\mathrm{m}$. Its FOV covers a volume of $10\,\mathrm{mm} \times 10\,\mathrm{mm} \times 2.66\,\mathrm{mm}$. Volume images are acquired with a size of $128 \times 128 \times 512$ voxels. While all images are volumetric, the visibility of a marker's interior structure largely depends on the object's reflective properties. If it reflects near-infrared radiation very well, only the object's surface will be visible in an OCT volume. This is a very relevant property when considering the pose estimation task. Typical 6D pose estimation frameworks [264] also rely on surface information obtained with time-of-flight depth cameras. Therefore, it appears natural to employ a similar framework for OCT images if mostly surfaces are visible without internal features. We investigate this assumption by training both on volume data and 2D surface extractions. Also, we train both on an opaque marker, whose surfaces are hardly penetrated and a marker with a distinct inner structure, visible in OCT volumes. Both approaches provide insight into the importance of volume data usage. Figure 6.17 shows the different markers with the different properties. We refer to the opaque marker as marker A and the marker with an inner structure as marker B.

Robot. The hexapod robot shown in Figure 6.16 is a 6-axis H-820.D1 hexapod distributed by Physik Instrumente GmbH. It is used to move the marker within the

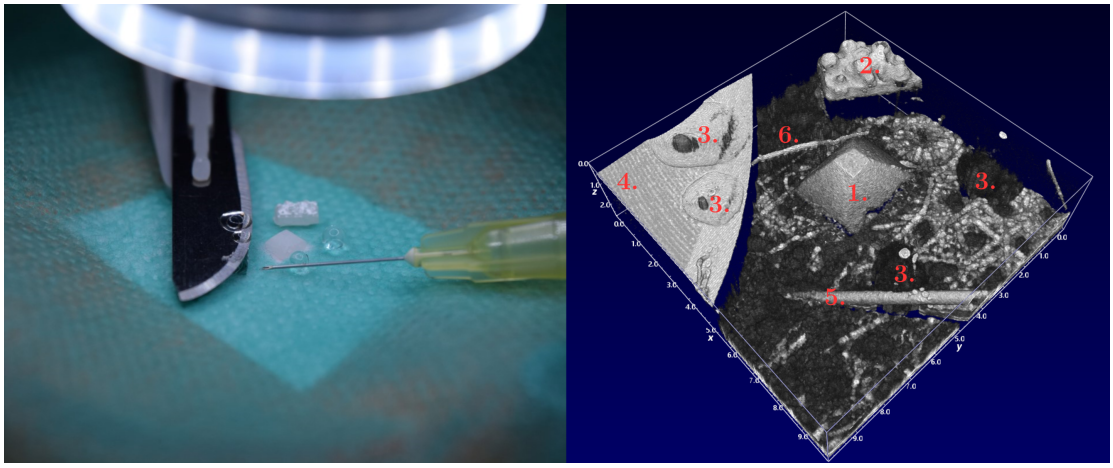


Fig. 6.18: We show an example of the occlusion dataset. Left, a photography of the setup is shown. Right, the corresponding OCT volume is shown. We use marker B for this experiment. Note that we vary both the position of the objects and the objects themselves during data acquisition of this set. 1. marker B 2. printed geometry/arbitrary marker 3. water droplets 4. scalpel 5. needle of a syringe 6. cloth fiber.

OCT's FOV as well as for obtaining ground-truth 6D pose labels. Its pose is expressed with respect to a reference point slightly below its top plate. Translations relative to that point are denoted as t_x, t_y and t_z. The rotations are expressed by rotation angles α_{rot}, β_{rot}, γ_{rot} around each axis of a coordinate frame shifted by t_x, t_y and t_z from the reference point. A rotation matrix is expressed by consecutively rotating with α_{rot}, β_{rot} and γ_{rot} around the moving axes x, y', z'', such that the rotation matrix can be expressed as $R = R(\alpha_{rot})R(\beta_{rot})R(\gamma_{rot})$. The rotation matrix R and the translations are used to form a homogeneous transformation matrix that is used to obtain the relative transformation matrix as shown in the right part of Figure 6.16. The target pose labels for learning take the form $\mathrm{y} = \{t_x, t_y, t_z, \alpha_{rot}, \beta_{rot}, \gamma rot\}$. Regarding accuracy, the robot is limited by a translational repeatability of $\pm 20\,\mu\mathrm{m}$ and a rotational repeatability of $\pm 11.46 \times 10^{-3\circ}$. The range of positions covered by the hexapod robot in the experiment corresponds to the OCT's FOV. The rotations are limited to a range of $(-10°, 10°)$ for each axis.

*Datasets.*For both markers we acquired several thousand data samples each, using roughly $80\,\%$ for training and $10\,\%$ for validation and $10\,\%$ for testing. Additionally, we acquired a dataset that contains occlusions. Here, we acquired a dataset where we added random objects around the marker. The occluding objects were repositioned and changed during acquisition. We used a variety of objects with different reflective properties such as a scalpel, parts of a syringe, needles, cloths, different plastic, and metal parts, surgical scissors, printed geometries that could be used as markers and water droplets on top of and next to the marker. An example occlusion scenario is shown in Figure 6.18. Our marker is the only object constantly appearing in all volumes, and we investigate whether this helps the model to learn robustness towards all other objects. For testing, we split off a dataset that contains objects that are not present anywhere else in the training dataset. Therefore, performance on this test set indicates how well the model

Tab. 6.10: Number of samples for each dataset. The occlusion dataset was recorded with marker B.

	Marker A	Marker B	Occlusion
Training	5850	5850	15000
Validation	900	900	-
Testing	900	900	2875

deals with objects that it has never seen before. This provides a realistic impression on how the model will perform in practice, where new objects are likely to appear in the OCT volumes. An overview of the datasets is shown in Table 6.10. All results we present refer to the test sets. Note, that there is no validation set for the occlusion dataset as we directly use it with our models that were fine-tuned on the other two datasets.

Preprocessing. For volume data, the volume size needs to be adjusted first due to computational requirements. We downsample the volumes from the acquisition size of $128 \times 128 \times 512$ to $64 \times 64 \times 16$. The depth dimension is reduced with a larger factor than the lateral dimensions because its original pixel spacing is much smaller. As a result, the pixel spacing for each dimension of the volume represents the same Cartesian distances. The target volume size is a trade-off between computational effort and potentially lost information during the downsampling process. The selected size leads to satisfactory results while keeping training times within feasible bounds.

For 2D depth data representations, we extract surface information from the OCT volumes to obtain a 2D depth representation that is similar to other RGB-D based 6D pose estimation frameworks [59]. This allows for comparison to other OCT-based tracking approaches where 2D depth representations were used for tracking a volume of interest with handcrafted feature matching [276]. We perform the extraction using MIPs from different views. This provides us with two different types of depth representations. The image index at which the maximum intensity was found represents the most intuitive notion of depth.

However, the maximum intensity itself also provides depth information. Considering a curved Gaussian beam model of the OCT's infrared light, the intensity at the top of the volume (closer to the light source) will be different than at the bottom. Moreover, the MIPs can also carry rotation information as the back-scattering from surfaces changes based on the angle. Therefore, both the normalized depth index and the maximum intensities themselves are considered as 2D depth representations for learning. The extraction process is illustrated in Figure 6.19. Since our data is volumetric, there are several options of which coordinate direction (x,y,z) should be chosen for extraction. Here, x and y are the lateral coordinate directions, and z is the depth direction along the OCT beam. Using several 2D projections from different angles is typically referred to as 2.5D and has been used for CNN training as a trade-off between less costly 2D and potentially more fruitful 3D representations [412].

The straightforward choice is the use of the MIP along the z-direction, as this is the actual traveling direction of the OCT light beam. Taking the maximum value along the z-direction results in a projection on the x-y plane. Although this is the primary, relevant direction for OCT, some information is likely lost through the projection. To

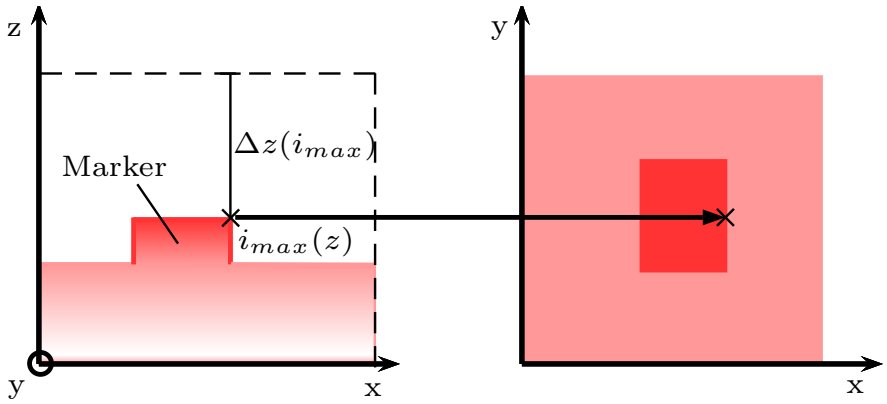


Fig. 6.19: The extraction process of 2D representations from an OCT volume is shown. Here, the extraction for an x-y projection is shown. Red color indicates the intensity in the volume. Typically, the highest intensity is found at the first surface hit by the infrared light. Inside, the intensity gradually decreases. Therefore, an MIP captures the surfaces visible in OCT data. For a depth map, the depth value $\Delta z(i_{max})$ is determined at every x-y location and transferred to a 2D map. For an MIP, the intensity $i_{max}(z)$ itself is used at every x-y location, as shown in the right part of the figure. Due to different lighting properties along the z-direction, both methods capture depth in a 2D image.

illustrate this, consider Figure 6.19. Potentially useful information below the surface is lost entirely through projection. Therefore, we also include z-y and z-x projections in our datasets. To maintain spatial alignment, we perform the MIP extraction from a volume size of $64 \times 64 \times 64$. This results in five different 2D datasets that we compare to the volumetric dataset:

1. $\mathcal{V}$: $64 \times 64 \times 16$ preprocessed volumes
2. $\mathcal{M}^1$: $64 \times 64 \times 1$ intensity values from the x-y projection
3. $\mathcal{D}^1$: $64 \times 64 \times 1$ normalized depth index values from the x-y projection
4. MD^1: $64 \times 64 \times 2$ normalized depth index values and intensity values from the x-y projection
5. $\mathcal{M}^3$: $64 \times 64 \times 3$ intensity values from the x-y, z-x and z-y projections
6. $\mathcal{D}^3$: $64 \times 64 \times 3$ normalized depth index values from the x-y, z-x and z-y projections

The third dimension is the channel dimensions.

In order to draw a connection between 2D and 3D data processing, we also consider the case of using 3D volume data with 2D kernels. Prior approaches handled OCT volume data by using 2D slices in the input data's channel dimension with 2D CNNs [426]. By default, a 2D kernel that is swept over a volume performs processing slice by slice without taking context between slices into account. For a meaningful comparison to 3D

Tab. 6.11: Feature map (f_m) choices for the residual blocks of the IN3D model.

	f_m^{j1}	f_m^{j2}	f_m^{j3}
Module $j = 1$ ResBlock /2	64	64	30
Module $j = 1$ ResBlock	42	42	20
Module $j = 2$ ResBlock /2	86	86	40
Module $j = 2$ ResBlock	64	64	30

CNNs, we extend the 3D volumes by a channel dimension for 2D kernel processing. Each channel contains a shifted version of the volume along the z-direction. Therefore, when processing each slide with a 2D kernel, the neighboring slices are also taken into account.

Summarized, we use five datasets with 2D depth representations for comparison to a volumetric dataset. This provides a comparison on how computationally cheaper 2D representations perform against more costly 3D data when being trained with a 2D CNN and 3D CNN, respectively. The baseline dataset for our evaluation is the volumetric dataset.

Deep Learning Models. We employ our carefully designed 3D CNN architecture extensions that we introduced in Section 4.1.2. The general architecture introduced in Section 4.1.2 focuses on feature extraction at intermediate volume sizes after the initial downsampling stage. Here, the greatest processing effort is focused on the two architecture-specific modules with spatial feature map sizes of $16 \times 16 \times 4$ and $8 \times 8 \times 2$. In this way, the computational effort is lowered significantly, and at the same time, more abstract, discriminative features can be learned.

We implement *RN3D-A* with an initial feature map size of $f_m^1 = 64$ that is doubled within the first ResBlock of each module. In total, the architecture contains 4 ResBlocks, equally distributed across the two modules. *RN3D-B* is implemented similarly with $f_m^1 = 64$ initial feature maps. Due to the more efficient bottleneck blocks, the model comes with 9 instead of 4 ResBlocks while keeping the number of model parameters at a similar level. *RX3D* uses the same number of feature maps and ResBlocks as RN3D-A. We employ a cardinality of $n_{cd} = 8$, which leads to 8 individual processing paths for each block. Finally, *IN3D* also starts with $f_m^1 = 64$ feature maps but uses different numbers of feature maps within the architecture paths. The feature map sizes are given in Table 6.11. From this architecture, we also build *IN2D*, a 2D variation of IN3D with a similar structure but 2D convolutional layers instead of 3D. An overview of the model's properties and parameter size is given in Table 6.12. Note that all our architectures are very efficient in terms of the number of parameters. For comparison, the standard RN2D-50 architecture [193] with 16 residual blocks and 2D convolutions comes with $21\,000\,000$ parameters. Inception-ResNet [475] contains 22 blocks and $56\,000\,000$ parameters.

Model Training. The learning task is formulated as a regression problem, which is why the error function to be minimized is chosen to be the MSE between network outputs and ground-truth labels. The CNNs are trained with mini-batch gradient descent. We use the Adam algorithm with an initial learning rate of $\alpha_{lr} = 10^{-4}$. When the validation error saturates, the learning rate is reduced by a factor of 5 until we observe

Tab. 6.12: Overview of the different architectures we extend from 2D to 3D. All models employ residual connections. Except for RN3D-A, all models make use of downsampling in the feature map dimension. IN3D and ResNeXt3D additionally contain multiple paths at each stage, representing feature extraction at different scales. IN3D's paths are individually fine-tuned while RX3D follows simple design rules for its path design. Lastly, the total number of parameters and blocks is provided for each model. Note that RN3D-A only has 4 blocks in order to keep its number of parameters in a similar range.

	RN3D-A	RN3D-B	IN3D	RX3D
Residual Connections	Yes	Yes	Yes	Yes
Bottleneck	No	Yes	Yes	Yes
Multi-Path	No	No	Yes	Yes
Individual Path Design	No	No	Yes	No
# of Parameters	6 161 907	3 451 507	3 568 913	3 042 931
# of Blocks	4	9	9	9

no further improvement.

We split the data set into training, validation, and test sets. The validation set is used for fine-tuning hyperparameters, the test set is used for evaluating the final performance. During training, we use a batch size of $N_b = 15$.

The labels used for training are provided by the hexapod robot. Due to the OCT's limited FOV, the positions are limited to $t_x, t_y \in [-5\,\mathrm{mm}, 5\,\mathrm{mm}]$ and $t_z \in [-1.2\,\mathrm{mm}, 1.2\,\mathrm{mm}]$. Similarly, we limit rotations to $\alpha_{rot}, \beta_{rot}, \gamma_{rot} \in [-10°, 10°]$. For training, we rescale the regression outputs to a range of $[0, 1]$. In particular, we rescale every output component individually based on the training set. For evaluation we transform the network's predictions back to the original scale and calculate error metrics on those values.

Another question that we address is whether training a single CNN for the entire pose label is the optimal choice. Multi-output regression has been addressed both by training a single model for the entire output and by training individual models for each output [55]. We study three different approaches. First, we train a single CNN to predict the complete 6D pose. Second, we train one CNN each for position and orientation prediction. Third, we train one CNN each to predict a single component of the pose vector. We choose the best performing approach for all other experiments.

The 3D CNN implementation is based on the TensorFlow environment [1], and training is performed with graphics cards of type NVIDIA GTX 1080 Ti with 11GB VRAM.

Visualization. Understanding and visualizing what CNNs learned after training is an important issue in the field of deep learning. In particular, for the problem at hand, it is crucial to understand what kind of image properties the CNNs leverage for pose estimation. In general, CNNs for classification are either visualized by image generation through maximization neuron activations or with saliency maps [583]. We utilize the latter since activation maximization is not immediately applicable to regression with continuous output values. Saliency maps visualize which region in a particular input image has the largest influence on a certain activation in the network. This is achieved

by computing the partial derivative of the activation with respect to the current input image, leading to a gradient image

$$g_{x,y,z} = \sum_i \frac{\partial \mathrm{y}_i}{\partial \mathrm{x}_{x,y,z}} \tag{6.1}$$

where $g_{x,y,z}$ is the saliency map, $\mathrm{x}_{x,y,z}$ is the input image, and y is a vector of activations. The partial derivatives for each vector element y_i are summed up to form the saliency map. We set y to be the output of our network, and thus, a saliency map tells us which region of an image leads to the most substantial change in the output. This allows us to visualize what our CNN focuses on when being trained on 2D data, when being trained on the marker with a surface structure and when being trained on a marker with inner features.

To enhance the saliency maps, we utilize guided backpropagation [463]. The key idea of this approach is to combine normal backpropagation with the deconvolution idea of Zeiler et al. [583]. Effectively, guided backpropagation changes the backward pass of the ReLu activation function such that negative gradients and thus components that reduce the target activation are suppressed. The method has been shown to perform better than normal backpropagation and deconvolutional visualization, for details see [463].

Experiments. We provide the results of the analysis of our pose estimation method in several steps:

1. We show results for our choice of splitting position and orientation learning.
2. We show pose estimation accuracy for 2D depth representations for 2D CNN training and 3D volumes for both 2D and 3D CNN training. Again, we employ Inception3D with a 2D counterpart for this comparison. We use marker A for this comparison. The marker is best suited for comparison with 2D depth representations as it largely shows surface information in OCT volumes.
3. We show how marker A compares to marker B in order to highlight the effects of the inner marker structure for 3D CNN learning. We use Inception3D for this comparison.
4. We visualize what our 3D CNN learns using saliency maps. This adds qualitative results and a better understanding of the previous, quantitative results.
5. We show the suitability of our method for online pose estimation by providing inference times for 2D and 3D CNN data processing.
6. We show our method's robustness by using our Inception3D model for a dataset with heavy occlusion.
7. We compare the 3D CNN models with respect to their performance for our pose estimation method. We use both markers for this comparison.

We evaluate pose estimation performance using the MAE, rMAE, and PCC. We test for statistically significant differences in the absolute errors with the Wilcoxon signed-rank test and a significance level of α = 5%.

Tab. 6.13: MAE (position in µm, orientation in °), rMAE and PCC for position and orientation prediction when training on position and orientation separately or simultaneously. 6D label refers to training with the entire pose as the network output. 3D label refers to the training of two separate networks for position and orientation. 1D label refers to the training of six networks on one part of the pose label each. The best category is marked bold. We used the IN3D model and marker B for this experiment.

	Position		
	MAE	rMAE (10^{-3})	PCC (%)
6D Label	25.32 ± 15	29 ± 24	99.91
3D Label	$\mathbf{14.89 \pm 9}$	$\mathbf{18 \pm 14}$	**99.96**
1D Label	15.88 ± 12	19 ± 15	**99.96**

	Orientation		
	MAE	rMAE (10^{-3})	PCC (%)
6D Label	0.099 ± 0.06	173 ± 15	**99.96**
3D Label	$\mathbf{0.096 \pm 0.07}$	$\mathbf{168 \pm 16}$	**99.96**
1D Label	0.119 ± 0.12	21 ± 20	99.93

Results

Learning Target. First, we investigate the effect of training different models for different parts of the target pose vector. The results for three approaches with different label splitting are shown in Table 6.13. For position prediction, splitting up the training improves performance. However, training on a single position output does not lead to improvement. For orientation prediction, removing the position part does not have a substantial effect. Splitting the labels up further, even deteriorates performance. Based on these observations, we choose to train position and orientation separately.

Data Representations. As a second step, we compare the accuracy when using 2D depth representations or full volumetric data for learning. The results are shown in Table 6.14. We used our IN3D architecture. For the 2D representations, we removed the filter's third dimension, resulting in IN2D. We conducted the experiment with marker A. This marker largely shows surface structures in OCT volumes. Therefore, 2D depth maps could be expected to contain a similar amount of information for learning.

Considering the comparison between 2D and 3D, the volumetric data representation that is used for training Inception3D clearly outperforms all 2D approaches. Note that the 2D CNN version has a smaller capacity since filters only cover two dimensions. However, the 2D CNN was always able to reach a similar training error. This shows that insufficient capacity cannot be the reason for the performance difference but rather the representations used for learning.

Out of all models with 2D filters, the model with volume inputs performs best. Here, volume data is processed in a z-slice-wise fashion with $3 \times 3 \times 1$ kernels while also taking neighboring slices into account.

Considering the difference between 2D representations, it is notable that a combination of depth and intensity information from a single MIP in z-direction performs best.

Tab. 6.14: MAE (position in µm, orientation in °), rMAE and PCC for position and orientation prediction for 2D representations with a 2D CNN in comparison to volumetric data with a 3D CNN. The best performing model is marked bold. All models are based on IN3D, for the 2D cases, the third dimension of filters is omitted. Marker A was used for this experiment.

	Position		
	MAE	rMAE (10^{-3})	PCC (%)
$\boldsymbol{\mathcal{V}}$	$\mathbf{23.65 \pm 16}$	$\mathbf{28 \pm 24}$	**99.86**
$\mathcal{M}^1$	46.16 ± 33	61 ± 47	99.31
$\mathcal{M}^3$	81.67 ± 42	89 ± 62	98.84
$\mathcal{D}^1$	58.32 ± 38	73 ± 50	99.12
$\mathcal{D}^3$	224.9 ± 155	182 ± 141	95.64
MD	43.45 ± 32	57 ± 45	99.46
$\mathcal{V}$ (2D)	28.84 ± 19	34 ± 28	99.80

	Orientation		
	MAE	rMAE (10^{-3})	PCC (%)
$\boldsymbol{\mathcal{V}}$	$\mathbf{0.268 \pm 0.22}$	$\mathbf{47 \pm 52}$	**99.75**
$\mathcal{M}^1$	0.741 ± 0.62	129 ± 118	98.28
$\mathcal{M}^3$	0.755 ± 0.65	132 ± 124	98.25
$\mathcal{D}^1$	0.763 ± 0.70	133 ± 115	97.65
$\mathcal{D}^3$	0.828 ± 0.72	145 ± 131	97.57
MD	0.597 ± 0.43	104 ± 89	98.84
$\mathcal{V}$ (2D)	0.290 ± 0.24	51 ± 44	99.73

Moreover, the single-channel representations that only leverage information from the z direction perform better than representations with additional x-z and y-z projections.

Inner Structure. Next, we show how a recognizable inner structure affects learning for 3D CNNs. Their key difference is that one marker has an opaque surface under infrared light (A), while the second marker has a visible inner structure in OCT images (B), see Figure 6.17. The results are shown in Table 6.15. Marker B clearly outperforms marker A. It is notable, that the position error goes beyond the assumed ground-truth label accuracy, induced by the robot's specified repeatability of $\pm 20\,\mu\mathrm{m}$.

Visualization. Next, we aim for a deeper understanding of what was learned by the 3D CNN. In particular, we investigate whether the 3D CNN leveraged the depth information given in the second marker. Saliency maps indicate which region in the image is largely responsible for the output, i.e., a change in that region leads to the largest change in the output.

To emphasize the importance of depth exploitation, we compare the 3D saliency maps from the two markers with 2D saliency maps from the approach of leveraging depth information from MIPs. The results for this are shown in Figure 6.20. The saliency maps for the 2D CNN show high intensities at characteristic surface features on the markers. The 3D saliency maps for the 3D CNN, which are represented by 2D MIPs, focus on a

Tab. 6.15: MAE (µm for position, ° for orientation), rMAE, and PCC for position and orientation prediction for the marker with surface structure (A) compared to the marker with a depth structure (B). The best category is marked bold. We used the Inception3D model for this experiment.

	Position		
	MAE	rMAE (10^{-3})	PCC (%)
Marker A	23.65 ± 16	28 ± 24	99.86
Marker B	$\mathbf{14.89 \pm 9}$	$\mathbf{18 \pm 14}$	**99.96**

	Orientation		
	MAE	rMAE (10^{-3})	PCC (%)
Marker A	0.268 ± 0.22	47 ± 52	99.75
Marker B	$\mathbf{0.096 \pm 0.07}$	$\mathbf{17 \pm 16}$	**99.96**

Tab. 6.16: Inference times (mean and standard deviation in ms) for IN3D and IN2D with 2D and 3D input data. The values are calculated based on 100 passes of a single sample through the network.

	Inference Time
IN3D	20.95 ± 1.05
IN2D	18.89 ± 1.67
IN2D (Vol.)	19.12 ± 1.59

region on the marker without sticking to specific surface features such as the pyramid tip. Note, that the same original test image was used for the 2D saliency maps and the 2D MIPs of the 3D saliency maps.

Furthermore, we present the saliency maps of two test images for the two markers in Figure 6.21. The saliency maps are shown in red as slices overlaid on top of slices of the test images. The cross-sectional view specifically shows what regions on and inside the marker have a large influence on the output. For the marker with a surface structure, the saliency map mostly lights up around the marker's surface. Note that the high-intensity saliency area spans above and below the surface, covering 3D space. For the marker with a depth structure, higher values in the saliency maps can be observed inside the marker. Furthermore, it should be noted that the 3D CNN's center of attention is indeed the marker itself. There appears to be no fitting on the ground surface or artifacts within the volume.

Inference Time and Robustness. Furthermore, we consider properties that are relevant for application, including inference times and robustness. The results for inference time measurement are shown in Table 6.16. We can observe that both CNNs allow sample processing at 50 Hz with the 2D CNNs being slightly faster. Note that the convolution operations only have a small influence with a total number of 68 out of 1734 operations and an average processing time of 0.065 ms for IN3D and 0.046 ms for IN2D. Also, note that these values are very hardware and software dependent.

Furthermore, we investigate how well our model performs when the OCT volume is

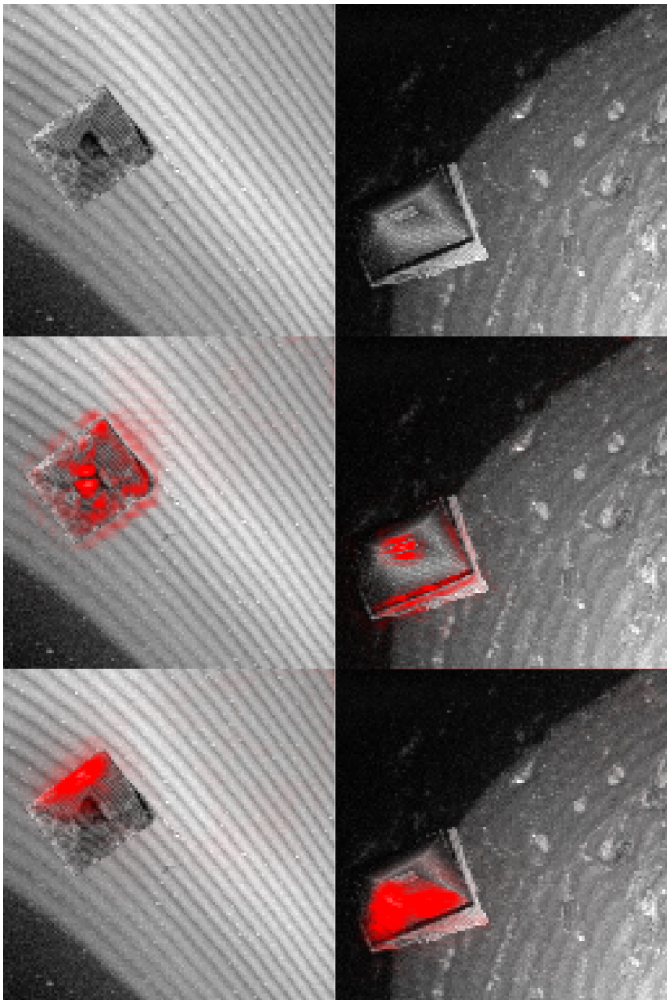


Fig. 6.20: Comparison of saliency maps for the 2D and 3D data representations. Left, images of marker B are shown, right, images of marker A are shown. At the top, 2D MIPs along the axial z-direction are shown. In the middle, 2D saliency maps of the 2D training approach are shown in red, overlaid on the original input image. At the bottom, 2D MIPs of the saliency maps are shown in red for the 3D CNN (IN3D) that was trained on volume data. Here, MIPs of the volumetric saliency maps are overlaid on the input image's MIP.

occluded with foreign objects. For this purpose, we use our third dataset, where different objects are placed around the marker during data acquisition. The results are shown in Table 6.17. The model's performance is still close to our other datasets, where mostly the marker itself was visible. For rotations, the performance deteriorates more.

3D CNN Architectures. For our deep learning framework, we employ our four proposed models for 3D data that come with different improved architectural ideas. The results for position training are shown in Table 6.18. With the most structural adjustments, IN3D outperforms the other models. Furthermore, RN3D-A, which uses the type of residual connections often employed for 3D CNNs [339, 575], lacks behind more significantly.

Additionally, Figure 6.22 shows the training behavior over time for all four models.

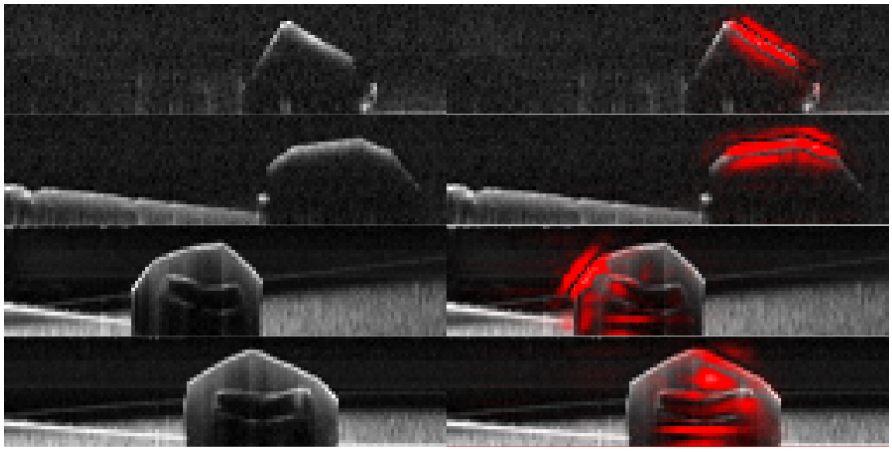


Fig. 6.21: Visualization of what the 3D CNN focuses on using saliency maps. Left, two lateral slices through each marker are shown. The top two images show marker A, the bottom two show marker B. Right, slices through the 3D saliency maps for each marker are shown, which are overlaid on the input image slices. The saliency maps show which region in the image shown on the left has the most substantial influence on the CNN's output. The key difference between the markers' saliency maps is the focus on the marker's surface and inner structure, respectively. Both images and saliency maps are originally volumetric.

In terms of convergence behavior, all models perform similarly, as all models have approximately the same number of parameters.

Long-range Connections. Next, we present results on how long-range residual connections that span over modules affect performance.

We present two types of long-range connections which are frequently used for feature transfer between similar-sized stages in 3D CNNs for segmentation. We extend this approach by drawing connections between different stages of the network and introduce the concept to IN3D by creating long-range connections between modules. In Table 6.19 the results for the use of residual connections, feature connections and no connections at all are shown. Note, that the use of long- and short-range residual connections is also referred to as mixed residual connections [575] and feature connections are also called dense connections [208]. Residual connections perform best, closely followed by feature connections. The model with no connections at all shows worse results compared to models with long-range connections. It should be noted that performance changes are small compared to using an entirely different architecture.

In Figure 6.23 the training behavior of the three model variations is shown. There is a clear difference in errors for the model without any connections, while the two models with connections are very close. The convergence behavior of the models is very similar once again. It should be noted that introducing the long-range connections leads to a negligible increase in parameters.

Tab. 6.17: MAE (µm for position, ° for orientation), rMAE, and PCC for position and orientation prediction with our occlusion dataset compared to the normal dataset. Marker B and IN3D were used for this experiment.

	Position		
	MAE	rMAE (10^{-3})	PCC (%)
Normal	14.89 ± 9	18 ± 14	99.96
Occlusion	16.62 ± 8	20 ± 15	99.96

	Orientation		
	MAE	rMAE (10^{-3})	PCC (%)
Normal	0.096 ± 0.07	17 ± 16	99.96
Occlusion	0.187 ± 0.09	40 ± 32	99.88

Discussion

We provide extensive results for our method of 6D pose estimation from volumetric OCT data, which leads to interesting insights for deep learning-based pose estimation, OCT data representations, and deep learning models.

6D pose estimation from OCT volumes with deep learning models is a novel approach. The approach appears to be promising as we generally achieve a PCC very close to 1, suggesting highly accurate estimates. Also, note that the position MAE is within the magnitude of the robot's repeatability, and thus the ground-truth labels. Therefore, our deep learning approach is likely limited by the labels' accuracy and not a lack of representational power. In addition, our framework is general enough to be employed for various pose estimation problems as the source of labels can be any robot or motor.

Furthermore, we investigate how splitting up training for different parts of the pose affects performance with significant improvement being observed when training only on positions, as shown in Table 6.13. Often, multi-output regression is addressed by training a single model with multiple outputs instead of using multiple models with single outputs [55]. This approach promises better performance by introducing regularization through additional supervision. The model's feature maps have to learn to represent features for all outputs simultaneously. However, we observe performance improvement for position learning when splitting the pose label. This effect can be explained by regularization through learned invariance. When training on positions only, the input data contains examples with the marker being in the same position with different orientations. Thus, the CNN's weights are forced to learn invariance towards orientation. This is linked to OCT's properties as light scattering, and surface visibility is highly dependent on the light beam's angle of impact. Therefore, invariance towards orientations also implicitly enforces invariance towards different light scattering properties in the data. Our results indicate that the effect of learned invariance significantly improves position learning. At the same time, there are no significant performance differences for orientation learning. Shifting positions within the volume does not change the OCT's light beam angle of impact. Therefore, in opposite to position learning, invariance towards positions for rotation learning does not implicitly enforce invariance towards different light scattering

Tab. 6.18: MAE (µm), rMAE (10^{-3}) and PCC for position prediction with four different 3D CNN architectures, see Section 4 for a detailed description. The best model is marked bold.

	Marker A		
	MAE	rMAE (10^{-3})	PCC (%)
IN3D	$\mathbf{23.65 \pm 16}$	$\mathbf{28 \pm 24}$	**99.86**
RX3D	26.87 ± 19	31 ± 28	99.84
RN3D-B	29.56 ± 23	36 ± 39	99.73
RN3D-A	39.18 ± 44	44 ± 49	99.62

	Marker B		
	MAE	rMAE (10^{-3})	PCC (%)
IN3D	$\mathbf{14.89 \pm 9}$	$\mathbf{18 \pm 14}$	**99.96**
RX3D	16.28 ± 10	21 ± 16	99.94
RN3D-B	17.68 ± 11	22 ± 18	99.93
RN3D-A	21.71 ± 11	27 ± 21	99.91

conditions. All in all, our training strategy with split labels improves position learning by taking advantage of domain knowledge on OCT's light-scattering properties.

We investigate **2D depth information and volume data** to draw a connection to OCT-based tracking, which has been performed on 2D projections [276]. The use of 2D depth representations can be motivated by the imaging property that many surfaces appear opaque under OCT as they cannot be penetrated by infrared light. Therefore, pure surface information extracted from the OCT volume could be deemed sufficient for most tasks.

However, our results in Table 6.14 show that moving towards volumetric data and 3D CNNs significantly increases performance. The use of volume data with flat 2D kernels already improves performance, which indicates that a significant amount of information is lost when creating 2D projections. The novel approach of employing 3D CNNs for OCT volume data improves performance even further. The volumetric receptive fields of stacked 3D convolutional layers appear to be able to capture relevant features for pose estimation more effectively. With these findings, we motivate the use of full volumetric information for OCT-based tracking and pose estimation frameworks that relied on 2D representations so far [72, 276]. Other OCT-based deep learning methods that have also relied on 2D representations so far [412, 510, 531] could also benefit from our insights.

We highlight the improved feature learning further with the use of saliency maps for 2D and 3D data, see Figure 6.20. For 2D data, the CNN appears to fit to distinct features on the marker surface that are visible in the 2D representation. The 3D CNN, however, appears to take advantage of other, more buried features that cannot be recognized on the surface. This leads to our investigation of deep subsurface feature learning.

Markers with surface and subsurface structure are compared to gain further insight on how 3D CNNs take advantage of inner features. Our results in Table 6.15 show that the marker with an inner structure performs significantly better than the marker that

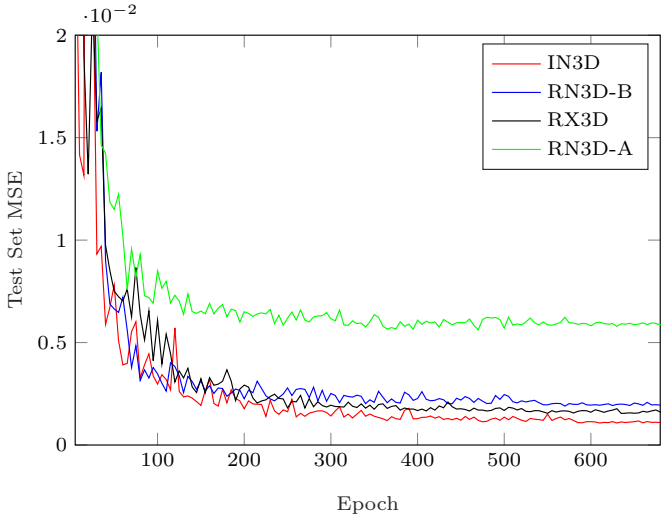


Fig. 6.22: Comparison of test errors for all four architectures we introduce. The test set MSE during training is shown. The training behavior for the models being trained on marker A is shown.

largely contains surface information in OCT images. This shows that the exploitation of OCT's 3D nature can be advantageous for volumetric feature learning with 3D CNNs. We support these quantitative results with additional saliency maps, see Figure 6.21. They highlight that the 3D CNNs indeed learned to exploit subsurface information when it was present in the volume data.

This finding shows that we can improve pose estimation performance without using a larger, more sophisticated marker. Ultimately, markers for surgery should be small and non-disruptive. Creating subsurface structures is an elegant solution to increase the learnable feature space without increasing the marker size. Thus, we combine the advantage of OCT's depth imaging with 3D CNN-powered volumetric feature learning for pose estimation.

All in all, these insights emphasize once more, that OCT's capability of producing volumetric information is very exploitable by 3D CNNs. We provide strong evidence that OCT-based 2D slicing and projection methods [412, 510, 531] could significantly benefit from 3D data usage and volumetric feature exploitation.

Moving towards clinical application scenarios is the next step for our method. We highlight its suitability for future clinical use by showing its real-time processing capability and its robustness towards occlusion.

Regarding the processing times shown in Table 6.16, it is notable that the change between 2D and 3D convolutions does not lead to a significant difference. The largest processing overhead is caused by other operations that are always present in the network, and neither the input size nor the different operations are a bottleneck. Therefore, our 3D CNNs are capable of online pose estimation. This is linked to our efficient 3D CNN architecture design with comparatively small numbers of parameters, as shown in Table 6.12.

For future applications in clinical scenarios, our marker system should be capable of being integrated into existing OCT setups for MIS without requiring special operating

Tab. 6.19: MAE (µm), rMAE, and PCC for position prediction with different types of long-range connections. Residual refers to long-range residual connections, Feature-Based refers to long-range feature concatenation and None indicates no use of such connections. The best model is marked bold.

	Marker A		
	MAE	rMAE (10^{-3})	PCC (%)
Residual	$\mathbf{23.65 \pm 16}$	$\mathbf{28 \pm 24}$	**99.86**
Feature-Based	23.99 ± 17	30 ± 29	99.83
None	27.17 ± 22	33 ± 39	99.72

	Marker B		
	MAE	rMAE (10^{-3})	PCC (%)
Residual	$\mathbf{14.89 \pm 9}$	$\mathbf{18 \pm 14}$	**99.96**
Feature-Based	15.29 ± 10	21 ± 16	99.94
None	19.53 ± 11	25 ± 19	99.92

conditions. Thus, it is crucial that our models deal well with unknown objects. Our occlusion dataset results in Table 6.17 show that our IN3D model was able to learn robustness towards new occluding objects by achieving a performance close to the initial dataset.

The application of **deep learning architectures for 3D OCT data** is a novel approach. When entering new problem domains with the use of deep learning, it is mostly unclear how existing models should be adopted [554]. Therefore, we created four different 3D CNN architectures with different design principles and showed how they affect performance for our novel learning problem.

In particular, the idea of downsampling intermediate network outputs with respect to their number of feature maps with bottlenecks appears to improve representational power significantly. The only model without this property, RN3D-A, performs significantly worse than the other models, see Table 6.18. The bottleneck idea has been successful for 2D CNNs [192], and we show that it is even more valuable for 3D CNNs. Bottlenecks address the key problem of model complexity and computational cost, which are particularly severe for 3D CNNs [575]. The increased efficiency in terms of the number of parameters allows for much deeper models. This insight relates to Yu et al. [574] who built very deep 2D CNNs for medical image analysis by relying on downsampling in the feature map dimension.

In addition to the bottleneck principle, we use IN3D and RX3D to address 3D CNN architecture design for our problem by showing the pay-off for extensive design and fine-tuning. Both architectures employ the successful principle of multiple paths at each scale [475]. However, for IN3D, we carefully tuned each path individually, while for RX3D, all paths are designed identically. Although there is a performance difference, it is notable that the simple design principles we followed for RX3D lead to similar performance, see Table 6.19. As a result, we argue that high-effort custom designs such as our IN3D might not be strictly necessary for practice as more simple design

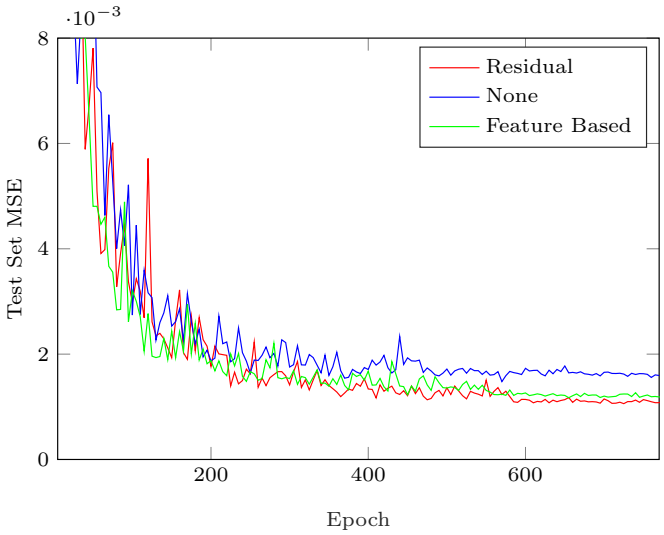


Fig. 6.23: Comparison of test errors for three different types of long range connections. Residual refers to long-range residual connections, Feature-Based refers to long-range feature concatenation, and None indicates no use of such connections. The test MSE during training is shown. The shown results are for training on marker A.

choices can already reach excellent performance. Still, if the goal is the best performance possible, extensive fine-tuning will be necessary when entering new problem domains, such as ours, with 3D CNNs.

Additionally, we introduce long-range feature transfer between different scales for our architecture. This extends the idea of Ronneberger et al. [409] and Yu et al. [575] who employed feature transfer between similar scales for segmentation tasks. As shown in Table 6.19, these connections do lead to an improved performance. This supports the idea that we both need to detect our marker in the full image, which requires high level, coarse features with a large implicit FOV, and we also need to detect fine-grained differences for accurate pose distinction. The combination of fine, local, and coarse, global features appears to lead to better pose estimation performance. This insight is in line with related ideas for object detection, where features are also transferred for a combination of local and global properties [448].

Since the 3D CNN architectures we use are all very generic, our results have broader implications. In particular, it should be noted that the design principles of downsampling in the number of feature maps and multi-scale feature extraction are still rarely found in 3D medical image analysis. Early 3D CNN architectures have already been criticized for lack of representational capabilities [575]. We extend on this point and argue that the design principles that we brought to the 3D domain with IN3D and our other models are insufficiently applied for 3D medical learning problems. Several 3D CNN architectures with effective designs have been successfully introduced to the 3D image domain [81, 112, 231, 575]. However, we argue that these well-designed architectures could benefit further from the efficiency-focused design principles we introduced to 3D. Based on our results, we see significant potential in current 2D CNN architectures for the 3D medical imaging domain.

Summary

We propose a framework for deep learning-based pose estimation with 3D OCT volumes. A robotic setup allows for automatic data acquisition and annotation for studying the properties of deep learning-based pose estimation with OCT. With respect to our first research question on data representations, we find that using full 3D volumes is preferable over 2D intensity projections and depth maps. Qualitative saliency maps confirm that subsurface 3D features are effectively exploited by CNNs using 3D data. Regarding our second research question on deep learning models, we find that carefully engineered and efficient CNNs are well suited for an extension from 2D to 3D, where real-time capabilities are important. In particular, bottlenecks and multi-path design patterns appear to be important for efficient architectures. Also, long-range connections enable effective feature reuse within the models, leading to improved performance.

6.6 3.5D and 4D Data: OCT-Based Motion Estimation

A problem closely related to OCT-based pose estimation is the problem of OCT-based motion estimation. Instead of estimating a single state, the goal is to estimate the *change* that occurred between two states. For computer-assisted interventions with OCT, motion estimation is relevant for problems such as automated FOV adjustment or surgical tool adjustment in the context of cochleostomy [45]. Recently, deep learning methods have been introduced as an alternative to traditional, registration-based motion estimation, for example, using phase correlation [428].

Using just a template and moving image for motion estimation can be problematic if motion between the original template and the current state is huge as the overlap between the images becomes small. Modern OCT systems could overcome this problem by acquiring entire sequences of OCT volumes, following the motion trajectory, as very high acquisition rates have been achieved [527]. Therefore, more information can be made available between an initial state and the current state, which could be useful for motion estimation. While deep learning approaches using two images could follow the trajectory with pair-wise comparisons, we hypothesize that processing an entire sequence of OCT volumes at once might provide more consistency and improved motion estimation performance. The concept is shown in Figure 6.24.

Therefore, we compare several deep learning methods and investigate whether using 4D spatio-temporal OCT data can improve deep learning-based motion estimation performance compared to using just a template and a moving image. A template and moving image volume can be treated as 3.5D data, which is well-suited for our Siamese CNN architectures we introduced in Section 4.2. For processing a full 4D sequence, we employ our different 4D spatio-temporal deep learning approaches, which includes both the different 4D CNN variants and convolutional-recurrent models we presented in Sections 4.1.4 and 4.3, respectively. Also, we study how a temporal regularization technique at the model output affects performance. Furthermore, we conduct experiments with respect to robustness towards motion artifacts and rotation distortions.

This concept also opens up the option motion *forecasting*. Given a sequence of OCT volumes that was acquired along a smooth trajectory, extrapolation to future motion vectors might be feasible. This could be particularly useful for tasks such as motion compensation, where a mechanical system induces a lag that needs to be compensated by forecasting. Thus, we also consider the task of predicting both the motion vector of the current trajectory and future motion vectors.

Similar to our OCT-based pose estimation framework, we propose an automated data acquisition setup that allows us to acquire both image sequences and corresponding ground-truth motion vectors. This enables us to build large datasets for supervised training and an in-depth analysis of deep learning-based motion estimation with OCT.

Summarized, with respect to our research question on data representations, we study how the use of 3.5D data compares to the use of full 4D spatio-temporal sequences. Regarding our research question on deep learning models, we employ our proposed 4D spatio-temporal deep learning models. Furthermore, we use our Siamese CNN architectures for 3.5D and explore its application to 4D data by pair-wise comparisons.

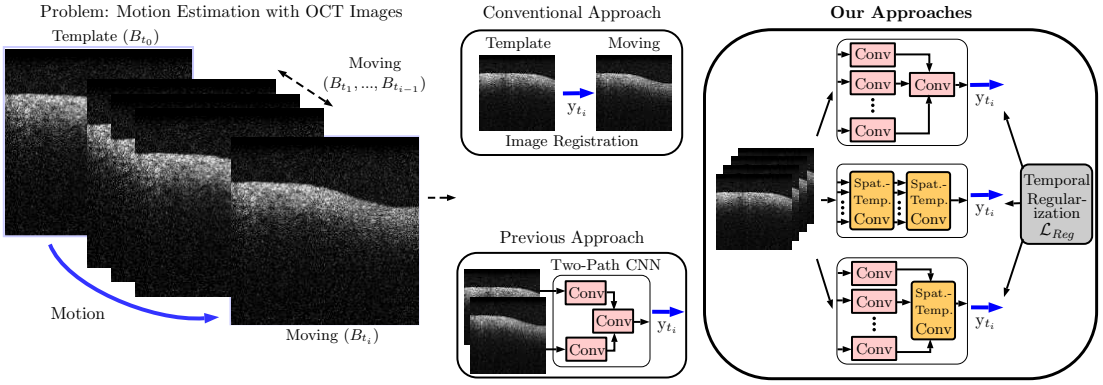


Fig. 6.24: Our approach for motion estimation in comparison to previous methods. The approach is illustrated for 2D OCT B-Scans B_{t_i} for simplicity. Note, we perform all experiments with 3D volumetric OCT images C_{t_i} and thus 4D spatio-temporal data.

Methods and Datasets

Problem Definition. We address the problem of predicting a target motion vector $\mathrm{y}_{t_i} \in \mathbb{R}^3$ between a volume C_{t_0} and C_{t_i} with $C_{t_i} \in \mathbb{R}^{n_h \times n_w \times n_d}$. The 4D spatio-temporal sequence $T_{t_i} = \{C_{t_0}, ..., C_{t_i}\}$ of size $T_{t_i} \in \mathbb{R}^{n_t \times n_h \times n_w \times n_d}$ consists of 2 up to n_t volumes. This extends to the problem of forecasting by trying the find a mapping to a motion vector sequence $\mathrm{Y}_{t_f} = \{\mathrm{y}_{t_i}, ..., \mathrm{y}_{t_f}\}$ of size $\mathrm{Y}_{t_f} \in \mathbb{R}^{n_f \times N_r}$, which represents both the current target motion vector y_{t_i} and n_f future target motion vectors. Thus, we try to find deep learning models $f_M : \mathbb{R}^{n_t \times n_h \times n_w \times n_d} \rightarrow \mathbb{R}^{n_f \times N_r}$.

Experimental Setup. We employ a setup that allows for automatic data acquisition and ground-truth annotation, see Figure 6.25. We use a commercially available swept-source OCT device (OMES, OptoRes) with a scan head, a second scanning stage with two mirror galvanometers, lenses for beam focusing, and a robot (ABB IRB 120). A chicken breast sample is attached with needles to a holder on the robot. Our OCT-setup allows for shifting the FOV without moving the scan head by using the second mirror galvanometers stage, and by changing the pathlength of the reference arm. Two stepper motors control the mirrors of the second scanning stage, which shift the FOV in the lateral directions. A third stepper motor changes the pathlength of the reference arm to translate the FOV in the axial dimension. We consider volumes of size $32 \times 32 \times 32$ with a corresponding FOV of approximately $5\,\mathrm{mm} \times 5\,\mathrm{mm} \times 3.5\,\mathrm{mm}$.

Data Acquisition. To assess our methods' generalization on different tissue regions, we consider 40 randomly chosen ROIs of a chicken breast sample with the same size as the OCT's FOV.

For motion estimation, only the relative movement between the FOV and ROI is relevant, hence moving the ROI and using a steady FOV is equivalent to moving the FOV and using a steady ROI. This can be exploited for the generation of both OCT images and ground-truth labels. By keeping the ROI steady and moving the FOV by a defined shift in stepper motor space, we simulate relative ROI movement. At the same time, the defined shift provides a ground-truth motion as we can transform the shift in motor space to the actual motion in image space using a hand-eye calibration.

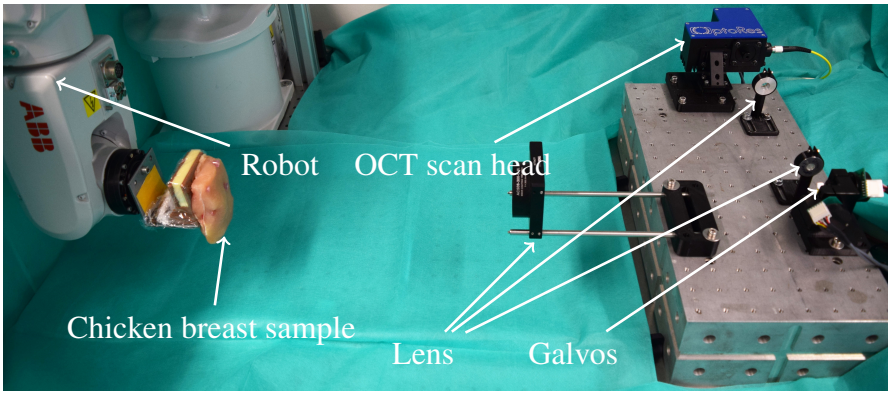


Fig. 6.25: The experimental setup for data acquisition and annotation. The chicken breast sample is attached with needles to a holder of the robot. The OCT device itself is not shown.

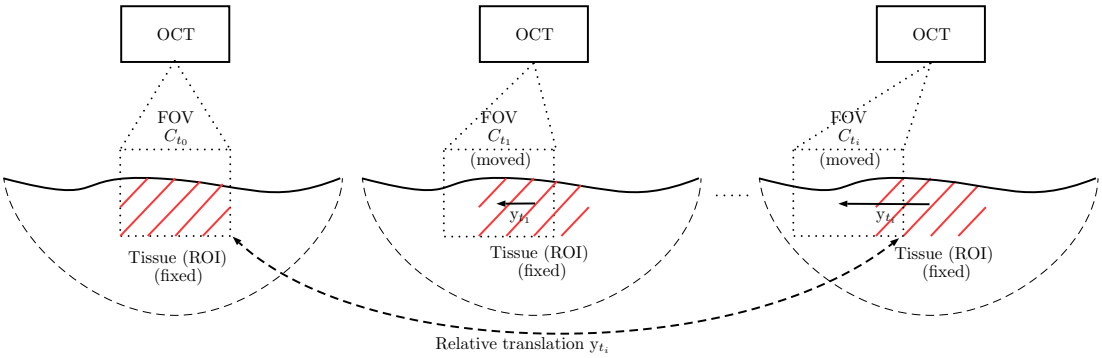


Fig. 6.26: Our data acquisition strategy. For motion estimation, only the relative movement is relevant, hence, we use a fixed ROI and move the FOV step-wise by $y_{t_i} - y_{t_{i-1}}$. This results in a sequence of OCT volumes T_{t_i} with the corresponding relative translation y_{t_i} between the initial volume C_{t_0} and the last volume C_{t_i} of a sequence.

Initially, the FOV completely overlaps with the target ROI. After acquiring an initial template image volume C_{t_0} of the ROI, we use the stepper motors to translate the FOV by y_{t_1} such that the target ROI only partially overlaps with the FOV. Now, we acquire an image volume C_{t_1} for the corresponding translation y_{t_1}. This step can be repeated multiple times, resulting in a sequence of shifted volumes C_{t_i} and known relative translations y_{t_i} between the initial ROI and a translated one. Note, each translation y_{t_i} is relative to the initial position of an ROI. The procedure is shown in Figure 6.26.

In this way, we obtain our supervised learning problem where we try to learn the relative translation y_{t_i} of an ROI experiencing motion with respect to its initial position, given a sequence of volumes $T_{t_i} = \{C_{t_0}, ..., C_{t_i}\}$. For forecasting, we also consider the following motion vectors $y_{t_{i+1}}$ and $y_{t_{i+2}}$, forming the motion vector sequence $Y_{t_f} = \{y_{t_i}, y_{t_{i+1}}, y_{t_{i+2}}\}$ with $n_f = 2$ time steps being forecasted.

For generation of a single motion trajectory, we consider a sequence of five target translations with target motor shifts $y_t = \{y_{t_0}, y_{t_1}, y_{t_2}, y_{t_3}, y_{t_4}\}$. To generate a smooth motion pattern, we randomly generate y_{t_4} and use spline interpolation between $y_{t_0} = \{0, 0, 0\}$,

y_{t_4}, and a randomly generated connecting point y_{t_c}. We sample the intermediate target shifts $\mathrm{y}_{t_1}, \mathrm{y}_{t_2}, \mathrm{y}_{t_3}$ from the spline function. This results in various patterns, where the FOV drifts away from the ROI. By using different distances between y_{t_0} and y_{t_4}, we simulate different magnitudes of motions (velocities) and obtain various different motor shift distances between subsequent volumes. Example trajectories are shown in Figure 6.27. We use a simple calibration between galvo motor steps and image coordinates to transform the shifts from stepper motor space to image space, resulting in a shift in millimeters.

For data acquisition, we follow three steps. First, we use the robot for randomly choosing an ROI. Then, the initial state of the three motors corresponds to a FOV completely overlapping with the ROI. Second, we randomly generate a sequence of five target motor states, which shifts the FOV out of the ROI. Third, at each of the target motor states, an OCT volume is acquired. Overall, for each ROI, we acquire OCT volumes of 200 motion patterns, where each movement consists of five target translations and five OCT volumes.

Moreover, we evaluate how the estimation performance is affected by relative rotations between volumes of a sequence. Note that our scanning setup is designed for translational motion, as rotation is difficult to perform using galvo mirrors. Therefore, we add rotations in a post-processing step by rotating acquired volumes of a sequence T_{t_i} around the axial axis. We define a maximal rotation α_{max} and transform each volume of a sequence with $\widetilde{T}_{t_i} = R(\alpha^i_{rot})T_{t_i}$ while $\alpha^i_{rot} = \frac{\alpha_{max}}{4} \cdot i,\ \forall i \in [0, 4]$. $R(\alpha^i_{rot})$ is the rotation matrix along the axial axis. First, we consider rotations as noise that is applied to the image data. Second, we incorporate the rotation into our motion and adapt the ground-truth with respect to the rotation.

Last, we also consider the effect of fast and irregular motion, such as high-frequency tremors that may cause distortion within an image. This effect is unlikely to occur with our current setup as our high acquisition frequency prevents common motion artifacts [579]. Nevertheless, we perform experiments with simulated motion artifacts due to relevance for slower OCT systems. We follow the findings of previous works [261, 559, 579], and consider motion distortions as lateral and axial shifts between B-scans of an OCT volume that has been acquired without motion distortions. In this way, we are able to augment our data with defined motion distortions in a post-processing step. To simulate different intensities of motion distortions, we introduce a factor p_{shift} that defines the probability that a B-scan is shifted. Also, we compare shifting the B-scans one or two pixels randomly along the spatial dimensions.

Deep Learning Models. All deep learning model consist of an initial processing block and a module block, following the general structure introduced in Section 4.1.2. Inside this general structure, we implement several different architecture concepts that we introduced in Chapter 4. In each model we use a DenseNet module block. Each module block consists of three DenseNet blocks connected by average pooling layers. Each DenseNet block consists of two convolutional layers with a growth rate of $g_k = 10$. At each model's output, we use a GAP layer for connecting the linear regression output layer.

First, we implement *TP-DN3D*, a version of our Siamese architecture with two initial processing paths that we introduced in Section 4.2. We use feature concatenation at the model's fusion point. Furthermore, we use three CNN layers for the initial two-path part

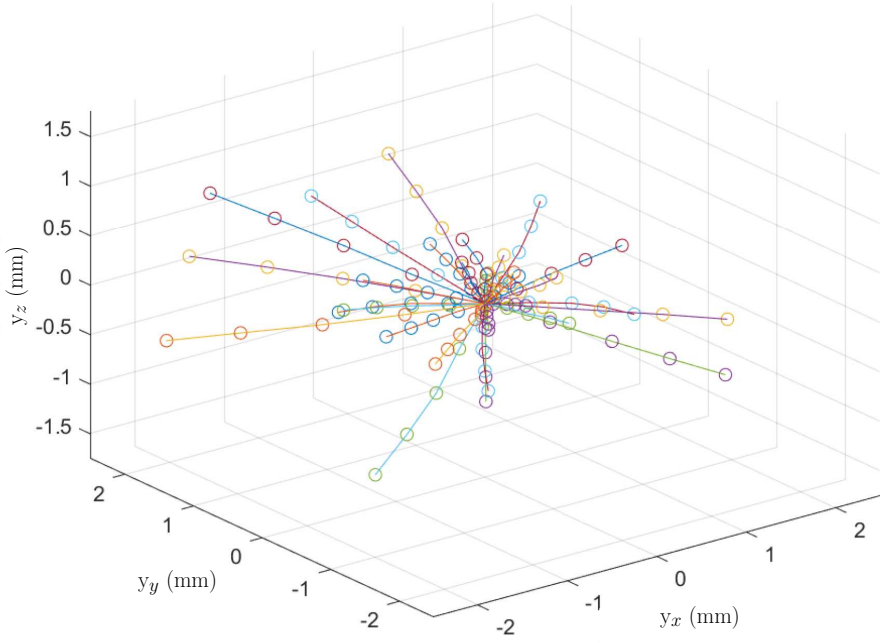


Fig. 6.27: We show 30 example trajectories for the translations in the spatial dimensions. Each trajectory consists of a sequence of five target shifts y_{t_i} (circle).

and our DenseNet module block with 3D convolutions after the concatenation point. In the first instance, we only consider the initial volume C_{t_0} and the last volume C_{t_i} of a sequence to estimate the relative translation.

Second, we use TP-DN3D and consider predicting the relative translation between the initial and last volume, based on the vector sum of the relative translations between two subsequent volumes of a sequence. In this way, the model obtains information from the entire sequence. The model receives the input pairs $\{C_{t_0}, C_{t_1}\}$, $\{C_{t_1}, C_{t_2}\}$, $\{C_{t_1}, C_{t_2}\}$, $\{C_{t_2}, C_{t_3}\}$, $\{C_{t_3}, C_{t_4}\}$, and the predictions are added to obtain the final network prediction $\hat{\mathrm{y}}$. Note, we train the model end-to-end based on the relative translation between the initial and the last volume and the model prediction $\hat{\mathrm{y}}$. We refer to this architecture as *Seq-TP-DN3D*.

Next, we implement the multi-path concept MP-CNN where multiple initial processing paths are used instead of two, see Section 4.2. Similar to TP-DN3D, the multi-path layers consist of three layers with shared weights, followed by our DenseNet module block with 3D convolutions after the fusion point. We refer to this architecture as *MP-DN3D*.

Fourth, we implement a DenseNet-based 4D CNN (*DN4D*) that jointly learns features from the temporal and spatial dimensions, see Section 4.1.4. The network's input is the entire 4D sequence T_{t_i}. This model consists of an initial convolutional part with three layers, followed by the DenseNet module block with 4D convolutions.

Fifth, we design *MP-DN4D*, an implementation of our MP-CNN4D architecture proposed in Section 4.1.4. At first, all volumes within the sequence are processed individually by 3D CNNs. Afterward, we reassemble the temporal dimension by concatenating the outputs into a temporal dimension. Then, we employ a our DenseNet

module block with 4D convolutions.

Furthermore, we consider several convolutional-recurrent models, as introduced in Section 4.3. We employ *cGRU-DN3D*, which is an implementation of the cGRU-CNN architecture. Here, the initial processing part is covered by a cGRU unit that aggregates the temporal sequence into a 3D spatial representation that is then processed by a DenseNet module with 3D convolutions, similar to the DenseNet module block of TP-DN3D. Furthermore, we employ *DN3D-GRU* and *DN3D-cGRU*, where the order of computations is flipped, as described in Section 4.3.

Training and Evaluation. We train our models to estimate the relative motion of an ROI using OCT volumes. Hence, we minimize the MSE loss function between the defined target motions y_{t_i} and our predicted motion vectors $\hat{\mathrm{y}}_{t_i}$.

$$J = \frac{1}{N_b} \sum_{j=1}^{N_b} \left\| \mathrm{y}_{t_i}^j - \hat{\mathrm{y}}_{t_i}^j \right\|^2 \tag{6.2}$$

Our goal is to estimate the relative motion between an initial volume C_{t_0} and a final volume C_{t_i}, corresponding to the target shift y_{t_i}. Given the nature of our acquisition setup, the intermediate shifts are also available. As these additional shifts represent additional motion information, we hypothesize that they could improve model training by enforcing more consistent estimates and thus regularize the problem.

We incorporate the additional motion information by forcing our models to also predict the relative shifts of previous volumes $C_{t_{i-1}}$ and $C_{t_{i-2}}$. Thus, we also consider the relative translations $\mathrm{y}_{t_{i-1}}$ and $\mathrm{y}_{t_{i-2}}$ and we extend the network output by also predicting $\hat{\mathrm{y}}_{t_{i-1}}$ and $\hat{\mathrm{y}}_{t_{i-2}}$. Note, the additional outputs $\hat{\mathrm{y}}_{t_{i-1}}$ and $\hat{\mathrm{y}}_{t_{i-2}}$ are only considered during training and are not required for application.

For optimization, we propose and evaluate the following loss function and introduce parameters $\lambda_{i-1}, \lambda_{i-2} \in [0, 1]$ for weighting of the additional temporal information, introduced as a regularization term.

$$J = \frac{1}{N_b} \sum_{j=1}^{N_b} \left\| \mathrm{y}_{t_i}^j - \hat{\mathrm{y}}_{t_i}^j \right\|^2 + \lambda_{i-1} \left\| \mathrm{y}_{t_{i-1}}^j - \hat{\mathrm{y}}_{t_{i-1}}^j \right\|^2 + \lambda_{i-2} \left\| \mathrm{y}_{t_{i-2}}^j - \hat{\mathrm{y}}_{t_{i-2}}^j \right\|^2 \tag{6.3}$$

The loss can be easily extended for the task of forecasting by additional loss functions terms for the prediction of $\mathrm{y}_{t_{i+1}}$ and $\mathrm{y}_{t_{i+2}}$. Each individual loss is calculated with Equation 6.2 and the total forecasting loss $\mathcal{J}^f$ is the sum of individual losses

$$J^f = \frac{1}{n_f} \sum_{i=1}^{n_f} \mathcal{L}_{t_i} \tag{6.4}$$

where $n_f = 3$ is the number of time steps to be forecasted.

We train all our models for $N_e = 150$ epochs, using Adam for optimization with a batch size of $N_b = 50$. To evaluate our models on previously unseen tissue regions, we randomly choose five independent ROIs for testing and validating each. For training, we use the remaining 30 ROIs.

We report MAE, the rMAE, and PCC for our experiments. The MAE is given in mm based on the calibration between galvo motor steps and image coordinates. We also

Tab. 6.20: Comparison of the different models for motion estimation. Our errors refer to the translation y between the template and the last volume of a motion sequence. The MAE is given in mm.

	MAE y_x	MAE y_y	MAE y_z	rMAE	PCC (%)
TP-DN3D	0.45 ± 0.52	0.42 ± 0.52	0.18 ± 0.15	0.34 ± 0.39	85.47
Seq-TP-DN3D	0.20 ± 0.21	0.15 ± 0.16	0.13 ± 0.12	0.16 ± 0.17	97.70
MP-DN3D	0.35 ± 0.45	0.18 ± 0.25	0.11 ± 0.09	0.21 ± 0.26	93.39
DN4D	0.22 ± 0.21	0.20 ± 0.24	0.13 ± 0.11	0.19 ± 0.19	96.86
MP-DN4D	$\mathbf{0.16 \pm 0.18}$	$\mathbf{0.13 \pm 0.15}$	$\mathbf{0.10 \pm 0.09}$	$\mathbf{0.13 \pm 0.14}$	**98.58**
cGRU-DN3D	0.21 ± 0.23	0.20 ± 0.25	0.14 ± 0.11	0.19 ± 0.20	96.52
DN3D-cGRU	0.19 ± 0.20	0.17 ± 0.20	0.12 ± 0.10	0.16 ± 0.15	97.76
DN3D-GRU	0.19 ± 0.19	0.16 ± 0.18	0.11 ± 0.11	0.15 ± 0.17	97.89

Tab. 6.21: Number of parameters and inference times for all models.

	Number of Parameters	Inf. Time
TP-DN3D	143 913	$3.74 \pm 0.52\,\mathrm{ms}$
Seq-TP-DN3D	143 913	$5.84 \pm 0.32\,\mathrm{ms}$
MP-DN3D	208 713	$5.23 \pm 0.27\,\mathrm{ms}$
DN4D	270 283	$9.78 \pm 0.74\,\mathrm{ms}$
MP-DN4D	258 323	$9.34 \pm 0.67\,\mathrm{ms}$
cGRU-DN3D	153 447	$15.84 \pm 0.52\,\mathrm{ms}$
DN3D-cGRU	396 223	$19.04 \pm 0.47\,\mathrm{ms}$
DN3D-GRU	238 183	$40.48 \pm 1.32\,\mathrm{ms}$

state the number of parameters and inference times for all models. For all experiments, we test our results for significant differences in the median of the absolute errors using Wilcoxon signed-rank test with a significance level of $\alpha = 0.05$.

Results

Results for the different CNN and convolutional-recurrent architectures are given in Table 6.20. Inference times and the number of trainable parameters are shown in Table 6.21. Overall, using a sequence of volumes improves performance significantly, and MP-DN4D performs best with a high PCC of 98.58%. Comparing MP-DN4D to TP-DN3D, the rMAE is reduced by a factor of approximately 2.6. Moreover, employing the two-path architecture on subsequent volumes and adding the estimations (Seq-TP-DN3D) performs significantly better than directly using the initial and the last volume (TP-DN3D) of a motion sequence.

Overall, MP-DN4D performs slightly better than the convolutional-recurrent models. DN3D-cGRU performs best among convolutional-recurrent models. Placing the recurrent processing unit at the end of the architecture tends to perform better.

Second, we show the MAE for different simulated velocities (motion magnitude) for the CNN-based models, see Figure 6.28. The error increases with increasing magnitude of the motion for all models. Comparing the different models shows that the error

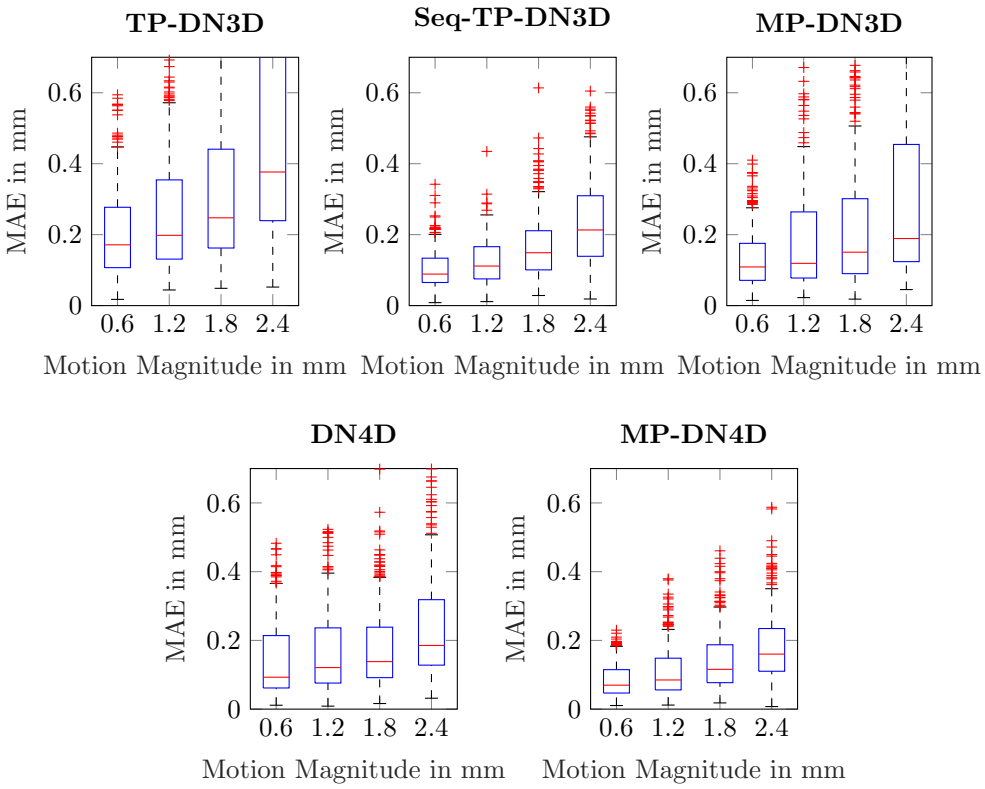


Fig. 6.28: MAE for increasing motion magnitudes with our CNN architectures. Results are shown for four motion groups, covering increasing magnitudes of motion.

increases only slightly for MP-DN4D, compared to the other models.

Third, Table 6.22 shows how rotations affect the performance of our best-performing model MP-DN4D during evaluation. First, we consider rotations as noise during motion and do not transform the target shifts. Second, we consider rotations as part of the motion and transform the target shifts accordingly. For small rotation angels $\alpha_{max} < 5°$, performance is hardly reduced. For larger rotation angels $\alpha_{max} > 5°$, lateral estimation performance is affected when rotations are considered as noise, while performance remains similar when rotations are considered as part of the motion.

Fourth, Table 6.23 demonstrates how motion distortions affect performance. We evaluate different magnitudes of motion distortions. The results show that performance is hardly reduced when only a few motion distortions are present ($p_{dist} < 10\%$). However, as we increase the amount of motion distortions, performance is notably affected, yet, performance is recovered when distortions are also considered during training.

Fifth, we address the temporal regularization strategy, see Table 6.24 for our best performing model MP-DN4D. We report performance metrics for various weighting factors λ_{i-1} and λ_{i-2}. Our results demonstrate that using the regularization strategy improves performance. Fine tuning the weights improves performance significantly with a high PCC of 99.06% for a weighting of $\lambda_{i-1} = 0.75$ and $\lambda_{i-2} = 0.75$.

Last, we address the problem of motion forecasting. The results are shown in Table 6.25. Again, models using full 4D sequences perform significantly better than the two-path models. Overall, MP-DN4D performs best. Predicting motion vectors further

Tab. 6.22: Evaluation of the performance for different rotation angels during motion. We evaluate the rotation as noise or as part of the motion, where the ground truth y is rotated accordingly. The rotation angle α_{max} refers to the relative rotation between the initial template volume and the last volume of a sequence. Results are shown for the architecture MP-DN4D. The errors refer to the translation y between the template and the last volume of a motion sequence and are given in mm.

	α_{max}	MAE y_x	MAE y_y	MAE y_z	rMAE	PCC (%)
Noise	2°	0.17 ± 0.18	0.13 ± 0.15	0.10 ± 0.09	0.13 ± 0.14	98.56
	5°	0.19 ± 0.18	0.15 ± 0.15	0.09 ± 0.09	0.14 ± 0.14	98.44
	10°	0.23 ± 0.19	0.16 ± 0.16	0.10 ± 0.09	0.17 ± 0.14	97.95
	20°	0.34 ± 0.25	0.23 ± 0.20	0.10 ± 0.09	0.22 ± 0.18	96.04
Motion	2°	0.16 ± 0.18	0.13 ± 0.15	0.09 ± 0.09	0.13 ± 0.14	98.60
	5°	0.16 ± 0.18	0.14 ± 0.15	0.10 ± 0.09	0.14 ± 0.14	98.55
	10°	0.17 ± 0.18	0.16 ± 0.15	0.10 ± 0.09	0.15 ± 0.14	98.35
	20°	0.19 ± 0.20	0.23 ± 0.19	0.10 ± 0.09	0.18 ± 0.16	97.48

in the future leads to larger errors.

Discussion

Motion estimation is a relevant problem for intraoperative OCT applications, for example, in the context of motion compensation [215] and surgical tool navigation [589]. While previous approaches for motion estimation relied on a template and moving images, we learn a motion vector from an entire sequence of OCT volumes. This leads to the challenging problem of 4D spatio-temporal deep learning.

Here, our proposed 4D spatio-temporal architectures, including 4D CNNs and recurrent-convolutional concepts, can be employed and compared. For a fair comparison, we also consider pairwise motion estimation along the sequence using TP-DN3D, aggregated to a final estimate. Our results in Table 6.20 show that the two-path method using only the start and the end volume perform worse than the other methods. This demonstrates that there is not enough information for motion estimation, or the motion is too large.

Using a full sequence of volumes with MP-DN3D performs significantly worse than the other deep learning approaches. This indicates that stacking multiple volumes in the models feature channel dimension is not optimal for temporal processing. This has also been observed for spatio-temporal problems in the natural image domain [491]. This is also supported by pairwise processing with Seq-TP-DN3D, which shows a significantly higher performance than the feature stacking approach and slightly higher performance than DN4D. Our proposed 4D architectures MP-DN4D outperforms all other approaches, including the previous deep learning concepts using two volumes [275] and pairwise processing.

Regarding convolutional-recurrent models, we find that performing temporal processing first with cGRU-DN3D does not perform as well as other methods. However, DN3D-cGRU and DN3D-GRU both perform very well. The key difference to other approaches that, for this application, temporal information is crucial and needs to be pre-

Tab. 6.23: Results for MP-DN4D when motion distortions are applied during evaluation; p_{dist} refers to the probability that a B-scan is shifted. We evaluate shifting the B-scans one (E-1) or two pixels (E-2) during evaluation. Also, we consider motion distortions of two pixels during training and evaluation (T/E-2). Our errors refer to the translation y between the template and the last volume of a motion sequence. The MAE is given in mm.

Type	p_{dist}	MAE y_x	MAE y_y	MAE y_z	rMAE	PCC (%)
E-1	50%	0.31 ± 0.33	0.29 ± 0.29	0.14 ± 0.11	0.25 ± 0.24	94.41
E-1	25%	0.20 ± 0.22	0.20 ± 0.20	0.11 ± 0.10	0.17 ± 0.17	97.37
E-1	10%	0.16 ± 0.18	0.16 ± 0.17	0.10 ± 0.09	0.14 ± 0.15	98.25
E-2	50%	0.33 ± 0.35	0.28 ± 0.28	0.14 ± 0.12	0.25 ± 0.24	94.27
E-2	25%	0.20 ± 0.21	0.20 ± 0.21	0.12 ± 0.10	0.17 ± 0.17	97.39
E-2	10%	0.17 ± 0.18	0.15 ± 0.16	0.10 ± 0.09	0.14 ± 0.14	98.27
T/E-2	50%	0.18 ± 0.21	0.15 ± 0.15	0.10 ± 0.08	0.14 ± 0.15	97.97

Tab. 6.24: Evaluation of the temporal loss regularization using different weighing factors λ_{i-1}, λ_{i-2}. Results are shown for the architecture MP-DN4D with respect to predicting the motion y between the template and the last volume of a sequence. Errors are given in mm.

λ_{i-1}	λ_{i-2}	MAE y_x	MAE y_x	MAE y_x	rMAE	PCC (%)
0	0	0.16 ± 0.18	0.13 ± 0.15	0.10 ± 0.09	0.13 ± 0.14	98.58
1	0	0.15 ± 0.22	0.12 ± 0.13	0.11 ± 0.10	0.13 ± 0.15	98.15
0.75	0	0.14 ± 0.13	0.11 ± 0.10	0.13 ± 0.10	0.14 ± 0.11	98.90
0.5	0	0.10 ± 0.09	0.14 ± 0.11	0.10 ± 0.08	0.12 ± 0.10	99.02
0.25	0	0.11 ± 0.11	0.14 ± 0.13	0.11 ± 0.09	0.12 ± 0.11	98.92
1	1	0.11 ± 0.10	0.19 ± 0.17	0.10 ± 0.09	0.14 ± 0.12	98.71
0.75	**0.75**	$\mathbf{0.09 \pm 0.09}$	$\mathbf{0.11 \pm 0.10}$	$\mathbf{0.10 \pm 0.08}$	$\mathbf{0.10 \pm 0.09}$	**99.06**
0.75	0.5	0.12 ± 0.10	0.10 ± 0.11	0.10 ± 0.08	0.11 ± 0.10	99.03

served throughout the model. In contrast, our other approaches of cGRU-CNN models largely rely on temporal data for more consistent estimates. Thus, the effectiveness of a convolutional-recurrent architecture is dependent on the application.

Next, we also consider the effect of different motor shift distances for our problem. Note, faster movements lead to a larger distance between subsequent volumes of a sequence and reduced overlap, making motion estimation harder as there are fewer features for finding correspondence. The results in Figure 6.28 show the performance for different distances between volumes. As expected, we observe a steady increase with larger distances for all models. For the approaches using just two volumes, the increase is substantial while it remains moderate for the 4D spatio-temporal models. Thus, 4D data is also beneficial for various magnitudes of motion to be estimated, and we demonstrate that the models effectively deal with different spatial distances between time steps.

Moreover, Table 6.22 shows how rotations affect performance for our best performing method when applied during evaluation. When rotations are considered as noise, only for

Tab. 6.25: Results for motion estimation (y_{t_i}) and forecasting ($\mathrm{y}_{t_{i+1}}$, $\mathrm{y}_{t_{i+2}}$). Errors are given in mm.

	MAE y_{t_i}	MAE $\mathrm{y}_{t_{i+1}}$	MAE $\mathrm{y}_{t_{i+2}}$	PCC (%)
TP-DN3D	0.40 ± 0.48	0.48 ± 0.61	0.57 ± 0.73	77.49
Seq-TP-DN3D	0.25 ± 0.29	0.31 ± 0.36	0.37 ± 0.44	92.27
MP-DN3D	0.19 ± 0.21	0.23 ± 0.26	0.28 ± 0.31	96.44
MP-DN4D	0.16 ± 0.18	0.19 ± 0.22	0.23 ± 0.27	97.41
cGRU-DN3D	0.20 ± 0.21	0.25 ± 0.26	0.31 ± 0.32	95.49
DN3D-cGRU	0.18 ± 0.20	0.24 ± 0.26	0.27 ± 0.28	96.98
DN3D-GRU	0.18 ± 0.22	0.23 ± 0.25	0.28 ± 0.30	96.80

large rotations $\alpha_{max} > 5°$ performance is notably reduced. However, when rotations are considered as part of the motion, performance remains similar even for larger rotations. As rotations were not present in the training data, the results indicate that our models are robust with respect to rotations.

Furthermore, we consider the problem of potential motion artifacts. The OCT device we employ is able to acquire an OCT volume in $1.2\,\mathrm{ms}$. According to Zawadzki et al., motion artifacts are not present for volume acquisition speeds below $100\,\mathrm{ms}$ [579]. However, to ensure that our methods are applicable to slower OCT devices as well, we consider the effect of fast and irregular motion that may cause image distortions. We consider motion distortions as lateral or axial shifts between B-scans of an OCT volume, similar to previous works [261, 559, 579]. The results in Table 6.23 demonstrate that motion distortions applied only during evaluation can affect performance. This highlights the importance of fast volumetric imaging when 4D data is used for motion estimation. However, when motion artifacts are also considered during training, performance can be recovered. These results indicate that using deep learning with 4D data is a viable approach, even if data is affected by fast and irregular motion distortions.

As temporal information appears to be beneficial at the model input, we also consider usage at the model output. Here, we introduce a regularization strategy that forces the model to learn consecutive motion steps. We also introduce weighting factors for fine-tuning of our approach. Our results in Table 6.24 demonstrate that the regularization method appears to be effective. While a weighting equal to one does not lead to immediate performance improvement, using a weighing of $\lambda_{i-1} = 0.75$, $\lambda_{i-2} = 0.75$ improves performance notably up to a PCC of 99.06 %. As a result, providing more information on the trajectory during training appears to be helpful for 4D motion estimation.

We also consider the task of motion forecasting, which is particularly relevant for applications such as motion compensation, where a mechanical system can lag behind. Our results in Table 6.25 show that forecasting is feasible, and errors only increase slightly for forecasting the next few time steps. This indicates that motion trajectory information can be effectively extracted from the 4D sequences and extrapolated for forecasting.

While our 4D deep learning methods significantly improve performance, their more costly 4D convolution operations also affect inference times, which is important for

application when real-time processing is required. Inference times in comparison to model size are shown in Table 6.21. While MP-DN4D significantly outperforms Seq-TP-DN3D in terms of motion estimation performance, Seq-TP-DN3D allows for faster predictions. Thus, there is a trade-off between performance and inference time for the different architectures. However, with an inference frequency of $107\,\mathrm{Hz}$, our 4D deep learning methods are already a viable approach for real-time motion estimation, which could be improved in the future by using more powerful hardware or additional low-level software optimization.

Summary

We address the problem of deep learning-based motion estimation and forecasting with 3.5D and 4D spatio-temporal OCT data. We address this challenge as a supervised learning problem where we obtain labeled data with a specialized data acquisition setup. Regarding our research question on data representations, we find that using 3.5D data with a template and a moving image volume performs worse than processing a full 4D sequence. While 3.5D data is competitive with a sequential processing approach, 4D data is significantly better for motion forecasting. Regarding our second research question on deep learning models, we find that separate spatial and temporal processing is advantageous over full 4D spatio-temporal processing with a plain 4D CNN. Both a multi-path CNN and a recurrent-convolutional model reach very high performance. The performance increase is traded off for longer processing times, however, real-time processing is feasible with our 4D deep learning models.

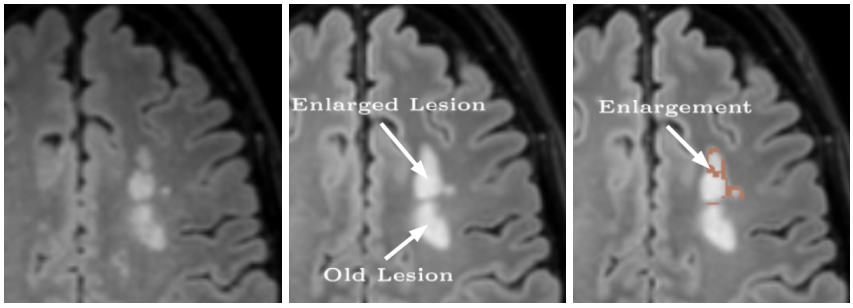
6.7 3.5D and 4D Data: MRI-Based Lesion Activity Segmentation

Another application scenario with 3.5D and 4D spatio-temporal data is the problem of lesion activity segmentation in MR images that we introduced in Section 5.5. In the context of MS, the most common image processing problem is the segmentation of lesions in individual MRI scans. However, for monitoring disease progression, lesion activity between two longitudinal MRI scans is the most important marker for inflammatory activity, and disease progression in MS [372]. The problem is particularly challenging as new lesions can be small, and changes are often subtle. So far, deep learning methods and most classical methods have only considered lesion segmentation for a single MRI volume. Thus, lesion activity is often derived from two independent segmentation maps, which is associated with high variability and inconsistencies [148]. Therefore, other approaches made use of information from the MRI volumes instead of lesion maps only, for example, using deformation fields or image differences. Overall, methods for detection of lesion activity have largely relied on classic image processing methods so far [86, 431].

Therefore, we study the segmentation of new and enlarged lesions that occurred between two time points using fully-convolutional CNNs. As a first step, we consider the 3.5D learning problem, where we aim to derive the lesion activity between two scans. A straight-forward deep learning approach for deriving lesion activity is to use a CNN for predicting individual lesion maps for each of the two time points and then taking the maps' difference. As previous work has demonstrated large inconsistencies for difference maps [148], combining volumes instead of final lesion maps might be beneficial. Approaches for volume combination include taking the volume difference, volume addition, or stacking volumes in the input channel dimension while using a standard single-path encoder-decoder model. However, this approach relies on high similarity between the scans and might suffer from inaccurate image registration or different acquisition parameters.

Therefore, initial independent processing might be advantageous as we have shown already in the context of IVOCT-based plaque classification, see Section 6.3. Thus, we consider two-path encoder-decoder 3D CNNs, a segmentation variant of our Siamese architecture we introduced in Section 4.2, where volumes are first processed independently by encoder paths. Then, the decoder jointly processes the combined feature maps from both volumes and predicts a segmentation of new and enlarged lesions. While initial independent processing might be beneficial, we hypothesize that some degree of information exchange in the encoder paths could improve performance. Therefore, we augment the encoders by the attention-guided interaction modules we proposed in Section 4.3.

As a second step, we extend the problem to full 4D deep learning by also considering scans taken before the first scan. Here, the idea is to provide an extended, potentially more robust baseline for a deep learning model. For example, the state of a lesion might be unclear in a single scan but becomes more pronounced when using an additional history that shows lesion development over time. For this deep learning problem, we employ our proposed cGRU-CNN-U architecture we described in Section 4.3 that



(a) Baseline Scan (b) Follow-up Scan (c) Labelled Scan

Fig. 6.29: Example of lesion activity in the case of an enlarged lesion. Old lesions and old lesion material belong to the background class for the task of lesion activity segmentation.

processes an arbitrary number of input images and produces a single spatial lesion map at the output.

Overall, regarding our first research question on data representations, we study the use of 3.5D and 4D MRI data for lesion activity segmentation. Notably, in this case, the temporal dimension needs to be understood as a longitudinal dimension in contrast to the previous short-term temporal dimension. With respect to our second research question on deep learning models, we study our Siamese architectures and our proposed attention-based augmentations for 3.5D data. For 4D data, we employ our proposed cGRU-CNN-U model and compare it to our multi-path Siamese approaches.

Methods and Datasets

Problem Definition. We consider the problem of finding a lesion activity map $\mathrm{y}_{t_i} \in \mathbb{R}^{n_h \times n_w \times n_d}$ using a longitudinal sequence of volumes $T_{t_i} = \{C_{t_0}, ..., C_{t_i}\}$ with $T_{t_i} \in \mathbb{R}^{n_t \times n_h \times n_w \times n_d}$. For the 3.5D case, we use $T_{t_i} = \{C_{t_{i-1}}, C_{t_i}\}$ and for the 4D case we employ $T_{t_i} = \{C_{t_{i-2}}, C_{t_{i-1}}, C_{t_i}\}$ where T_{t_i} is treated as a 4D spatio-temporal tensor. Thus, we try to find deep learning models $f_M : \mathbb{R}^{n_t \times n_h \times n_w \times n_d} \rightarrow \mathbb{R}^{n_h \times n_w \times n_d}$. Following conventions in the MS literature, we refer to the most recent scan C_{t_i} as the follow-up scan (C_{FU}), the previous scan $C_{t_{i-1}}$ as the baseline scan (C_{BL}), and scans before that as history scans (C_{HS}).

Dataset Properties and Labeling. The first dataset we use was obtained in a study that investigated MS heterogeneity at the University Hospital of Zurich, Switzerland. A 3.0T Philips Ingenia Scanner (Philips, Eindhoven, the Netherlands) was used for image acquisition using similar acquisition parameters for all scans. We use the FLAIR images as they are the recommended modality for MS lesion assessment [414].

For the 3.5D case, we consider a baseline scan C_{BL} and a follow-up scan C_{FU} for each patient. The mean slice thickness is $1.2\,\mathrm{mm}$ and the in-plane pixel spacing is $0.92\,\mathrm{mm} \times 0.92\,\mathrm{mm}$. Three independent experts provide a set of annotations each. As labeling is time-consuming, we reslice C_{FU} to $2\,\mathrm{mm}$ axial slice thickness. Then, we register C_{BL} to the resliced volume using a rigid registration. The raters view both C_{FU} and C_{BL} simultaneously while labeling the volumes slice by slice.

Raters mark new lesions that were not present in C_{BL} and also new lesion material

that appeared around lesions that were already present in C_{BL}. Thus, we consider the task of segmenting *lesion activity*, which is characterized by new lesions in C_{FU} and lesions that have grown between C_{BL} and C_{FU}. Lesions are treated as enlarged if there is an increase in size by at least $50\,\%$, following [347]. Old lesion regions are treated as background. Thus, the task is to learn *how much* lesions have changed over time. Example images with annotations for enlarged lesions are shown in Figure 6.29. We fuse the three raters' annotations by voxel-vise majority voting.

In total, the dataset contains data from 89 MS cases. For each case, the baseline scan and one follow-up scan are included. The mean time between BL and FU scans was 2.21 ± 1.09 years, and patients' mean age was 36.76 ± 8.67 years. The labeling process resulted in 43 out of 89 cases containing new or enlarged lesions. For cases with lesion activity, on average, 3.52 new and enlarged lesions were present per case. We refer to the dataset as the two time point dataset 1 ($\mathcal{V}_1$).

To ensure that our insights are also applicable to other datasets, we consider a second dataset for lesion activity segmentation. The dataset consists of 97 labeled pairs from 33 patients. All images were acquired at the Faculty of Medicine Carl Gustav Carus at Technische Universität Dresden using a Siemens MAGNETOM Verio 3.0T Scanner (Siemens, Germany). The slice thickness is $1.5\,\mathrm{mm}$ with an in-plane pixel spacing of $1.25\,\mathrm{mm} \times 1.25\,\mathrm{mm}$. The annotation strategy is the same strategy we used for dataset $\mathcal{V}_1$. Annotations are provided by one rater. The mean time between BL and FU scans was 1.00 ± 0.09 years, and patients' mean age was 39.64 ± 10.78 years. In total 41 out of 97 pairs contained new and enlarged lesions. On average, pairs with new and enlarged lesions contain 3.17 new and enlarged lesions. We refer to this dataset as the two time point dataset 2 ($\mathcal{V}_2$).

We compare our approach of explicit lesion activity segmentation to the use of difference maps, derived from full lesion segmentation maps of individual scans [148]. For this purpose, we also consider a dataset for per-scan lesion segmentation, similar to a majority of the publicly available MS datasets [74, 288]. The dataset contains 1500 FLAIR images, acquired during clinical routine. The images were anonymized and subsequently analyzed by jung diagnostics GmbH. Ground-truth lesion annotations are semi-automatically generated throughout the quality control process at jung diagnostics GmbH. Besides the different ground-truth annotation, the volumes are processed similar to the two time point dataset. We refer to this dataset as the single time point ($\mathcal{V}_3$) dataset.

For the extension to full the full 4D spatio-temporal learning problem, we consider a subset of $\mathcal{V}_1$ where one additional scan is available that was acquired before C_{BL}. We refer to this time point as C_{HS}. Thus, each example in the dataset is a spatio-temporal tensor $T_{t_i} \in \mathbb{R}^{n_t \times n_h \times n_w \times n_d}$ with $n_t = 3$. The learning object and ground-truth labels are the same as before. The dataset contains 44 cases. Here, we split off a validation set of 5 cases for hyperparameter tuning and a test set with 10 cases for evaluation. We refer to this dataset as the multi time point dataset ($\mathcal{V}_4$).

Preprocessing. We resample all volumes to $1\,\mathrm{mm} \times 1\,\mathrm{mm} \times 1\,\mathrm{mm}$ and rigidly register C_{BL} and C_{HS} to C_{FU}. We standardize each volume individually by subtracting the mean and dividing by the standard deviation. Then, we clip intensities at the 1^{st} and 99^{th} percentile. We resample the ground-truth volumes to $1\,\mathrm{mm} \times 1\,\mathrm{mm} \times 1\,\mathrm{mm}$ to match the input volume size. Following Egger et al. [118], we exclude very small lesions from our final ground-truth masks with less than $0.01\,\mathrm{ml}$ volume as they are not well defined

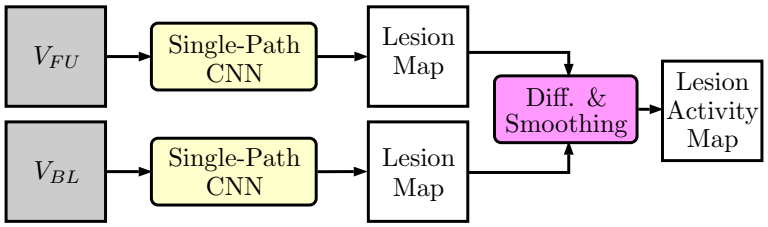


Fig. 6.30: Prediction process for our single-path, single time point ($\mathcal{V}_3$) CNN approach.

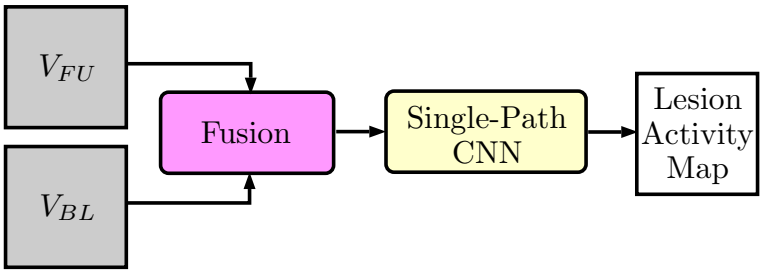


Fig. 6.31: Prediction process for our single-path CNNs with volume fusion. Fusion is performed by addition (RN3D-U Add), difference (RN3D-U Diff), or concatenation (RN3D-U Stack).

and likely false positives.

Deep Learning Models. As a first reference method, we apply the longitudinal pipeline of the LST toolbox, a popular non-deep learning (classic) method for longitudinal lesion change segmentation to our data [429]. The method first computes lesion probability maps for the individual time points using a logistic regression model. Then, a second algorithm compares the lesion probability maps and decides whether a change in lesion probability is significant or caused by FLAIR intensity variations. The method is available as a toolbox and does not require any hyperparameter tuning.

As a second reference, we consider a single-path (SP) fully-convolutional encoder-decoder (U-Net-like [409]) architecture (RN3D-U). The model is a ResNet-based segmentation architecture, as described in Section 3.3.6. After an initial convolutional layer with 32 feature maps, we use residual blocks [193] both in the encoder and decoder. In the encoder, spatial feature map sizes are reduced 3 times with convolutions having a stride of 2, which results in 4 spatial scales s_c^i inside the model and a maximum reduction of the spatial size by a factor of 8. We double the number of feature maps when the spatial size is halved. At scales s_c^1, s_c^2, s_c^3 and s_c^4 we employ 1, 2, 2 and 4 residual blocks, respectively. In the decoder, we use a single convolution followed by nearest-neighbor upsampling and a subsequent residual block at each scale. For the long-range connections between encoder and decoder, we follow VoxResNet [81] and use residual connections (summation) instead of feature concatenation. The model output is a dense segmentation of the same size as the input where each voxel contains the probability of having a positive label, which indicates lesion activity. Due to small batch size, we use instance normalization, a variant of group normalization [548], instead of batch normalization [214].

We employ this architecture for our second reference method with the $\mathcal{V}_3$ dataset.

Here, a single volume is fed into the CNN, and a segmentation of all lesions is predicted. This process is repeated independently for C_{BL} and C_{FU}. Afterward, we subtract the predicted lesion maps to obtain a map of new and enlarged lesions. Old lesions and lesions that shrunk in size are removed. We refer to this approach as RN3D-U STP. The process for deriving lesion activity is depicted in Figure 6.30.

Also, we use this architecture for volume fusion strategies at the input where we take the volume difference (RN3D-U Diff), volume addition (RN3D-U Add), or stack the two volumes in the channel dimension (RN3D-U Stack). The key difference to the RN3D-U STP method is that the volumes from both time points are processed jointly, and lesion activity is directly predicted by the models. The prediction pipeline is shown in Figure 6.31.

Next, we transform the single-path CNN into a two-path (TP) architecture such that the two volumes are processed in two phases. This is an implementation of Siamese architectures we proposed for segmentation, see Section 4.3. Here, the volumes are processed individually in the encoder path. Then, the volumes are processed jointly in the decoder. Before entering the decoder, we aggregate the feature maps from both paths. We consider fusion by subtraction (TP-RN3D-U Diff), addition (TP-RN3D-U Add), and by feature map concatenation (TP-RN3D-U Stack), which resembles the fusion techniques at the input for the SP CNNs. For the long-range residual connections between encoder and decoder, we concatenate the feature maps from the two encoder paths and then add the result to the decoder feature map.

This strategy allows for individual feature learning for each time point before joint learning. To draw a connection to our single-path approaches, where the volumes are processed jointly from the beginning, we also incorporate targeted information exchange between the encoder paths. For this purpose, we employ our attention-guided interaction modules that we proposed in Section 4.2. We consider the attention variations described in Section 4.2. Attention method A generates two attention maps where the map for the baseline path is generated from the follow-up path and vice-versa. Attention method B generates two attention maps where feature tensors from both the baseline and the follow-up paths are used. Attention method C uses the same input but generates a joint attention map for both paths.

For the 4D problem, we first consider a straightforward extension of the Siamese concept to 4D. Here, we employ n_t paths instead of two paths for processing all n_t volumes in the encoder (MP-RN3D-U Stack). By sharing parameters for the encoder part, there is no increase in terms of model parameters. Furthermore, we employ our cGRU-CNN-U model with a ResNet backbone (cGRU-RN3D-U), as introduced in Section 4.3. Structurally, it is similar to the multi-path Siamese CNN extension, except for the temporal aggregation before the decoder. Here, we employ a convolutional GRU layer with 256 feature maps that processes the temporal sequence and outputs a single spatial representation. The long-range connections are also augmented by convolutional GRU units with the number of feature maps that match the respective resolution level.

Training and Evaluation. During training, we randomly crop subvolumes of size $128 \times 128 \times 128$ from the volumes. We randomly flip the volumes along the x, y, and z direction with a probability of $p = 0.5$. We train all models using Adam with a dice loss function and a batch size of $N_b = 1$ for $N_e = 300$ epochs. We use an initial learning rate of $\alpha_{lr} = 10^{-4}$ with exponential decay. For evaluation, we obtain predictions for entire

volumes by taking 27 evenly spaced, overlapping crops from each volume, and obtain their prediction. We merge all predicted crops to one prediction volume by calculating the mean probability in overlapping regions. Then, we calculate metrics using the ground-truth label volumes from each rater. The final values are averaged across all raters.

Evaluation, Metrics, and Experiments. We use a three-fold cross-validation strategy for validation and testing. We define three mutually exclusive folds with 26 cases each. Each fold is divided into a validation and a testing split. For hyperparameter tuning, we train on the remaining cases for each fold and chose hyperparameters based on validation performance, averaged across the three validation splits. For testing, we train on all cases except for the test splits for each fold. Performance metrics are reported for the test splits. Metrics are calculated for all cases and aggregated across all test splits. The same strategy is used for datasets $\mathcal{V}_1$ and $\mathcal{V}_2$. For dataset $\mathcal{V}_4$, we use a similar strategy with fewer cases per fold. For experiments with dataset $\mathcal{V}_3$, training for single time point segmentation is performed on all cases in $\mathcal{V}_3$. Testing is performed on the test splits of dataset $\mathcal{V}_1$. Experiments with two-path models are performed on $\mathcal{V}_1$ unless indicated otherwise.

In terms of metrics, we consider lesion-wise metrics and the dice coefficient. The number of new and enlarging lesions are the most relevant indicators of disease progression. Therefore, our primary metrics are the lesion-wise true positive rate (LTPR) and lesion-wise false positive rate (LFPR). We define lesions as groups of 26-connected voxels. LTPR is the number of lesions that overlap in a prediction and ground-truth map divided by the total number of lesions in the ground-truth map. LFPR is the number of lesions that do not overlap in a prediction and ground-truth map divided by the total number of predicted lesions. While LFPR and LTPR are indicators for lesion presence, we use the dice score to quantify lesion overlap. Thus, in case a predicted and a ground-truth lesion overlap, we calculate the dice score for that lesion. Note that calculating a dice score for non-overlapping lesions is not meaningful as the score is always 0. All three metrics depend on a decision threshold (typically 0.5) for a model's predicted probabilities. For each model, we chose the optimal threshold based on ROC analysis, where the sum of LTPR and $1-$LFPR is maximized. For each metric, we provide the mean value and standard deviation. We test for a significant difference in the median of the different metrics using Wilcoxon's signed-rank test with a confidence level of $\alpha = 0.05$. Thus, we claim statistical significance if the test yields $p < 0.05$. For dataset $\mathcal{V}_1$, we consider the mean interrater performance, calculated over all pairwise comparisons between the three raters for comparison to model results.

We first consider the two reference scenarios with RN3D-U STP and the approach using the LST toolbox. We compare this approach to several single-path, fusion-based deep learning methods, including volume subtraction and addition, channel stacking, and basic two-path architectures. Second, we provide results for two-path architectures, enhanced by our novel attention-guided interaction modules. We consider different attention blocks, different attention locations, and an additional evaluation with dataset $\mathcal{V}_2$. Then, we present qualitative results for our attention maps. Last, we present results for the full 4D problem.

Tab. 6.26: Mean value and standard deviation of dice, LTPR, and LFPR in percent for the reference methods LST and RN3D-U STP in comparison to single- and two-path deep learning methods with different fusion techniques (difference, addition, channel stacking). The best performing method for each metric is marked bold.

	Dice	LTPR	LFPR
Interrater	67.2 ± 23.9	72.6 ± 32.1	22.9 ± 28.3
LST	50.0 ± 22.1	65.8 ± 33.7	65.6 ± 28.8
RN3D-U STP	52.2 ± 26.2	66.9 ± 35.9	55.1 ± 35.4
RN3D-U Diff	48.4 ± 29.5	52.9 ± 38.0	38.5 ± 35.7
RN3D-U Add	46.3 ± 28.2	54.0 ± 36.1	43.6 ± 30.7
RN3D-U Stack	56.2 ± 26.9	59.7 ± 36.6	33.6 ± 33.5
TP-RN3D-U Diff	58.1 ± 30.9	61.5 ± 35.7	$\mathbf{28.3 \pm 29.6}$
TP-RN3D-U Add	59.2 ± 25.3	$\mathbf{63.7 \pm 35.8}$	30.4 ± 32.5
TP-RN3D-U Stack	$\mathbf{58.3 \pm 29.4}$	60.6 ± 36.0	31.7 ± 33.9

Results

First, we consider results for the reference approaches compared to several single-path, fusion-based deep learning methods, see Table 6.26. The LST method's LFPR is very high. Note that the LST result is not directly comparable to the other result, as the predictions cannot be evaluated at the respective interrater LFPR. Comparing the RN3D-U STP to the RN3D-U model, metrics are similar for the addition and subtraction of baseline and follow-up scan. However, the performance difference is statistically significant between RN3D-U Stack and RN3D-U STP both for the LTPR and LFPR. For the TP-RN3D-U models, it is notable that all variants perform better than the RN3D-U approaches. In particular, TP-RN3D-U Diff and TP-RN3D-U Add significantly outperform all other RN3D-U variants in terms of the LTPR and LFPR. There is no significant difference in performance between the three variants. Notably, the performance difference between all models and the interrater performance is statistically significant for the LTPR and LFPR.

Second, we present results for our different attention mechanisms at location 16^3 for the three two-path models TP-RN3D-U Diff, TP-RN3D-U Add, and TP-RN3D-U Stack, see Table 6.27. For all three fusion methods, the attention blocks improve performance. Comparing the attention methods to their baseline without attention, the median LTPR and dice coefficients are significantly different across all fusion and attention methods. For the LFPR, the attention methods also improve performance, but the difference in the median is not significant. Comparing attention method C to the interrater performance, there is no significant difference in the median of all metrics for all fusion techniques.

Third, we demonstrate how the location of the attention block affects performance for the two-path models, see Table 6.28. Attention blocks at location 16^3 tend to perform best. When attention blocks are placed further towards the model input, performance tends to go down. When placing attention blocks at all three locations at the same time,

Tab. 6.27: Mean value and standard deviation of dice, LTPR, and LFPR in percent for our different attention methods A, B, and C at location 16^3. The best performing attention method for each two-path fusion type (difference, addition, or channel stacking) is marked bold.

	Dice	LTPR	LFPR
Interrater	67.2 ± 23.9	72.6 ± 32.1	22.9 ± 28.3
TP-RN3D-U Diff	58.1 ± 30.9	61.5 ± 35.7	28.3 ± 29.6
TP-RN3D-U Diff A	62.4 ± 23.6	72.0 ± 33.2	27.6 ± 29.2
TP-RN3D-U Diff B	61.9 ± 24.2	72.4 ± 32.4	28.0 ± 32.5
TP-RN3D-U Diff C	$\mathbf{63.2 \pm 22.9}$	$\mathbf{73.2 \pm 32.0}$	$\mathbf{26.2 \pm 29.1}$
TP-RN3D-U Add	59.2 ± 25.3	63.7 ± 35.8	30.4 ± 32.5
TP-RN3D-U Add A	60.9 ± 24.4	69.9 ± 33.4	31.4 ± 34.4
TP-RN3D-U Add B	61.0 ± 25.9	69.4 ± 35.0	28.9 ± 32.8
TP-RN3D-U Add C	$\mathbf{62.2 \pm 22.2}$	$\mathbf{74.2 \pm 31.5}$	$\mathbf{26.4 \pm 30.2}$
TP-RN3D-U Stack	58.3 ± 29.4	60.6 ± 36.0	31.7 ± 33.9
TP-RN3D-U Stack A	64.7 ± 24.7	71.2 ± 33.2	28.5 ± 30.1
TP-RN3D-U Stack B	63.2 ± 26.0	70.9 ± 34.4	27.5 ± 30.5
TP-RN3D-U Stack C	$\mathbf{65.6 \pm 24.8}$	$\mathbf{73.1 \pm 32.0}$	$\mathbf{26.9 \pm 30.6}$

LTPR and LFPR do not improve further. The dice score, however, improves for some scenarios.

Fourth, we show boxplots of the three TP architectures with attention C at location 16^3, see Figure 6.32. In terms of the LFPR, the distribution of each model is similar to the interrater distribution. For the LTPR, the same observation can be made. The distribution of the interrater dice scores is slightly higher than the dice score of TP-RN3D-U Add C. For TP-RN3D-U Diff C and TP-RN3D-U Stack C, the distribution has a similar median, and there is no significant difference in the median, compared to the interrater performance.

Fifth, we investigate whether our attention method is also advantageous on another dataset, using dataset $\mathcal{T}_2$. The results for attention location 16^3 are shown in Table 6.29. In general, the attention methods improve performance once again, across multiple fusion strategies. The difference is largest for the LTPR, where all attention methods across all fusion strategies significantly outperform the baseline. For the other metrics, there is also consistent improvement.

Sixth, we show qualitative results for our attention mechanism, see Figure 6.33. The Figure shows two examples for the effect of our attention mechanism for the model TP-RN3D-U Stack C 64^3. Feature maps from the baseline and follow-up scan are used to compute an attention map. This map is then multiplied with each feature map. The attention maps show very low-intensity regions, where old lesions are present in the baseline scan. In the follow-up scan's feature map, regions corresponding to lesions show a high intensity. After applying the attention maps, we can observe a masking

Tab. 6.28: Mean value and percentile ranges of dice, LTPR, and LFPR in percent for different attention locations with our three two-path models with different fusion techniques (difference, addition, and channel stacking) and attention C. The location is indicated by the size of the feature maps at that level. *All* refers to attention modules at all three locations.

	Dice	LTPR	LFPR
Interrater	67.2 ± 23.9	72.6 ± 32.1	22.9 ± 28.3
TP-RN3D-U Diff 16^3	63.2 ± 22.9	$\mathbf{73.2 \pm 32.0}$	$\mathbf{26.2 \pm 29.1}$
TP-RN3D-U Diff 32^3	60.1 ± 23.5	72.2 ± 32.9	28.5 ± 32.9
TP-RN3D-U Diff 64^3	62.0 ± 25.6	70.0 ± 34.9	29.1 ± 33.9
TP-RN3D-U Diff all	$\mathbf{64.2 \pm 21.0}$	73.0 ± 30.3	27.8 ± 29.9
TP-RN3D-U Add 16^3	62.2 ± 22.2	$\mathbf{74.2 \pm 31.5}$	$\mathbf{26.4 \pm 30.2}$
TP-RN3D-U Add 32^3	61.0 ± 25.9	70.2 ± 35.0	29.8 ± 31.5
TP-RN3D-U Add 64^3	62.8 ± 24.3	71.4 ± 32.5	30.0 ± 31.1
TP-RN3D-U Add all	$\mathbf{64.2 \pm 23.9}$	71.9 ± 32.3	27.8 ± 32.6
TP-RN3D-U Stack 16^3	$\mathbf{65.6 \pm 24.8}$	$\mathbf{73.1 \pm 32.0}$	$\mathbf{26.9 \pm 30.6}$
TP-RN3D-U Stack 32^3	62.4 ± 27.2	72.2 ± 33.6	28.4 ± 32.5
TP-RN3D-U Stack 64^3	62.8 ± 26.2	72.0 ± 34.8	29.0 ± 32.9
TP-RN3D-U Stack all	62.5 ± 26.8	72.5 ± 34.1	29.3 ± 31.0

Tab. 6.29: Mean value and standard deviation of dice, LTPR, and LFPR in percent for our different attention methods A, B, and C at location 16^3 for the second dataset $\mathcal{V}_2$. The best performing attention method for each two-path fusion type (difference, addition, or channel stacking) is marked bold.

	Dice	LTPR	LFPR
TP-RN3D-U Diff	60.6 ± 29.1	67.5 ± 48.8	27.0 ± 33.9
TP-RN3D-U Diff A	61.9 ± 27.0	75.1 ± 33.8	$\mathbf{24.9 \pm 30.9}$
TP-RN3D-U Diff B	63.1 ± 27.4	75.3 ± 33.5	28.2 ± 32.2
TP-RN3D-U Diff C	$\mathbf{64.1 \pm 26.6}$	$\mathbf{76.5 \pm 33.2}$	25.0 ± 27.7
TP-RN3D-U Add	60.9 ± 25.4	68.6 ± 47.8	34.2 ± 35.5
TP-RN3D-U Add A	$\mathbf{63.3 \pm 28.3}$	71.6 ± 35.2	28.6 ± 29.7
TP-RN3D-U Add B	60.9 ± 26.1	$\mathbf{75.3 \pm 33.7}$	$\mathbf{22.7 \pm 32.3}$
TP-RN3D-U Add C	61.2 ± 27.6	73.7 ± 35.2	26.4 ± 30.2
TP-RN3D-U Stack	62.7 ± 26.2	71.8 ± 33.5	28.1 ± 30.9
TP-RN3D-U Stack A	$\mathbf{65.2 \pm 23.4}$	$\mathbf{79.4 \pm 29.7}$	22.8 ± 26.9
TP-RN3D-U Stack B	64.6 ± 24.2	76.0 ± 32.1	26.5 ± 30.6
TP-RN3D-U Stack C	64.3 ± 25.5	77.2 ± 33.0	$\mathbf{21.8 \pm 27.8}$

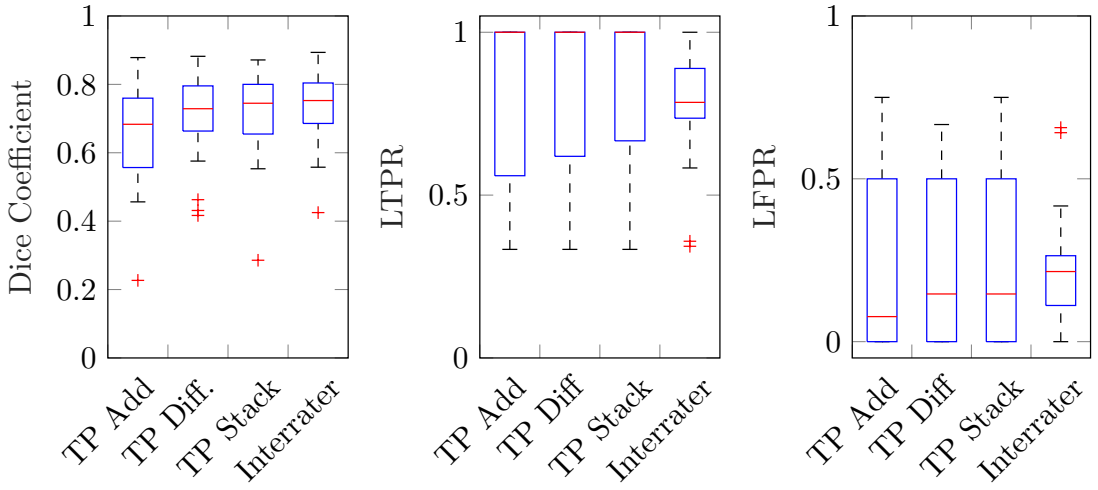


Fig. 6.32: Boxplots for the three two-path models with attention (TP-RN3D-U C) at location 16^3 compared to the interrater performance.

Tab. 6.30: Mean value and standard deviation of dice, LTPR, and LFPR in percent for the 4D deep learning problem with dataset $\mathcal{V}_4$. We consider 3.5D and 4D data and different models for 4D data.

	Dice	LTPR	LFPR
TP-RN3D-U Stack $n_t = 2$	62.1 ± 17.3	81.2 ± 28.3	30.1 ± 29.5
MP-RN3D-U Stack $n_t = 3$	59.8 ± 16.1	83.2 ± 28.0	35.4 ± 33.2
TP-RN3D-U Add $n_t = 2$	61.0 ± 16.2	80.8 ± 29.1	31.4 ± 32.6
MP-RN3D-U Add $n_t = 3$	61.5 ± 14.8	81.4 ± 28.8	33.8 ± 34.9
TP-RN3D-U Diff $n_t = 2$	61.7 ± 15.0	81.7 ± 27.9	32.0 ± 32.1
MP-RN3D-U Diff $n_t = 3$	60.9 ± 15.7	79.9 ± 29.4	28.1 ± 29.2
cGRU-RN3D-U $n_t = 2$	63.3 ± 13.6	81.0 ± 27.8	21.8 ± 23.8
cGRU-RN3D-U $n_t = 3$	$\mathbf{64.5 \pm 21.6}$	$\mathbf{84.3 \pm 27.3}$	$\mathbf{19.6 \pm 20.5}$

effect. Attention maps mask our lesions that were already present in the baseline scan. Thus, in the follow-up scan's feature map, only small high-intensity regions remain. Comparing these regions to the final predictions and ground-truth maps shows that the remaining high-intensity regions correspond to a new lesion. Therefore, the attention maps masked out old lesions while preserving the new lesions.

Last, we show results for the 4D problem, see Table 6.30. For the Siamese two path models, there is no notable increase in performance when using the multi-path extension with $n_t = 3$. However, for cGRU-RN3D-U, performance improves notably. In particular, the difference in the median is significant for the LFPR. For the other metrics, there is no significant difference but also a notable increase.

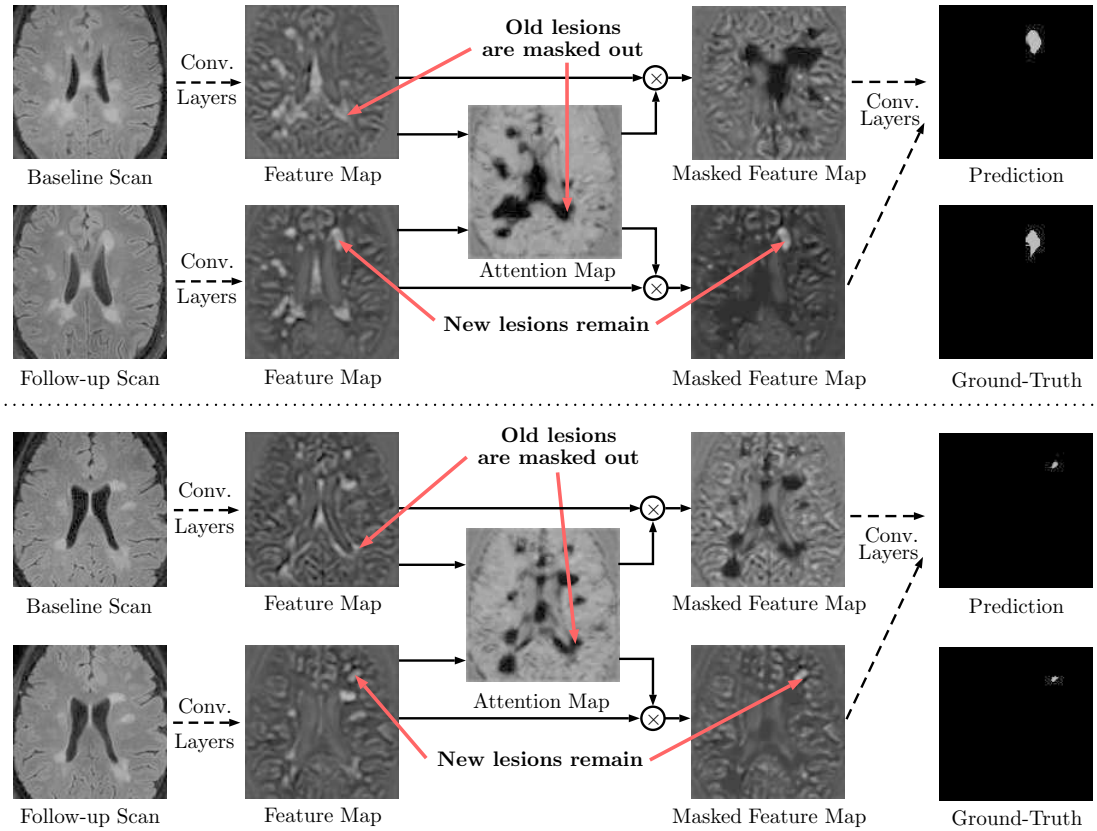


Fig. 6.33: Visualization of the effect of our attention method for the model TP-RN3D-U Stack C 64^3. Two example cases are shown (top and bottom). Left, an axial slice of the baseline and follow-up scan are shown. In the center, example features maps, the attention maps, and the masked feature maps are shown. Right, the final model prediction and ground-truth are shown.

Discussion

We address the problem of multiple sclerosis lesion activity segmentation from two time points using deep learning models. Compared to lesion segmentation in a single scan, the task of lesion activity segmentation is much more challenging as lesion size can be small and it is difficult to distinguish lesion activity from slight registration errors and intensity variations between two scans.

As a reference for lesion activity segmentation from two MRI scans, we consider two methods. The first method is the LST toolbox, an established classic, non-deep learning method [431] that uses FLAIR images for lesion prediction. The second method uses the typical approach of individual lesion map differences [148]. We compare these references to deep learning-based, single-path fusion techniques for baseline and follow-up scan aggregation by addition and difference. This follows the rationale of previous methods that relied on difference images [146]. The two reference methods perform similar to these approaches, which indicates that more advanced architectures are required to improve lesion activity segmentation. When stacking baseline and follow-up scan into the input channel dimension, which is typically used for color channels in the natural image domain, performance increases significantly. This indicates that individual volume processing is advantageous for this task. Therefore, we design a two-path architecture where volumes are processed individually in the encoder path. The two paths are combined with three different fusion methods, which all perform better than the single-path models with volume fusion and the two reference methods. In particular, using addition or difference for fusion leads to statistically significant differences in terms of our primary metrics, the LTPR and LFPR. However, all these methods show a performance that is still significantly different from the interrater performance.

To improve performance further, we, therefore, employ our proposed attention-guided interaction modules. These modules allow for information exchange between the two encoder paths before feature fusion at the model's decoder. Overall, we can observe a performance improvement across all fusion techniques and attention modules for the LFPR and the LTPR. In particular, attention methods C stands out as it performs best, and the performance improvement over the normal two-path models is statistically significant for the LTPR across all fusion techniques while maintaining a low LFPR. For all metrics, there is no significant difference between the interrater performance and our two-path model with attention method C. This suggests the usability of our proposed method. This insight is confirmed on a second, independent dataset that was acquired at a different location with a different device. Here, it is notable that the other attention methods A and B also lead to high performance. Furthermore, we study how the location of our attention blocks inside the network affects performance, see Table 6.28. Here, we can observe a decrease in performance if the modules are moved further away from the fusion point towards the model input. This could indicate that our attention blocks are most effective for guiding feature fusion. If the information exchange happens too early inside the network, the effect potentially decreases until the actual feature fusion occurs. In addition, we also consider a scenario where we place attention blocks at all three locations. However, we do not observe a synergy effect as the LTPR and LFPR do not improve further. This indicates that additional flow of information between path does not provide an additional benefit and that a single attention module is likely sufficient to

guide information exchange.

Next, we provide a qualitative evaluation of our attention method. While quantitative results already imply the effectiveness of the proposed attention modules, we also provide a visualization of the computed attention maps, see Figure 6.33. The attention maps appear to provide an effect that is similar to masking. This is particularly visible for the feature maps derived from the follow-up scan. After the masking process, the old lesions are attenuated, and only a new lesion remains with high intensity. This implies that our module design for information exchange between the two paths is very effective as key information is successfully transferred.

Last, we also consider the full 4D problem with an additional history. While the straightforward multi-path extension of Siamese CNNs does not improve performance, our proposed cGRU-RN3D-U model significantly improves performance in terms of the LFPR. This indicates that using 4D data instead of 3.5D data can be advantageous. However, it depends on the deep learning model that is employed. Also, a total temporal dimension size of $n_t = 3$ is relatively small. Datasets with a longer history might show additional benefits of 4D data. Furthermore, our attention-guided interaction methods could be extended to the 4D domain in future work.

Summary

We consider the problem of deep learning-based lesion activity segmentation with 3.5D and 4D spatio-temporal MRI data. For 3.5D data with a baseline scan and a follow-up scan, we find that deep learning approaches are more effective than classical methods. Also, using a Siamese structure with two encoders is preferable over combing the MRI scans before processing by the model. Performance is improved further when employing our attention-guided interaction models. This can be observed across two independent datasets, and plausible attention maps demonstrate a masking effect that focuses the network on new lesions. For the 4D case, we observe improved performance over 3.5D data if we employ our proposed convolution-recurrent model. Thus, in terms of our research question on data representations, we find that 4D could be advantageous over 3.5D, given the right deep learning model. In terms of our second research question on deep learning models, we find that our proposed architectures with attention mechanisms and recurrent temporal aggregation are effective for the problem of lesion activity segmentation.

6.8 2D, 3D, and 4D Data: Volume-Based Force Estimation

Another problem that comes with different data representations is vision- and volume-based force estimation, which we introduced in Section 5.1. The problem needs to be differentiated from needle-based force estimation where scalar signals, 1D spatial or 2D spatio-temporal images are used. Most vision-based force estimation approaches rely on 2D color or depth images, which leads to 3D spatio-temporal data in case temporal information is taken into account.

Here, we explore vision-based force estimation with the imaging modality OCT and deep learning methods. The use of OCT opens up several interesting aspects in terms of data representations and deep learning methodology that need to be studied. For the problem of vision-based force estimation, we address three significant challenges.

The first challenge we address is the high-dimensional nature of OCT data. While 4D spatio-temporal data should be rich in information, processing it is computationally expensive. One approach for reducing computational effort is to encode the spatial 3D volume information in a lower-dimensional space. For vision-based force estimation, it is reasonable to assume that the tissue *surface* being deformed by a force carries information that can be encoded in a space that is smaller than full 3D volumes. For example, previous vision-based force estimation methods have captured deformation with a feature vector or 2D depth maps [28]. This raises the question of whether full 3D volumes are advantageous, or whether lower-dimensional surface encodings are sufficient. Volumes could potentially capture sub-surface tissue compression and thus provide a richer feature space, as we demonstrated for OCT-based pose estimation. Therefore, we study different representations of the spatial image information by considering full 3D volumes as well as 2D and 3D image data representations of the deformed surface.

Another way of controlling deep learning-based 4D spatio-temporal data processing effort is to reduce or extend the temporal dimension. Thus, the second challenge we address is the effective use of the temporal data dimension. We investigate and quantify the effect and benefit of different temporal processing techniques as well as a longer or shorter temporal history. In this context, we also consider the problem of short-term force forecasting. Assuming some non-random deformation and force patterns over time, forces should be predictable, given a history of data. Forecasting forces could enable safety mechanisms for robot-assisted interventions [184, 188], as large force value increases could be detected earlier. Although spatio-temporal data has been used for force estimation, forecasting has not been studied.

The third challenge we address is 4D spatio-temporal deep learning. Similar to our other 4D deep learning applications, this problem is complicated due to the substantial increase in computational and memory requirements compared to architectures for lower-dimensional data. Also, it is unclear what type of spatial and temporal processing mechanisms should be employed for OCT volume-based force estimation. Therefore, we employ our different mechanisms for 4D spatio-temporal deep learning that we introduced in Section 4. We compare the different processing techniques on 4D data and also consider their 3D counterparts that process 3D spatio-temporal data consisting of

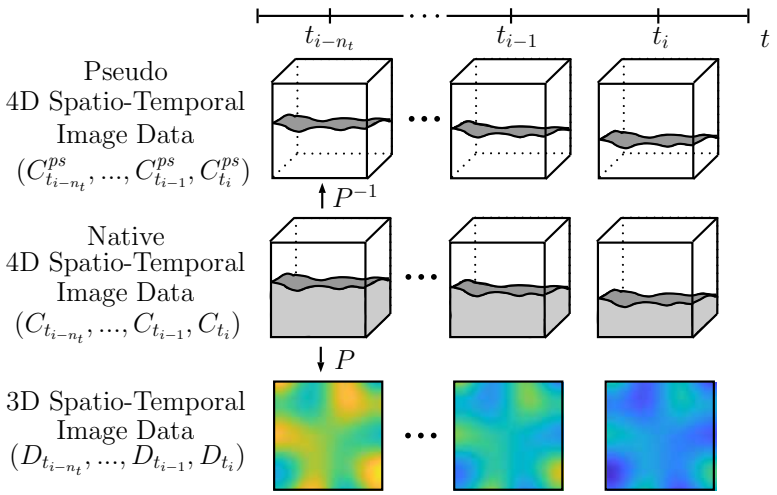


Fig. 6.34: The image data representations that we study. In the middle row, a sequence of full volumes is shown. Bottom, a sequence of the extracted surface, encoded as a depth image, is shown. Depth is represented by color value. Top, the extracted surfaces are represented by points in a volume.

streams of depth maps.

Summarized, in terms of our research question on data representations, we study the use of spatial 2D and 3D as well as spatio-temporal 3D and 4D data for vision-based force estimation. We explore different 3D spatial data representations and the effect of a temporal history. Furthermore, we study the prediction of future force values and the relation of prediction performance to the temporal history. Regarding our second research question of deep learning methods, we study CNN-based models and recurrent-convolution models we proposed for both 3D and 4D spatio-temporal data processing.

Methods and Datasets

Problem Definition. Formally, we consider $T_{t_i} = \{C_{t_{i-n_t+1}}, ..., C_{t_{i-1}}, C_{t_i}\}$, a sequence of volumes, which is used to estimate the target force $\mathrm{y}_{t_{i+n_f}}$. The volumes $C_{t_i} \in \mathbb{R}^{n_h \times n_w \times n_d}$ capture the deformation of tissue to which target forces $\mathrm{y}_{t_{i+n_f}} \in \mathbb{R}$ are applied by an instrument. The 4D spatio-temporal tensor is of size $T_{t_i} \in \mathbb{R}^{n_t \times n_h \times n_w \times n_d}$. Along the temporal dimension, n_t denotes the number of volumes being used for prediction, and n_f denotes the prediction horizion in terms of the number of time steps that we predict into the future. For $n_f = 0$ we refer to the problem as force *estimation*, for $n_f > 0$ we refer to the problem as force *forecasting*.

We also study two transformations of spatial data. First, we consider a projection $P : \mathbb{R}^{n_h \times n_w \times n_d} \rightarrow \mathbb{R}^{n_h \times n_w}$. The resulting depth images $D_{t_i} \in \mathbb{R}^{n_h \times n_w}$ represent a surface within the 3D volumes. Second, we represent the surface by points in the actual 3D volume by employing $P^{-1} : \mathbb{R}^{n_h \times n_w} \rightarrow \mathbb{R}^{n_h \times n_w \times n_d}$, where depth images are encoded as *pseudo volumes* $C^{ps}_{t_i} \in \mathbb{R}^{n_h \times n_w \times n_d}$ in 3D space. All data representations are shown in Figure 6.34.

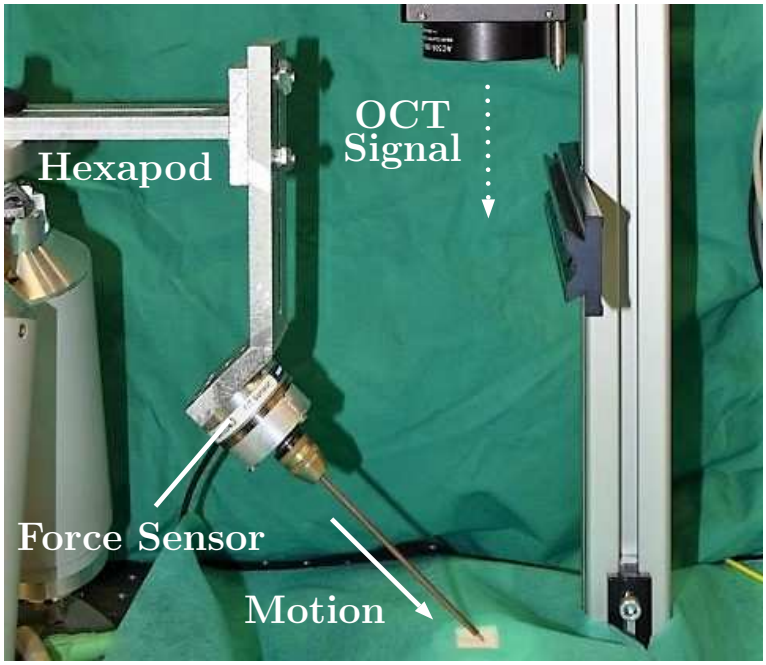


Fig. 6.35: The experimental setup we use for data acquisition.

We design and evaluate models with spatio-temporal (ST) and spatial (S) data for learning $f_M : \mathbb{R}^{n_t \times n_h \times n_w \times n_d} \to \mathbb{R}$ (4D-ST), and compare to models $f_M : \mathbb{R}^{n_t \times n_h \times n_w \times n_d} \to \mathbb{R}$ (ps-4D-ST), $f_M : \mathbb{R}^{n_t \times n_h \times n_w} \to \mathbb{R}$ (3D-ST), $f_M : \mathbb{R}^{n_h \times n_w \times n_d} \to \mathbb{R}$ (3D-S), and $f_M : \mathbb{R}^{n_h \times n_w} \to \mathbb{R}$ (2D-S).

Experimental Setup and Datasets. The experimental setup is shown in Figure 6.35. A hexapod robot (H-820.D1, Physik Instrumente) is equipped with a force sensor (Nano 43, ATI) for ground-truth annotation and an instrument. We use a needle tool for our experiments. The hexapod performs movement along the needle axis, which deforms a silicon phantom, representing a typical surgical pushing task [321]. The phantom is imaged by an OCT scan head, which continuously acquires volumes. Note that the force sensor is only required for training data acquisition and not for the actual application. To ensure generalization to real-world tissue applications, we also consider a dataset where animal heart muscle tissue is used instead of a phantom.

The OCT device is a swept-source OCT (OMES, OptoRes, Germany). Based on interferometry and infrared light with a central wavelength of $1315\,\mathrm{nm}$, the inner structure of scattering materials can be imaged up to $1\,\mathrm{mm}$ in depth. A volume can be acquired by repeatedly scanning at neighboring lateral positions. We scan at 32×32 lateral positions. With an A-Scan resolution of 430 pixels, the raw volumes thus have a size of $32 \times 32 \times 430$. We downsample the raw OCT volumes to a reduced size of $32 \times 32 \times 32$ using cubic interpolation due to the high computational and memory requirements of 4D architectures. This leads to similar spatial resolutions in all directions, and thus simplified architecture design requirements. Preliminary experiments showed no major improvements with larger spatial resolutions. The volumes cover an FOV of $3\,\mathrm{mm} \times 3\,\mathrm{mm} \times 3.5\,\mathrm{mm}$. Overlaid example volumes are shown in Fig. 6.36.

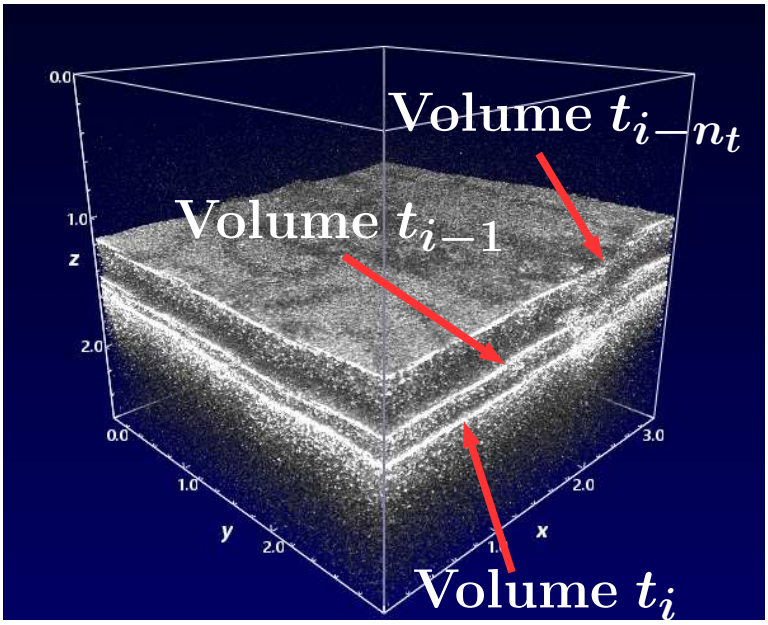


Fig. 6.36: Three overlaid OCT volumes from three time steps.

In addition to using streams of the full 3D image volumes, we consider 2D and 3D representations of the tissue surface. Typically, the largest fraction of the infrared light is reflected at the tissue surface. Hence, we employ a maximum intensity projection P to localize the tissue surface in the image volumes, as introduce in Section 6.5 for OCT-based pose estimation. The surface is either represented by a 2D projection of depth values or by a discrete 3D point cloud, where voxels representing the surface are set to 1, and all other voxels are set to 0. These pseudo volumes are shown in Figure 6.37. To encode the surface as accurately as possible, extraction is performed using the raw OCT volumes with full depth resolution. The resulting depth maps have a size of 32×32. For the pseudo volumes, we reproject the depth maps into volumes of size $32 \times 32 \times 32$ to match our downsampled volumes in terms of size.

The 2D depth images are a small, efficient representation; however, subsurface information is lost through the projection. They are similar to time-of-flight depth images that have been previously used for force estimation [147, 321]. The 3D pseudo volumes encode surface information similar to their 2D counterpart, but they are processed in a higher-dimensional space.

The phantom was manufactured using translucent silicone mixed with titan dioxide, which leads to light scattering that is similar to the signal found in tissue.

The phantom datasets were acquired with two types of instrument motion along the needle's shaft. First, we performed a sinusoidal movement with varying amplitudes and frequencies, which results in a smooth pattern. For this dataset we acquired $32\,000$ samples, which we partitioned into sets of $24\,000$, 2600 and 5400 samples for training, validation, and testing, respectively (dataset $\mathcal{V}_1$). Second, we considered a movement based on cubic splines. We randomly sampled depth values from $[d_{p_0}, d_{p_{max}}]$ where d_{p_0} is the point of tissue contact and $d_{p_{max}}$ the maximum deformation depth along the needle shaft. While being more random, this also provides smooth curves with predictable

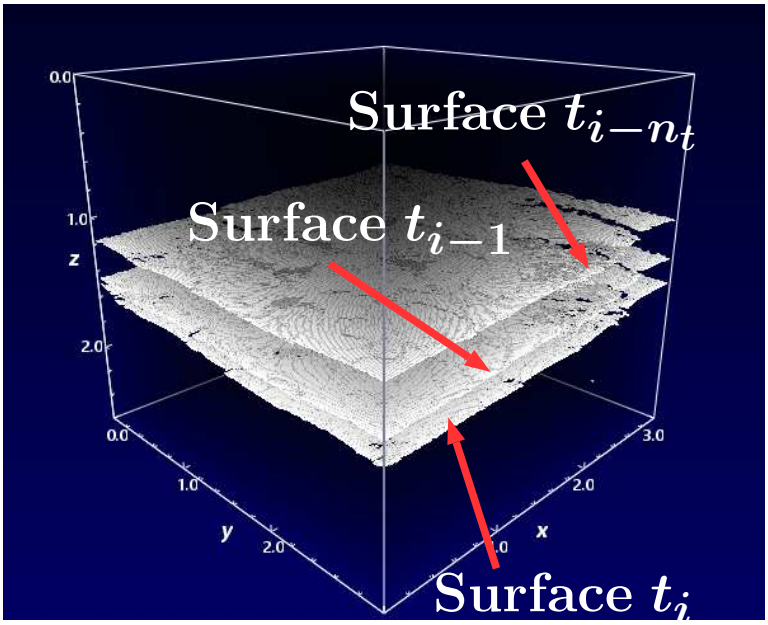


Fig. 6.37: Three pseudo OCT volumes from three time steps.

patterns. Here, we acquired $38\,000$ samples which we partition into sets of $31\,000$, 2900 and 4100 samples for training, validation, and testing, respectively (dataset $\mathcal{V}_2$). For both types of acquisition, we acquired all data across multiple experiments with ≈ 2500 samples each. For each experiment, a new amplitude and frequency (dataset $\mathcal{V}_1$) or depth pattern (dataset $\mathcal{V}_2$) was defined. Depths vary from $1\,\mathrm{mm}$ to $3\,\mathrm{mm}$ and frequencies vary from $3\,\mathrm{Hz}$ to $6\,\mathrm{Hz}$. We also varied the instrument tip's orientation and relative position with respect to the ROI for each experiment to induce more variation and to avoid overfitting to a particular instrument location or orientation. Similarly, we changed the ROI between experiments to ensure different speckle patterns within the datasets. All data splits are separated by experiments. Thus, all samples of an experiment are only part of one data split. Summarized, datasets $\mathcal{V}_1$ and $\mathcal{V}_2$ represent different types of motion for a homogeneous phantom.

The tissue dataset (dataset $\mathcal{V}_3$) represents a scenario closer to application that we use to test robustness and variation, see Figure 6.38 for an example. In total, we acquired $30\,000$ samples for this dataset. We rely on the same movement patterns as for dataset $\mathcal{V}_2$. Again, we vary the instrument tip's orientation and its relative position with respect to the ROI. Also, we vary the ROI between experiments.

In contrast to the phantom, different ROIs can be more different in terms of their sub-surface structure relating to fat and muscle composition. Thus, the selection of a single validation/test ROI could induce a bias. Therefore, we perform 20-fold cross-validation, where one experiment corresponding to one ROI is excluded in each experiment. In summary, the tissue dataset is closer to applications and allows us to study the robustness of our approach.

The OCT volumes are acquired at $60\,\mathrm{Hz}$, and the force data is acquired at $500\,\mathrm{Hz}$. We synchronize both data streams based on timestamps, and we assign a force measurement to each OCT volume. We transform the forces from the force sensor's coordinate frame

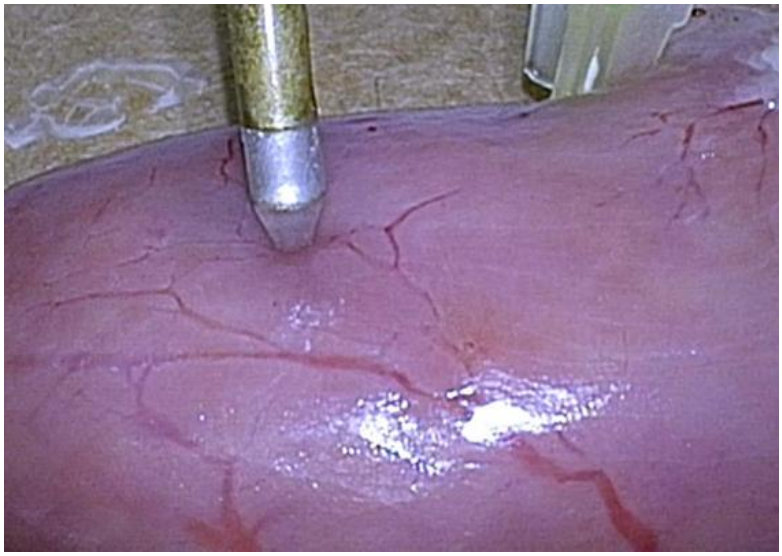


Fig. 6.38: An example image for the tissue being deformed by the needle tool.

to a coordinate frame located at the needle tip using the known spatial dimensions of the needle and force sensor. Then, we take the magnitude of the resulting force vector as the final label for our learning problem.

Deep Learning Models. We implement several different models that are based on the spatial and spatio-temporal deep learning architectures that we introduced in Chapter 4. For each model, we consider a 3D-ST version for comparison that matches the 4D-ST versions in terms of structure. The CNN part of each architecture is built on the ResNet principle. We use vanilla ResNet blocks without bottlenecks or other variations. While building upon the same ResNet backbone, each architecture uses a very different way of processing the temporal dimension together with the spatial dimension.

RN4D is an implementation of the CNN4D model we proposed in Section 4.1.4. The model processes feature maps of size $\mathrm{x} \in \mathbb{R}^{n_t \times n_h \times n_w \times n_d \times n_c}$, excluding the batch dimension. n_c is the channel dimension of the feature map. The final architecture starts off with a normal convolutional layer, followed by 5 ResBlocks with a total spatial output stride of $r_o = 16$. The initial feature maps size is $n_c = 16$, which is doubled each time we halve the spatial dimension using a convolution with spatial stride $r = 2$. As we keep n_t small, the temporal dimension is not reduced by strides.

For models using 3D-ST data, we use the same architecture with 3D convolutions (RN3D-ST). The model is an implementation of a ResNet-based 3D CNN, compare Table 4.2. Here, feature maps are of size $\mathrm{x} \in \mathbb{R}^{n_t \times n_h \times n_w \times n_c}$. To match the models' capacity, we also consider deeper (RN3D-ST-d) and wider (RN3D-ST-w) 3D variants. For the deeper version, we use a total of 9 ResBlocks instead of 5 in the baseline model. For the wider version, we double the number of feature maps by using $n_c = 32$ instead of $n_c = 16$ for the initial convolutional layer. We use kernels of size $k_j = 3$ for every dimension.

For spatial approaches using 3D-S and 2D-S data, we consider an RN3D-S and RN2D-S with 3D and 2D convolutional operations, respectively. In terms of structure, the CNNs

are similar to the ST models; however, they receive and process spatial information only. Thus, RN2D-S processes feature maps of size $\mathrm{x} \in \mathbb{R}^{n_h \times n_w \times n_c}$ and RN3D-S processes feature maps of size $\mathrm{x} \in \mathbb{R}^{n_h \times n_w \times n_d \times n_c}$.

facRN4D is a more efficient variant of RN4D, which uses factorized convolutions that follows the facCNN4D principle that we introduced in Section 4.1.4 for 4D CNNs. The model processes 4D volumes of size $\mathrm{x} \in \mathbb{R}^{n_t \times n_h \times n_w \times n_d \times n_c}$. In each ResBlock, each convolution is split into two convolutions with spatial and temporal kernels. For 3D-ST data, we employ a facRN3D model, where spatial and temporal convolutions are also split, similar to RN3D-ST.

RN3D-GRU is an implementation of the CNN-GRU principle that we describe in Section 4.3. The model processes full 4D volumes $\mathrm{x} \in \mathbb{R}^{n_t \times n_h \times n_w \times n_d \times n_c}$. The CNN part has the same structure as RN3D-S described above. The features produced by the CNN are pooled and then fed into two GRU layers with 1024 feature maps each. The GRUs are augmented by recurrent dropout and recurrent batch normalization. The entire architecture is trained end-to-end.

For 3D-ST data, we design RN2D-GRU, where the initial ResNet for spatial processing uses 2D convolutions to process the 2D depth images. After pooling the spatial representation into a feature vector, the same temporal processing with two GRU layers is applied.

RN3D-cGRU is an implementation of the CNN-cGRU architecture concept that we introduced in Section 4.3. This model also processes 4D data $\mathrm{x} \in \mathbb{R}^{n_t \times n_h \times n_w \times n_d \times n_c}$. The model comes with the same structure as RN3D-GRU, except that convolutional GRUs are used instead of normal GRUs. In terms of implementation, there is no GAP layer after the last CNN's last layer and the convolutional GRU directly receives a spatial tensor for processing. After the convolutional GRU layer, GAP is applied and the output layer follows.

For 3D-ST data, we also employ a RN2D-cGRU variant where the ResNet-based CNN is replaced by a 2D CNN with 2D convolutional layers, similar to the other models for 3D-ST data.

cGRU-RN3D is an implementation of our proposed cGRU-CNN architecture for 4D data $\mathrm{x} \in \mathbb{R}^{n_t \times n_h \times n_w \times n_d \times n_c}$, see Section 4.3. The convolutional GRU with 32 feature maps also employs recurrent batch normalization, and the gates use 3D convolutions. The following ResNet uses normal 3D convolution operations and performs spatial processing only. Compared to RN4D and facRN4D, this architecture also keeps the temporal and the spatial processing parts separate. The key difference to RN3D-GRU is that the order of temporal and spatial processing is reversed. For 3D-ST data, we use a cGRU-RN2D where all 3D convolution operations are replaced by 2D variants. All other properties are the same as for cGRU-RN3D.

Training. For all model variants, we chose the learning rate and batch size individually based on validation performance. For 3D architectures, we use a batch size of $N_b = 16$ and learning rate of $\alpha_{lr} = 5 \times 10^{-4}$. For 4D architectures, we use a batch size of $N_b = 8$ and a learning rate of $\alpha_{lr} = 2.5 \times 10^{-4}$. During validation experiments, we did not observe any overfitting, which could require techniques such as early stopping. Weights are initialized using a truncated normal distribution with a standard deviation $std = 0.01$, where values are redrawn if their magnitude is larger than twice the standard deviation. We train for $N_e = 100$ epochs using the Adam algorithm with the recommended

Tab. 6.31: Comparison of 3D-ST and 4D-ST architectures for both datasets. The MAE is given in mN. The best values for each dataset are marked bold.

		MAE	**rMAE** (10^{-3})	**PCC** (%)
Dataset $\mathcal{V}_1$	RN4D	11.9 ± 12.3	42.7 ± 44.5	**99.84**
	facRN4D	12.3 ± 13.9	44.2 ± 49.7	99.82
	RN3D-GRU	19.1 ± 23.9	68.4 ± 86	99.47
	cGRU-RN3D	$\mathbf{11.7 \pm 15.0}$	$\mathbf{42.3 \pm 54.0}$	99.80
	RN3D-ST	21.9 ± 33.6	78.8 ± 120	98.98
	facRN3D	21.4 ± 34.7	76.8 ± 124	98.93
	RN2D-GRU	30.8 ± 36.4	110 ± 130	98.59
	cGRU-RN2D	23.3 ± 36.2	83.7 ± 120	98.85
Dataset $\mathcal{V}_1$	RN4D	12.0 ± 11.2	59.5 ± 55.9	99.68
	facRN4D	13.5 ± 13.2	67.1 ± 65.4	99.58
	RN3D-GRU	26.3 ± 23.9	130 ± 118	98.46
	cGRU-RN3D	$\mathbf{10.7 \pm 11.1}$	$\mathbf{53.2 \pm 55.5}$	**99.71**
	RN3D-ST	25.4 ± 30.6	125 ± 152	98.04
	facRN3D	24.8 ± 31.3	122 ± 155	98.09
	RN2D-GRU	35.9 ± 30.8	178 ± 153	97.21
	cGRU-RN2D	25.4 ± 29.0	126 ± 144	98.17

standard parameters. During training, we track exponential moving averages of all trainable parameters with a decay rate of $d_c = 0.999$. Thus, during the evaluation, we use a more consistent moving average of our model parameters instead of a point estimate at the last iteration. The loss function is the MSE. We implement our models and training environment in Tensorflow [1]. Training is performed on NVIDIA GEFORCE 1080Ti graphics cards.

Experiment Overview. All model performances are reported the test set of dataset $\mathcal{V}_1$ and $\mathcal{V}_2$ or the CV result for dataset $\mathcal{V}_3$. First, we compare our proposed 4D-ST architecture designs and evaluate their performance with respect to their 3D-ST counterparts. We relate model performance to architecture efficiency in terms of the number of trainable parameters. For better interpretability, we provide a regression plot of the entire force range. Also, we study robustness and variation with tissue dataset $\mathcal{V}_3$. Second, we investigate deep learning models using 2D-S, 3D-S, 3D-ST, and 4D-ST data. We consider model versions with different capacity to account for the natural increase in the number of parameters for models with higher-dimensional data processing. Furthermore, we use our ps-4D-ST data derived from depth images to investigate the advantages of 4D-ST data processing. Last, we provide results for different force forecasting horizons $n_f \in \{0, 1, 2, 3, 4\}$ and different lengths of temporal history $n_t \in \{2, 4, 6, 8\}$. As a baseline, all spatio-temporal models use $n_t = 6$. We report the MAE in mN as an absolute metric and the rMAE and PCC as relative metrics. For the MAE and rMAE we provide the standard deviation. We test for significant differences in the median of the absolute errors using the Wilcoxon signed-rank test with a significance level of $\alpha = 0.05$.

Tab. 6.32: Comparison of different GRU types and their position within the architecture. The MAE is given in mN. The best values for each dataset are marked bold.

		MAE	**rMAE** (10^{-3})	**PCC** (%)
Dataset $\mathcal{V}_1$	RN3D-cGRU	16.8 ± 19.3	60.4 ± 81	99.66
	RN3D-GRU	19.1 ± 23.9	68.4 ± 86	99.47
	cGRU-RN3D	$\mathbf{11.7 \pm 15.0}$	$\mathbf{42.3 \pm 54.0}$	**99.80**
	RN2D-cGRU	27.5 ± 35.7	98.2 ± 126	98.71
	RN2D-GRU	30.8 ± 36.4	110 ± 130	98.59
	cGRU-RN2D	23.3 ± 36.2	83.7 ± 120	98.85
Dataset $\mathcal{V}_2$	RN3D-cGRU	21.6 ± 20.0	110 ± 96.3	99.04
	RN3D-GRU	26.3 ± 23.9	130 ± 118	98.46
	cGRU-RN3D	$\mathbf{10.7 \pm 11.1}$	$\mathbf{53.2 \pm 55.5}$	**99.71**
	RN2D-cGRU	29.9 ± 30.8	156 ± 160	97.61
	RN2D-GRU	35.9 ± 30.8	178 ± 153	97.21
	cGRU-RN2D	25.4 ± 29.0	126 ± 144	98.17

Results

4D-ST and 3D-ST Architectures. The results for all 3D-ST and 4D-ST architectures are shown in Table 6.31. Comparing 4D-ST architectures, across both datasets, RN4D and cGRU-RN3D perform best. For dataset $\mathcal{V}_2$, cGRU-RN3D performs better as there is a significant difference in the absolute errors. Also, cGRU-RN3D performs significantly better than facRN4D across both datasets. Between all models and datasets, RN3D-GRU performs worst. In general, the relative metrics are slightly lower for dataset $\mathcal{V}_2$ with spline-based trajectories. Comparing 3D-ST and 4D-ST architectures, the latter significantly outperform their counterparts across all our proposed architectures.

Next, we show how model capacity relates to performance, see Figure 6.39. While RN4D and cGRU-RN3D perform similarly, the latter comes with substantially fewer trainable parameters. The 3D-ST models perform significantly worse; however, their capacity in terms of the number of parameters is also lower.

Furthermore, we consider a comparison between different types of GRUs and their position in the architecture, see Table 6.32. Across 4D and 3D spatio-temporal data, we observe that cGRU-RN3D performs best. Using a convolutional GRU instead of a GRU at the end of the architecture leads to minor performance improvements.

For qualitative interpretation, we provide a regression plot between predicted and target force values, see Figure 6.40. Across the entire force range, predictions closely match the targets. For larger force values closer to $1\,\mathrm{N}$, there are some outliers.

To highlight the advantage of 4D-ST architectures, we also consider a tissue experiment with 20-fold CV to assess variation and robustness, see 6.41. In general, the MAE is lower while the relative metrics are worse compared to the phantom datasets. The size of the boxplots shows that there is some variation across CV folds. Comparing 4D-ST and 3D-ST architectures, the performance difference is similar to the phantom datasets with 4D-ST models performing significantly better. The regression plot in Figure 6.42 shows the smaller force range for the tissue dataset. Again, predictions match the targets

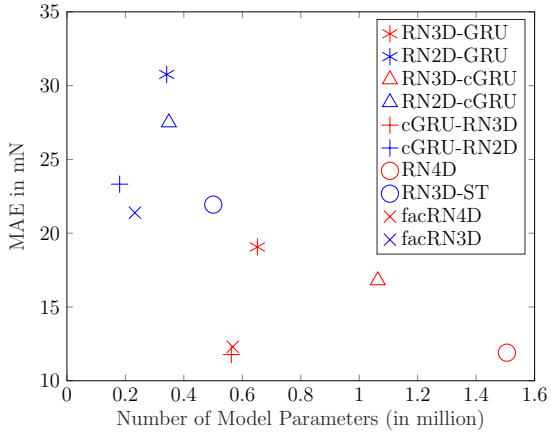


Fig. 6.39: All 4D-ST architectures (red) and their 3D-ST counterpart (blue) in comparison to their MAE and their number of trainable parameters. Models in the lower-left corner have a lower error and fewer parameters.

well with a high R^2-value.

Multi-Dimensional Data Representations. Next, we study how deep learning models with spatial data compare to models using spatio-temporal data. The results are shown in Table 6.33. The 2D-S models perform worst. Adding a temporal dimension (3D-ST) improves performance substantially. The performance increase is statistically significant across all datasets and models. Using volumes (3D-S) for learning performs better than 2D-S or 3D-ST data. Increasing the lower-dimensional models' capacity leads to minor performance improvements. Overall, the 4D-ST deep learning model performs best. The performance difference in terms of the median of the absolute errors is statistically significant for all models and datasets.

Moreover, we consider ps-4D-ST as a higher-dimensional encoding of the depth images. The results are shown in Table 6.34. Overall, the ps-4D-ST models' performance is much closer to the 3D-ST models than to the 4D-ST models. However, the performance difference is statistically significant across both datasets for RN4D, facRN4D, and cGRU-RN3D. For RN3D-GRU, the performance deteriorates.

Temporal Information and Force Forecasting. Last, we investigate the temporal properties of our two top-performing models RN4D and cGRU-RN3D. The results for variations of the temporal history n_t and forecasting horizon n_f are shown in Figure 6.43. For each combination of n_t and n_f, a model was trained. Considering force estimation ($n_f = 0$), adding more temporal history improves performance. The largest improvement can be observed between $n_t = 2$ and $n_t = 4$. Then, performance tends to saturate. Similar observations can be made for $n_f > 0$ and increasing values for n_t. When increasing n_f, forecasting works well as the MAE only increases by $\approx 25\,\%$ (dataset $\mathcal{V}_1$) and $\approx 39\,\%$ (dataset $\mathcal{V}_2$) between $n_f = 0$ and $n_f = 4$ for $n_t \in \{4, 6, 8\}$ with cGRU-RN3D. Comparing RN4D and cGRU-RN3D, both show similar trends for increasing n_f. However, for increasing n_t, cGRU-RN3D shows a consistent performance improvement for all n_f and both datasets, while RN4D shows varying results for $n_t \in \{4, 6, 8\}$ across all n_f.

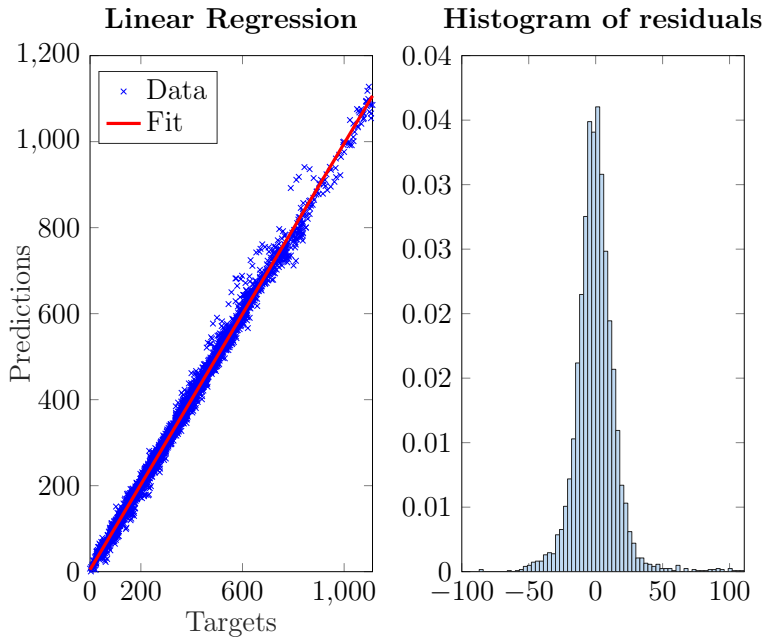


Fig. 6.40: Linear regression plot (left) and the histogram of residuals (right) between targets and predictions, given in mN. The relationship is significant ($p < 0.05$) with an R^2-value of 0.994. The cGRU-RN3D model's predictions and dataset $\mathcal{V}_2$ were used.

Last, we also provide qualitative results for predicting forces four time steps into the future with different lengths of history, see Figure 6.44. For a longer history, predictions are more consistent. In particular, sudden changes in force trends lead to overshooting and error spikes.

Discussion

We study volume-based force estimation with deep learning methods in the context of different data representations and spatio-temporal deep learning concepts. Here, 4D spatio-temporal data processing is particularly interesting as higher-dimensional data promises more context for learning while also being very challenging to handle. Overall, our cGRU-RN3D, which performs temporal processing first, followed by spatial processing, shows the best performance. This is particularly interesting as a lot of spatio-temporal deep learning approaches in the natural image domain perform spatial feature extraction first, followed by temporal processing of the spatial features (RN3D-GRU and RN3D-cGRU) [19]. However, our observations match our results for other tasks such as needle-based force estimation. In addition, RN4D shows performance that is similar to cGRU-RB3D; however, the model comes with substantially more parameters. Decomposing the convolutions into a spatial and a temporal part with facRN4D leads to a lower model capacity but also slightly lower performance. Thus, overall, cGRU-RN3D represents a high-performing and efficient deep learning architecture for 4D OCT-based force estimation.

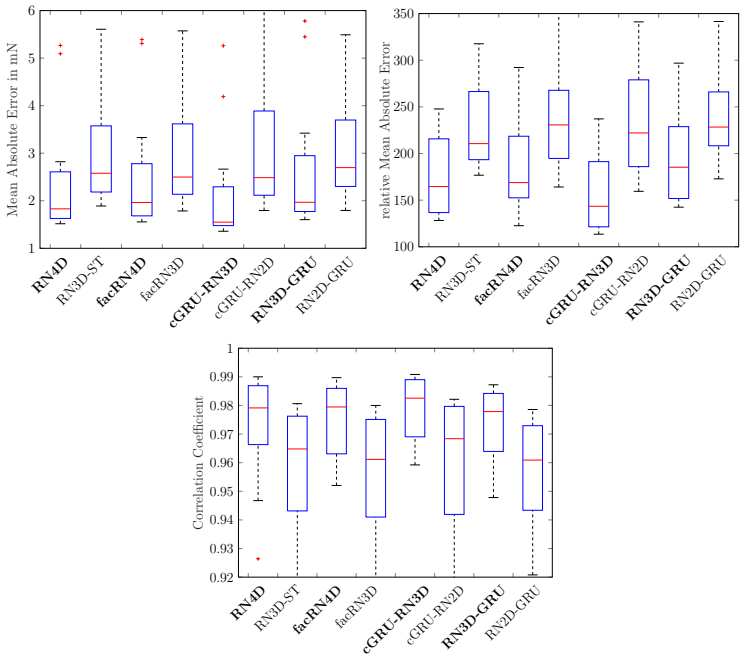


Fig. 6.41: Boxplots for the three metrics for all 4D-ST (marked bold) and 3D-ST architectures are shown. Each boxplot is generated with 20 values from 20-fold CV with our tissue dataset $\mathcal{V}_3$.

To highlight the value of 4D spatio-temporal data processing, we compare our approaches to their 3D spatio-temporal counterparts. These models use depth images extracted from volumes, which is similar to previous force estimation approaches relying on depth representations from time-of-flight cameras [147]. We find that across all our concepts for spatio-temporal processing, the 4D models consistently outperform their respective 3D counterpart with a statistically significant performance difference both for phantom and tissue data. Considering absolute and relative metric results, absolute MAE values are slightly larger for phantom experiments compared to tissue experiments due to a larger force range, as shown in Figure 6.40. While this force range is typical for applications such as lung tumor localization [329], other applications such as retinal microsurgery require a smaller force range and higher resolution [183]. Our tissue experiments demonstrate that our approach is also scalable to smaller force ranges, see Figure 6.42. Due to the larger variability between tissue samples compared to phantom data, estimation appears to be more difficult as relative metrics are slightly lower than for the phantom experiments. Nevertheless, regarding absolute values, the MAE is around $2\,\mathrm{mN}$ for our 4D-ST architectures. This suggests the suitability of our approach for a variety of clinical applications where force sensing can be helpful and different force ranges are required [492].

All in all, our 4D deep learning models consistently outperform 3D approaches across multiple datasets. While other 4D spatio-temporal CNN applications with different medical imaging modalities [92,210] did not show improvements over lower-dimensional approaches, we highlight the value of utilizing full 4D information when it is available.

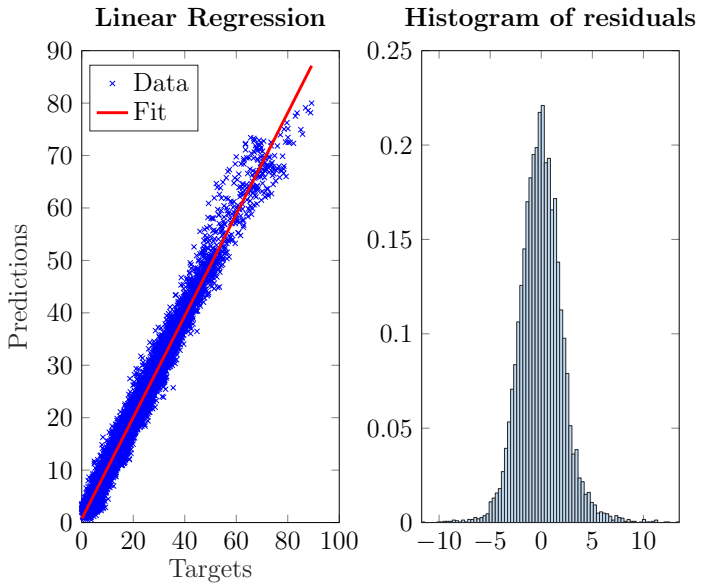


Fig. 6.42: Linear regression plot (left) and the histogram of residuals (right) between targets and predictions, given in mN. The relationship is significant ($p < 0.05$) with an R^2-value of 0.973. The cGRU-RN3D model's predictions and dataset $\mathcal{V}_3$ were used.

For OCT data, this is particularly interesting, since devices for 4D OCT imaging with high temporal resolution are available [449]. However, there have been no studies utilizing the full 4D information in deep learning models. Thus our insights could improve other medical OCT applications where 4D OCT data is available.

Given the same high-level architecture, deep learning models processing lower-dimensional data contain fewer parameters, thus, having a lower capacity. Therefore, we also evaluate the four CNN-based lower-dimensional data processing techniques with increased model capacity. Even when adding more layers to the models or increasing the number of feature maps, the lower-dimensional models' performance difference is significantly lower than RN4D's performance. This demonstrates that there is an inherent advantage of using 4D data, which is not connected to the natural increase in model capacity for higher-dimensional data processing methods.

The high performance of 4D-ST models leads to the question of whether the advantage is caused by richer subsurface information being present or processing in a higher-dimensional space. The depth maps we use have a smaller size than the volumes, which makes an immediate comparison to larger volumes difficult. Therefore, we disentangle the aspects of input size and subsurface information by using pseudo volumes that encode the depth images in a volume that has the same size as our full OCT volumes. These pseudo 3D volumes encode the same information as the 2D depth images. Overall, we find that pseudo 4D-ST data processing performance is closer to 3D-ST data than 4D-ST data, which indicates that the majority of the advantage can be traced back to richer information. However, using pseudo 4D-ST data leads to a statistically significant performance improvement for three of our four architectures

Tab. 6.33: Comparison of models using 2D-S, 3D-S, 3D-ST and 4D-ST data. *-W* denotes models with more feature maps per layer, and *-D* describes models with more layers. The MAE is given in mN.

		MAE	**rMAE** (10^{-3})	**PCC** (%)
Dataset $\mathcal{V}_1$	RN2D-S	30.7 ± 39.6	110 ± 142	98.38
	RN2D-S-W	28.6 ± 38.6	103 ± 139	98.51
	RN2D-S-D	31.0 ± 41.8	111 ± 150	98.24
	RN3D-S	16.2 ± 22.6	58.0 ± 81.0	99.55
	RN3D-S-W	15.2 ± 21.7	54.7 ± 78.1	99.59
	RN3D-S-D	14.9 ± 21.6	53.7 ± 77.5	99.59
	RN3D-ST	21.9 ± 33.6	78.8 ± 120	98.98
	RN3D-ST-W	21.1 ± 32.9	75.6 ± 118	99.04
	RN3D-ST-D	22.5 ± 34.8	80.1 ± 129	98.85
	RN4D	$\mathbf{11.9 \pm 12.3}$	$\mathbf{42.7 \pm 44.5}$	**99.84**
Dataset $\mathcal{V}_2$	RN2D-S	35.1 ± 34.0	174 ± 168	97.08
	RN2D-S-W	33.2 ± 31.5	165 ± 156	97.41
	RN2D-S-D	35.3 ± 33.9	175 ± 158	97.04
	RN3D-S	19.7 ± 22.1	98.0 ± 110	98.94
	RN3D-S-W	18.4 ± 20.8	91.3 ± 103	99.07
	RN3D-S-D	19.2 ± 21.4	95.2 ± 106	99.00
	RN3D-ST	25.4 ± 30.6	125 ± 152	98.04
	RN3D-ST-W	23.3 ± 28.7	116 ± 142	98.32
	RN3D-ST-D	23.1 ± 30.0	115 ± 149	98.25
	RN4D	$\mathbf{12.0 \pm 11.2}$	$\mathbf{59.5 \pm 55.9}$	**99.68**

across both datasets. This indicates that there is an inherent advantage of using higher-dimensional data representations for our problem. This insight comes with important implications for other force estimation approaches. 4D data processing is not only advantageous when native 4D data is available, but performance can also be improved by transforming streams of 2D depth representations into a higher-dimensional space. Thus, previous force estimation approaches using RGBD-based 3D-ST data could potentially benefit from our insights on pseudo 4D-ST data.

Last, we also provide a detailed analysis of the temporal dimension, see Figure 6.43. Previous approaches for force estimation have used temporal information successfully as well [27, 147]; however, they have not attempted force forecasting. Given a smooth change in forces and deformation, forces should be predictable, which we demonstrate successfully for different forecasting horizons. Due to the volatile nature of movement during surgery, long-term forecasting is not reasonable. However, for our time horizon of $16\,\mathrm{ms}$ to $64\,\mathrm{ms}$, forecasting should be possible. There are limitations to predictability, in particular, for unexpected changes. We demonstrate this aspect in Figure 6.44, where sudden changes in the force trend visibly impact forecasting. At the same time, a longer history appears to help in obtaining more consistent estimates. For clinical application, force forecasting could be useful in the context of striving towards fully automated

Tab. 6.34: Comparison of 3D-ST and pseudo 4D-ST data representations for both datasets. Pseudo 4D-ST models are labeled by the prefix ps. The MAE is given in mN. The best values for each dataset are marked bold.

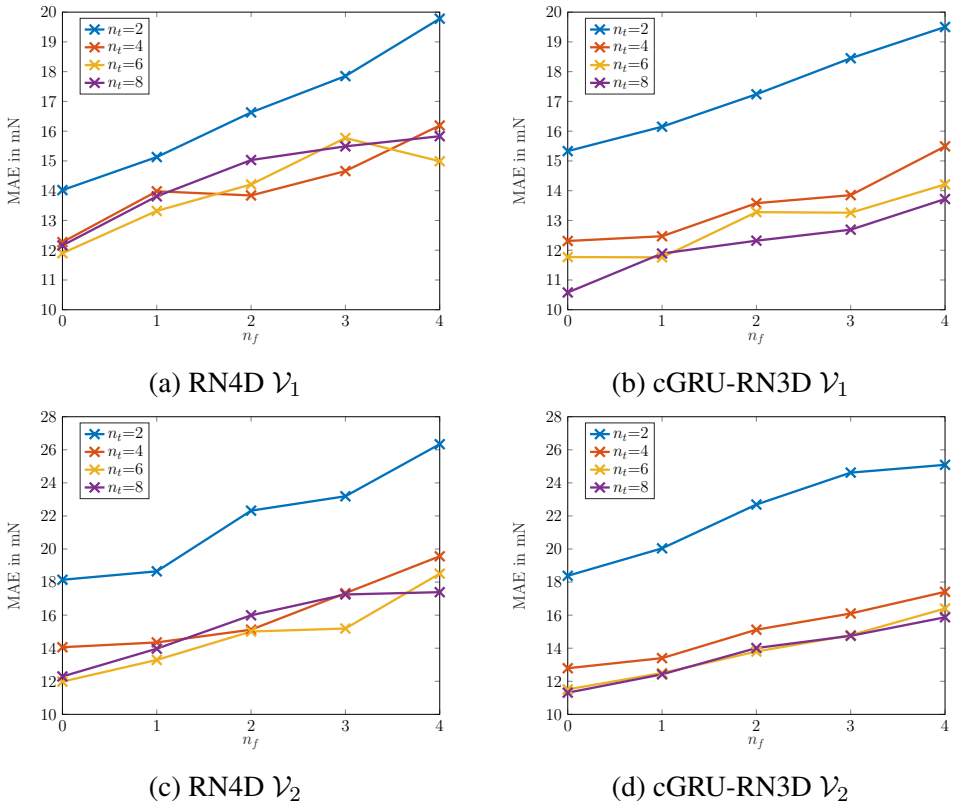
		MAE	**rMAE** (10^{-3})	**PCC** (%)
Dataset $\mathcal{V}_1$	ps-RN4D	18.7 ± 24	67.1 ± 91.3	99.42
	ps-facRN4D	17.9 ± 23.0	64.1 ± 84.5	99.43
	ps-RN3D-GRU	43.3 ± 50.8	155 ± 182	98.48
	ps-cGRU-RN3D	$\mathbf{15.0 \pm 20.8}$	$\mathbf{53.8 \pm 74.8}$	**99.59**
	RN3D-ST	21.9 ± 33.6	78.8 ± 120	98.98
	facRN3D	21.4 ± 34.7	76.8 ± 124	98.93
	RN2D-GRU	30.8 ± 36.4	110 ± 130	98.59
	cGRU-RN2D	23.3 ± 36.2	83.7 ± 120	98.85
Dataset $\mathcal{V}_2$	ps-RN4D	17.8 ± 15.9	88.2 ± 78.7	99.30
	ps-facRN4D	$19.2 \pm 17.9)$	95.2 ± 88.7	99.15
	ps-RN3D-GRU	37.8 ± 35.7	188 ± 177	97.05
	ps-cGRU-RN3D	$\mathbf{15.4 \pm 15.8}$	$\mathbf{76.5 \pm 78.6}$	**99.40**
	RN3D-ST	25.4 ± 30.6	125 ± 152	98.04
	facRN3D	24.8 ± 31.3	122 ± 155	98.09
	RN2D-GRU	35.9 ± 30.8	178 ± 153	97.21
	cGRU-RN2D	25.4 ± 29.0	126 ± 144	98.17

robotic interventions [184]. Highly accurate force estimates could allow for precise robot control. In particular, force forecasting would be useful for safety features in robot-assisted interventions as an automatic system could stop before certain force thresholds with a risk of trauma are exceeded [188].

Regarding quantitative results, we find that an increasing length of the history n_t leads to improvements for RN4D and cGRU-RN3D both for estimation and forecasting, which is particularly consistent for cGRU-RN3D. This indicates that the model is very effective at utilizing temporal information.

Summary

We address the problem of volume-based force estimation with 2D, 3D, and 4D OCT data and deep learning models. Regarding our first research question and data representations, we find that using 4D data tends to perform best. Also, using 3D volumes instead of 2D depth maps significantly improves performance. Notably, encoding 2D depth maps as 3D volumes also improves force estimation performance, indicating that higher-dimensional processing is generally advantageous. Using a temporal history consistently improves performance, in particular for force forecasting. With respect to our second research question on deep learning models, we find that our proposed cGRU-CNN principle is preferable over other convolutional and recurrent architecture concepts. Thus, performing temporal processing before spatial processing appears to be advantageous. Regarding CNN models, efficient design through factorization significantly reduces the number of parameters at the cost of slightly reduced performance.



(a) RN4D $\mathcal{V}_1$

(b) cGRU-RN3D $\mathcal{V}_1$

(c) RN4D $\mathcal{V}_2$

(d) cGRU-RN3D $\mathcal{V}_2$

Fig. 6.43: Force estimation and prediction results for varying n_t and n_f with two datasets $\mathcal{V}_1$ and $\mathcal{V}_2$ with architectures RN4D and cGRU-RN3D. For each setting, a new model is trained and evaluated.

6.9 Summary

In this chapter, we presented our experimental results, where we applied our adapted and proposed deep learning methods to multi-dimensional medical applications. For each application, we outlined the problem-specific motivation in terms of our research questions, described datasets for training and evaluation, and detailed the implementation of our deep learning methods. Then, we presented both quantitative and qualitative results and discussed problem-specific insights.

We started with lower-dimensional 1D and 2D OCT image data. 1D A-Scans can be acquired with small-scale optical fibers that are useful for computer-assisted interventions. Here, we study the use of 1D spatial and 2D spatio-temporal data with our deep learning methods. We implement 1D CNNs, 2D CNNs, and several convolutional-recurrent architectures that were handcrafted for this problem. We find that the use of temporal information improves performance and that our newly proposed cGRU-CNN concept outperforms other methods. Next, we investigate the use of 1D spatial and 2D spatial OCT data for the problem of retina layer segmentation. Here, we find that higher-dimensional 2D data improves performance once again. We implement our NAS approach for this problem and find that learned architectures transfer well across dimensions.

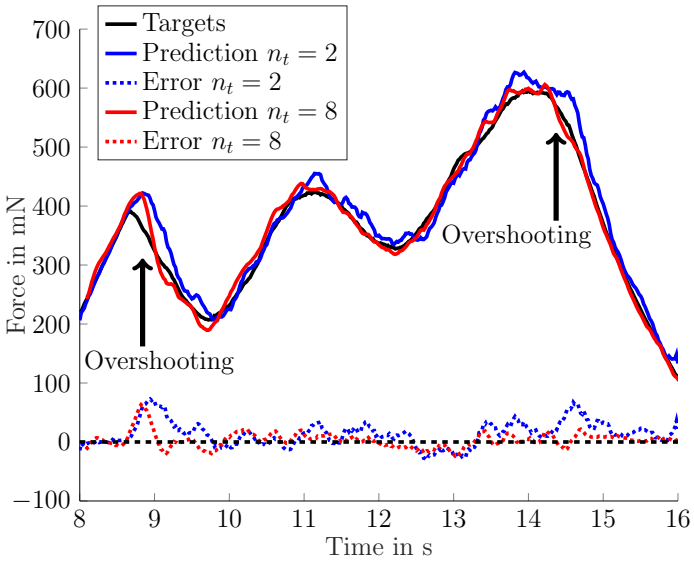


Fig. 6.44: Example of the force trend when forecasting four time steps into the future using cGRU-RN3D and a history of $n_t = 2$ or $n_t = 8$ time steps. At sudden force trend changes, we can observe overshooting of the predictions.

Regarding 2D OCT data, there are also different spatial data representations, for example, in IVOCT imaging. We considered both 2D Cartesian and 2D polar data representations for the problem of plaque classification. In terms of deep learning methods, we explored transfer learning with 2D CNN architectures, adopted from the natural image domain. We provided evidence that using Cartesian images is advantageous, enabled by representation-specific data augmentation and that using both data representations in a single model can boost performance further. Also, we found that using pretrained weights is advantageous, but full-fine tuning is required. Furthermore, we showed that our Siamese architecture concept, coupled with our multi-dimensional transfer learning strategy, leads to a model that improves performance with two data representations being used.

Another application in cardiac imaging is MRI-based LVQ. We explored the use of 2D spatial MRI slices in comparison to 3D spatio-temporal data for the assessment of the heart's geometric parameters along a cardiac cycle. Similar to our IVOCT application scenario, we employed 2D CNNs, adopted from the natural image domain, combined with transfer learning. For the extension to 3D data, we used a straightforward adaptation by isotropically extending all operations in a 2D CNN by a third dimension. We enabled this method with our multi-dimensional transfer learning strategy and weight scaling for stable training. We found that using higher-dimensional data improves performance, given our multi-dimensional transfer learning strategy is employed. Furthermore, we showed that the type of architecture being employed also has a significant impact on performance.

OCT-based pose estimation is a problem that is also concerned with 3D image data. For this application, we proposed a new annotation strategy with a robot that allows for

acquiring large datasets for supervised learning. In terms of image data, we considered both 2D depth representations, as previously used, and full 3D spatial volumes. As real-time processing would be required for application, we employed our efficiently engineered 3D CNN models that are based on 2D CNN concepts from the natural image domain. We showed that using volumetric data is advantageous for OCT-based pose estimation, both quantitatively and qualitatively. Moreover, we demonstrated that carefully handcrafting architectures helps to improve performance; however, more simple design principles could also lead to that goal.

Pose estimation can be extended to motion estimation by considering a sequence of poses that needs to be tracked. Motion estimation is required for motion compensation or motion prediction. Using 3D OCT volumes for individual poses leads to 4D spatio-temporal data for sequences of poses. First, we considered a simplified case where a motion vector between two states is estimated with 3.5D data. Second, we used full 4D spatio-temporal data for motion estimation. We implemented our proposed, efficiently designed, 4D CNN architectures, as well as convolutional-recurrent architectures using a DenseNet backbone for this problem. Overall, we found that the use of higher-dimensional data is advantageous and that our proposed 4D CNN models perform very well.

3.5D and 4D data is also present in the problem of MRI-based lesion activity segmentation. Here, we try to detect new and enlarged lesions that appeared between two longitudinal MRI scans. When addressing the problem with 3.5D data, we employed our Siamese CNN architectures, augmented by attention-guided interaction modules, to improve lesion localization. In the 4D case, we implemented our cGRU-CNN-U architecture with a ResNet backbone. For the 3.5D case, we provided evidence that using our attention-guided interaction modules improves performance over plain Siamese CNNs. For the 4D case, we found that using more time points is preferable and that our proposed cGRU-CNN-U architecture outperforms a more simple multi-encoder approach.

Finally, we addressed vision-based force estimation with OCT data. We considered 2D, 3D, and 4D data, including different data representations for each level of dimensionality. We employed our ResNet-based CNNs for 2D, 3D, and 4D image data, as well as, our convolutional-recurrent models for 3D and 4D spatio-temporal image data. Similar to our other applications, we observe that higher-dimensional data performs consistently better across deep learning models. Furthermore, our cGRU-CNN stands out once again by showing high performance while also being light-weight in terms of the number of trainable parameters. In addition, we provide evidence that using higher-dimensional data representations can be advantageous, even if the representation does not capture more context or information.

Overall, we considered a large number of applications for validating our adapted and proposed deep learning methods with multi-dimensional medical image data. For each application, we derived valuable and new insights that could also benefit other medical deep learning applications. In particular, the large number of applications and the variety of deep learning models we employed allows us to obtain more broad and generalizable insights for medical image analysis, which we address in the next chapter.

7 Discussion

In this chapter, we discuss our findings based on our experimental results in Chapter 6. In the previous chapter, we discussed problem-specific insights for each of our applications. Here, we discuss our findings on a broader scale and point out implications derived from observations that repeat across applications. We relate these general findings to the literature we reviewed in Chapter 5.

We start with a discussion of our adapted and proposed deep learning models, which relates to our first research question. First, we propose multiple different approaches for dealing with 1D, 2D, and 3D image data. This includes approaches for handcrafting architectures, direct adoption through multi-dimensional transfer learning, and learning novel architectures through NAS. Second, we propose recurrent-convolutional models that are successful for multiple spatio-temporal learning problems. Third, we consider the challenging problem of 4D deep learning and different concepts for efficient architecture design.

Concerning our second research on multi-dimensional data representation, we then discuss findings across our different applications. For OCT image data, we address how using more or different spatial context affects performs. Also, we highlight the important role of a temporal history for multiple applications. For MRI data, we discuss the use of temporal context both in terms of short-term sequences and long-term longitudinal image sequences.

7.1 Deep Learning Methods

Ever since the massive increase in popularity of CNNs in the natural image domain [262, 279], one of the major research questions in many fields has been: How can we adopt CNNs and replicate their success in our field? In the medical domain, this is of particular interest as there is a vast amount of imaging modalities and problems to solve in the fields of decision support and computer-assisted interventions. A key difference to the natural image domain is the difference in data representations and data dimensionality. While natural color images are 2D, typically with three color channels, medical images can range from 1D to 4D, including edge cases such as 2.5D or 3.5D data. Thus, from the perspective of the medical image domain, deep learning model design boils down to the question of how we need to change 2D CNNs and other methods from the natural image domain to make them work for our specific medical imaging modality and application.

7.1.1 1D, 2D, and 3D Data: Designing, Adapting or Learning CNN Architectures

Moving from the natural image domain's 2D models to models for the medical image domain has been facilitated in different ways in the past. The most popular way of adopting 2D CNNs has been to use a data representation that is also 2D, where model inputs are of size $\mathrm{x} \in \mathbb{R}^{n_h \times n_w \times n_c}$. This allows both for the direct use of models and also the use of pretrained models. Thus, transfer learning can be employed to reuse features that have been learned for tasks in the 2D natural image domain [447]. Typical examples for OCT images include the use of 2D B-Scans for retina layer segmentation [326] or intravascular plaque classification [305]. Similarly, for MRI data, 2D slices are often used, for example for multiple sclerosis lesion segmentation [63] or tumor segmentation [191]. In our work, we also make use of transfer learning with 2D IVOCT data in Section 6.3 and 2D cardiac MRI data in Section 6.4, which generally works better than training from scratch and matches results in the literature [447]. However, for transfer learning with 2D data, architecture adaption does not include multi-dimensional aspects and rather leads to the question of whether using 2D data representations is a reasonable choice, which we discuss in the next section.

First adaptations of 2D CNNs to 3D were largely driven by the idea of capturing more context with volumetric instead of slice-wise processing, for example, in the context of 3D MRI volumes [231]. Here, previous work was largely focused on *designing* new architectures for the 3D domain, loosely inspired by deep learning concepts from the 2D domain. This is largely motivated by the fact that the higher dimensional nature of 3D CNNs is problematic in terms of the number of trainable parameters and memory requirements. In earlier work on 3D CNNs, these 3D models have been described as "a nightmare" to design and train [327]. Thus, very efficient architecture designs are required. So far, most 3D CNN extensions have been focused on segmentation tasks, the most common image analysis task in the medical image domain [111, 231, 301, 575].

In the field of computer-assisted interventions, regression tasks are more common, which require a very different architecture design than segmentation models as the goal is to predict a vector instead of a dense segmentation map. For this type of task, we propose multiple 3D deep learning architectures that bring popular concepts from the 2D domain to 3D, see Section 4.1.2. In particular, we design models that make use of the bottleneck principle [193], Inception-like [477] blocks, and multi-path ResNeXt [554] blocks that have been completely novel for the 3D CNN domain at the time of publication. This addresses the particular need for a more advanced 3D CNN model design that had been advocated in the literature due to simplistic design of previous 3D CNNs [575].

We employ 3D CNN extension across multiple tasks, including pose, motion, and force estimation from OCT volumes. Across these applications, we find that our 3D CNN extensions are effective for regression problems. Also, despite the natural increase in terms of the number of trainable parameters, the final architectures are efficient as they do not contain more parameters than typical 2D CNN models. Also, inference times are generally low enough for real-time applications. Thus, we find the concept of designing specific architectures for 3D domain to be effective, as very efficient solutions can be built. This also matches observations for medical segmentation problems where 3D deep learning is enabled by efficiently handcrafted architectures [231]. Similar to our results

in Sections 6.5 and 6.8, reported results in the literature show that 3D CNNs consistently outperform 2D CNNs across applications, including brain lesion segmentation [231] and brain tumor segmentation [451].

However, extensive handcrafting of architectures has also been criticized as it is highly subjective and time-consuming [602]. In particular, architecture concepts such as Inception are heavily engineered concepts. While having shown excellent performance for a variety of problems [129], newer concepts such as ResNeXt have demonstrated that much more simple designs are also useful. This raises the question of whether architecture design can also be simplified for building 3D CNN models.

Another way of moving from 2D CNNs to 3D is to attempt a straight-forward *adaptation*. Here, we propose the idea to directly reuse 2D architectures by extending each of their internal operations from 2D to 3D. Thus, all convolutional layers are isotropically extended by a third dimension. A key advantage of this method is that no additional architecture engineering is required, as successful 2D models can be directly used in the 3D domain. A substantial downside is the immense increase in terms of the number of trainable parameters and memory requirements. Only with the recent advancement of deep learning hardware, this concept was enabled in terms of the memory requirements. Yet, the problem of parameters remains as there is a severe risk of overfitting due to overparameterization.

One way to overcome this problem is to employ transfer learning in the 3D domain. By reusing pretrained weights, the risk of overfitting can be reduced, and training can be sped up. Therefore, we propose a specialized initialization strategy where 2D weights are copied into the third dimension and rescaled to maintain consistent feature scales, see Section 4.1.3. We find this concept of multi-dimensional transfer learning to be very effective for the LVQ problem, see Section 6.4. Also, we adopt this concept for adapting 2D CNNs for 2.5D data in the context of IVOCT-based plaque detection, see Section 6.3. For both problems, we find that the initializing strategy is particularly important for achieving high performance.

Some approaches in the literature have also tried to reuse existing 2D CNNs for 3D problems. Diba et al. [106] used a 2D CNN as a teacher model for another 3D CNN, achieving a performance inrease by $3.3\,\%$. While their 3D CNN captures 3D context, weights are not directly reused, and the 3D CNN still needs to be trained from scratch. We find that weight reuse and our weight scaling strategy improve performance by $\approx 28\,\%$ and $\approx 6.3\,\%$ for LVQ in Section 6.4 and IVOCT classification in Section 6.3, respectively. Another approach reused pretrained 2D CNN weights in a 3D CNN [441]. However, the method did not perform any weight extension to the third dimension. Thus, contrary to our approach, the CNN still performed 2D processing on a 3D input without considering context from the additional dimension. As a result, we propose weight reuse and coverage of full 3D context with 3D CNNs, which combines and extends previous approaches that tried to reuse 2D CNNs for 3D problems.

Another way of taking away the need for architecture engineering is the concept of NAS [602]. Here, another layer of optimization is introduced where a subset of architecture hyperparameters $h^A_{ML} \subset h_{ML}$ is optimized in additional to model weight optimization. From a historical perspective, *learning* architectures appears to be the next reasonable step, given that the last major change in deep learning pipeline design has been the move from feature engineering to feature learning. With NAS, architecture

engineering is replaced by architecture learning, and the engineering task is shifted to designing search spaces.

A significant downside of NAS are the immense computational requirements. In its vanilla state, each iteration of architecture optimization requires complete training until convergence. Also, most architecture hyperparameters are not differentiable with respect to a loss function, which aggravates the optimization problem. This leads to computational resource requirements that encompass several thousand GPU hours for early NAS approaches [603]. Given that a major challenge of 3D CNN design is the increase in computational requirements, NAS does not appear to be appealing for this problem.

However, if we consider another multi-dimensional problem that is moving from 1D CNNs to 2D CNNs and vice-versa, the situation is very different. 1D deep learning comes with the great advantage of being computationally cheap. Thus, low computational requirements open up the possibility of employing NAS. We leverage this idea to propose NAS for architecture transfer between dimensions, see Section 4.1.1. We hypothesize that learning an architecture on lower-dimensional data also transfers well to higher-dimensional. In this way, we make use of NAS' great potential while keeping computational effort bounded.

We propose this NAS idea for medical image segmentation tasks by learning the microstructure of a U-Net-like CNN architecture. We study the problem for transfer between 1D and 2D images, where we find that architectures learned on 1D data also perform well on 2D data while needing only a fraction of the search time required for searching on 2D data, see Section 6.2. Thus, we propose a very efficient approach for learning architectures that can be employed for higher-dimensional data. In particular, the idea could also be leveraged for finding 3D CNN architectures, for example, by learning from 2D or even 1D data. For the medical domain, this is an auspicious approach as there are hundreds to thousands of deep learning problems that require architecture engineering. Automatizing this process with NAS could provide a great benefit, in particular, for niche problems that get little attention otherwise.

In the literature, there are only a few applications of NAS in the medical domain so far. Weng et al. [536] investigated learning the cells of U-Net architecture when searching on a 2D natural image dataset. In contrast to our work, the authors did not focus on search times or search efficiency. Other approaches have directly applied NAS to the same task and 2D data representations for both the search task and final applications [130, 523]. Thus, we provide a new way of improving NAS search times by moving between data dimensions and achieving a search time reduction by $87.5\,\%$, see Section 6.2.

So far, almost all applications focus on the micro search space for finding cells. In the natural image domain, Auto-DeepLab [302] has demonstrated that learning the macro-level architecture can also be very beneficial. Adopting this idea for U-Net-like architectures might be an important next step for architecture design in the medical image domain.

7.1.2 Temporal Data: Recurrent-Convolutional Models

Another class of architectures that is well-suited for multi-dimensional image data processing is a combination of convolutional and recurrent models. For this type of

architectures, a typical application is a spatio-temporal data problem with one to three spatial data dimensions and a temporal data dimension. Convolutions typically process the spatial data dimensions, and a recurrent architecture processes the temporal dimension. Convolutions are also common for temporal processing; nevertheless, recurrent processing for temporal data is still very popular [391].

The most straightforward approach is to use a CNN that processes each temporal instance within the spatio-temporal tensor independently. This results in a vector of feature representations that are then processed by a recurrent model. Finally, all states are aggregated or one state, usually the last one, is selected and fed into a fully-connected output layer for classification or regression. This concept has been widely used both in the natural [497] and in the medical imaging domain [228]. We also adopt this concept for volume-based force estimation in Section 6.8, needle-based force estimation in Section 6.1, and motion estimation in Section 6.6.

A common way to improve convolutional and recurrent models has been to use a better feature extractor for the CNN part or a more advanced recurrent architecture. Improved CNNs have included popular architectures such as ResNet, Inception, ResNeXt, DenseNet, or, very recently, EfficientNet [482]. The architecture changes are generally focused on more efficient optimization, computations, and feature reuse without a focus on multi-dimensional aspects. Similarly, for recurrent architectures, a lot of variants have been proposed where the most popular ones include LSTMs, GRUs, and attention RNNs [524]. These methods are generally focused on capturing temporal dependencies in better ways.

The introduction of convolutional LSTMs [556], however, represents an interesting multi-dimensional intersection of spatial and temporal processing. Here, the convolutional gates enable spatial processing of temporal sequences. When placing this module after a CNN-based feature extractor in a model, spatial features do not need to be pooled. Thus, the recurrent cells are still able to process spatially resolved information. We find that this method is effective for our spatio-temporal regression problems, generally outperforming the use of a GRU that does not perform spatial processing, see Sections 6.1, 6.6, and 6.8. While these types of models are rarely employed in the medical image domain, similar improvements have been observed in the natural image domain, for example, for gesture recognition [596] and object detection [458].

We propose cGRU-CNN in Section 4.3, an extension of this method, which is conceptually simple but very impactful. We move the convolutional recurrent module to the front of the model. In this way, we perform temporal processing first while preserving the spatial structure of the input. The recurrent model outputs a single spatial state that is then processed by the CNN. Intuitively, this approach puts a larger focus on spatial processing. The initial, recurrent module provides an aggregated spatial representation that is then extensively processed by the CNN.

This can be particularly useful if there is redundancy in the temporal dimension, and the convolutional recurrent module can first learn a compact representation. The idea of potential redundancy in the temporal dimension particularly applies to needle- and volume-based force estimation, where the temporal dimension largely serves the purpose of providing more consistency. Across our force estimation applications, we find that this novel method performs very well, outperforming other spatio-temporal processing methods, see Sections 6.1 and 6.8.

Our use of temporal information for force estimation is very different from tasks such as surgical action recognition [11], where temporal information is crucial for the task. This is reflected in our results in other tasks, such as motion estimation in Section 6.6. Here, the relation between temporal frames is important to predict a relative motion vector between frames. Thus, while our new method still performs well, it is on a similar level as spatio-temporal models that perform spatial processing first.

The idea of convolutional recurrent modules is beneficial in a lot of cases where temporal information needs to be aggregated for extensive spatial processing. One example are spatio-temporal segmentation problems such as lesion activity segmentation. The segmentation task requires the model to be suitable for dense image segmentation, for example, in the form of a U-Net-like model.

For processing a spatio-temporal sequence for a segmentation task, we propose an encoder-cGRU-decoder (cGRU-CNN-U) model that takes care of temporal aggregation in between the encoder and the decoder, see Section 4.3. Thus, in terms of positioning of the temporal unit, this model represents a trade-off between prioritization of spatial and temporal processing. A part of spatial processing is performed before taking any temporal context into account. Then, these more abstract representations are aggregated and processed again by convolutional layers for obtaining the final segmentation map.

In the literature, the fusion of recurrent and convolutional models for segmentation tasks is rare. One approach does not use recurrent processing for a temporal dimension but instead proposes to use recurrent processing to capture inter-slice context for a segmentation task [14]. Another method utilized recurrent processing for iterative refinement of images for artifact removal [442]. Thus, recurrent segmentation architectures for temporal processing are still rare. For temporal problems, simple concatenation of features has been used instead of temporal aggregation [51]. Here, features are stacked after processing by the encoder and then reweighted and added in the first layer of the decoder. In contrast to our method, the approach does not utilize gating for information aggregation. In particular, we find that using recurrent gating outperforms feature concatenation that was used by Birenbaum et al. [51] for multi-time point MS segmentation, see Section 6.7.

Furthermore, we extend the approach of fusion by concatenation by studying architecture design with other aggregation methods between encoder and decoder. For two temporal states, the concept is particularly promising as complex temporal aggregation with gating mechanisms might not be required. These architectures are referred to as Siamese models with two processing paths before aggregation, see Section 4.2. We consider different ways of aggregation by pixel-wise addition or subtraction, as well as feature concatenation. In general, we find that these Siamese models work well both for lesion activity segmentation in Section 6.7 and also motion estimation in Section 6.6.

Regarding related work, Siamese models are rare in the medical image domain. Except for the work by Birenbaum et al. on brain lesion segmentation [51], the concept has mostly been used to aggregate different spatial resolutions instead of temporal or other types of states [231]. Thus, we present new Siamese models and new potential applications of Siamese models that have not been considered in the literature so far.

We further augment Siamese models by proposing attention-guided interaction modules for the case of two states to be aggregated. We find these modules to be useful for lesion activity segmentation, see Section 6.7. Providing additional context through

attention for spatio-temporal problems has recently gained traction for a lot of problems in the natural image domain [459]. In the medical image domain, attention for spatio-temporal processing is still rare. However, for purely spatial problems, attention has been successful for different tasks where a model is guided to focus on particular regions through attention mechanisms [364, 417]. In our work, we also derive spatially resolved attention; however, it is derived from temporal context and thus multi-dimensional in nature. This new type of model could potentially find application in other spatio-temporal medical problems where disease changes over time need to be localized.

7.1.3 4D Deep Learning

Learning from 4D data is a problem that has hardly been addressed so far, both in the natural and in the medical imaging domain. For the natural image domain, the main reason is the rarity of 4D data. One of the few potential applications is the processing of spatio-temporal depth data. Time-of-flight depth cameras capture 3D context that can be encoded in 3D volumes using occupancy grids [327]. A stream of these volumes would also represent 4D data; however, most approaches have opted for a lower-dimensional encoding such as depth maps for processing streams of depth camera images [124]. Only very recently, 4D deep learning methods have emerged in the natural image domain [88].

Typically, novel deep learning methods have appeared in the natural image domain and were then transferred to the medical image domain. Therefore, the lack of 4D deep learning approaches in the natural image domain can also be seen as a cause for the lack of 4D deep learning methods in the medical image domain. Yet, 4D deep learning is much more interesting for the medical image domain as there are lots of 3D imaging methods that could provide streams of 4D spatio-temporal data. In particular, throughout the last few years, imaging systems for OCT and MRI have become faster in terms of acquisition speed, providing both high spatial and temporal resolution [505, 537].

As a result, we identify a strong need for 4D deep learning research as there are great opportunities for exploiting high-dimensional context in medical image data. This leads to one of the key challenges we address in this thesis, which is 4D deep learning model design. In general, 4D model design is complicated as the curse of dimensionality comes into full effect, leading to a significant increase in terms of the number of trainable parameters and computational requirements. Given that 3D deep learning model design has already been considered very challenging [327], 4D model design is going to require a lot of research effort.

Previously, we discussed three primary techniques for CNN model creation: design, adaption, and learning. Given the immense computational requirements of 4D deep learning, the only viable choice, as of today, is handcrafted model design. In this thesis, we address 4D deep learning model design both for regression and segmentation tasks. For regression problems, we design 4D CNNs that process 4D spatio-temporal data in different ways, see Section 4.1.4. We consider full 4D convolutions that treat the spatial and temporal dimension equally, as well as a factorized and a multi-path version that process the spatio-temporal data in a more efficient way.

For the factorized model, we decompose spatial and temporal convolutions, which leads to fewer model parameters to partially counter the increase in parameters for 4D models. The downside of this approach is that the CNN is now only able to learn

separable kernels which might hinder its representational capabilities. Indeed we find that performance is slightly reduced with factorized convolutions instead of full convolutions across different data representations, see Section 6.8, which suggests that there is a trade-off between model capacity and performance.

We also design a multi-path CNN that first processes each spatial volume in the spatio-temporal tensor independently with a 3D CNN. Here, a higher level of efficiency is achieved by avoiding 4D processing at first. Then, after learning spatial representations, spatio-temporal processing is performed with a more light-weight 4D CNN. We find this approach to be particularly useful for motion estimation, see Section 6.6. Implemented with a DenseNet architecture core, the model outperforms a full 4D CNN while also being more efficient in terms of the number of parameters.

The idea of performing more efficient 3D CNN processing first can also be exploited with convolutional-recurrent models. After the initial processing stage, the spatial features can be pooled and fed into a standard recurrent model. This results in a model such as RN3D-GRU, which we employ for force estimation, see Section 6.8. This model performs completely separate spatial and temporal processing, and we find it to be far less effective than our 4D CNN variants. However, when replacing the standard recurrent module with a convolutional GRU that preserves spatial structure while processing, we can slightly improve performance. We observe improved performance both for force and motion estimation.

Another 4D approach that we study is a 4D extension of our proposed cGRU-CNN model. The adaptation is straightforward as all convolutional layers can be extended to 3D for volume processing. We explore our model both with a ResNet- and a DenseNet-based implementation, and we find it to be particularly effective for force estimation, see Section 6.8. The model is both more efficient than all other 4D models and also performs best across all models. However, for motion estimation, we find that the model only achieves mediocre performance and is slightly outperformed by its counterpart CNN-cGRU. This suggests that the model's effectiveness is application-specific, and one particular 4D model might not be optimal for all applications. As discussed in the previous section, we also propose an adaptation of the cGRU-CNN for 4D segmentation problems, where the concept outperforms previous approaches, see Section 6.7. Thus, overall, cGRU-CNN appears to be a viable alternative to other deep learning methods for different applications and different data dimensionality.

Considering approaches in the literature, there have been hardly any 4D deep learning models. One of the few examples in the natural image domain is a 4D CNN that is designed for sparse volumes derived from depth camera images [88]. The 4D CNN architecture is only evaluated on an artificial dataset. In contrast, we provide extensive results for multiple real-world datasets and applications, including motion estimation in Section 6.6, force estimation in Section 6.8, and MS lesion segmentation in Section 6.7. Another approach employed a 4D CNN for video data that was transformed into a 4D representation [588]. Here, the authors consider two temporal dimensions with the usual short-term dimension and an additional long-term dimension. Thus, it is demonstrated that applying higher-dimensional processing can be advantageous, even if it is not strictly necessary. This follows our proposed idea of striving towards the use of higher-dimensional data if it is available.

In the medical domain, a lot of approaches have processed 4D data [275, 276, 284,

379, 436]; however, most applications avoid full 4D deep learning models and use a lower-dimensional representation instead. Still, two approaches for 4D CT images employed 4D CNNs for processing [92, 210]. However, in both cases, the authors were unable to demonstrate an advantage of 4D processing over 3D processing. In contrast, we demonstrated performance improvements with 4D deep learning for multiple applications. This suggests that 4D deep learning is still within its early stages and requires more extensive research. More and more applications are showing up, which indicates that 4D deep learning is going to be an active research field in the next few years.

7.2 Data Representations and Applications

Multi-dimensional deep learning models are closely connected to the data representation that is employed. The standard data representation in the natural image domain is a 2D spatial color image, sometimes augmented by a temporal dimension or the addition of a depth map. In the medical image domain, data representations are very diverse, and in many cases, individual imaging modalities come with different 2D, 3D, and 4D data representations. Still, due to the primary purpose of visualization for medical experts, most medical images are presented as a 2D data representation, for example, as a slice or a projection.

The fact that the natural image domain mostly uses 2D representations and the common approach of 2D visualization have lead to a large presence of 2D data representations despite the 3D volumetric or 4D spatio-temporal nature of most imaging modalities in the medical image domain. Another important factor are computational and memory requirements, which have made lower-dimensional data representations more attractive for deep learning. However, the steady advancement of deep learning hardware has enabled the use of higher-dimensional data representations.

The success of early deep learning approaches with 3D volumetric data motivates the question of which data representation to use for deep learning. While some studies indicated that volumetric processing is preferable, a majority of deep learning applications still rely on 2D representations. Also, processing in a higher-dimensional space is computationally expensive and needs to be traded off for a potential performance gain. This motivates an extensive investigation of different data representations for medical deep learning problems. In this thesis, we address multi-dimensional data for OCT and MRI, which both offer many different data representations and applications.

7.2.1 Optical Coherence Tomography

In its most fundamental implementation, OCT devices provide 1D depth images. Due to its fiber-optical imaging principle, 1D OCT imaging systems can be implemented using a single optical fiber. This enables a variety of applications for minimally invasive medical interventions.

One example is the possibility of embedding an OCT fiber in a needle. A lot of interventions such as brachytherapy or biopsy require needle insertions and accurate placement. Here, an OCT fiber imaging the tissue in front of the needle tip could

potentially provide valuable information for guiding the insertion process. For example, estimating the force acting on the needle tip can be used to generate haptic feedback or provide information on the tissue type that the needle is currently placed in.

We propose a needle design for fiber-based force estimation using OCT, see Section 6.1. While there are alternative fiber-optic designs for that task, for example, using Fabry-Perot interferometry [466] or fiber Bragg gratings [265], our approach comes with the advantage of being easy to manufacture and being flexible in design with the option to adapt for different force ranges and applications. Also, a slight modification of our design could enable immediate imaging of the tissue in front of the needle [368]. Simultaneous estimation of tissue properties and forces acting on the tissue might help in improving needle placement accuracy, which is currently very error-prone [237].

An aspect that differentiates our needle design from other fiber-optic approaches is the actual imaging capability of OCT that results in 1D images and not just scalar estimates. Thus, a complex mapping from images to force is required, which is the task of vision-based force estimation [179]. The 1D spatial OCT data representation is well-suited for processing with 1D CNNs. However, due to OCT systems' high temporal resolution, using a 2D spatio-temporal data representation is also well-suited for the problem. This is motivated by the idea that a temporal history might provide a more consistent estimate than a single depth scan. Indeed we find that using spatio-temporal data representations generally performs better and that a higher-dimensional data representation seems to be preferable as we improve performance by $54\,\%$, see Section 6.1. Previous approaches for OCT fiber-based deep learning have not used temporal context so far [367, 368]. When considering the extended and potentially more difficult task of simultaneous tissue property and force estimation, the additional spatio-temporal context might be even more beneficial.

Another medical intervention where 1D OCT imaging can be used is intravascular OCT. Here, a single optical fiber scans the arterial walls of a patient, which can be used to obtain cross-sectional images. Practitioners use these cross-sectional images to detect plaque deposits and make a treatment decision or to assess a stent that has been implanted into the artery. As 1D depth scans alone cannot provide sufficient context and morphological structure for interpretation, scanning techniques are used to create higher-dimensional images. IVOCT scanning leads to a spatio-temporal 2D M-Scan that can potentially be reconstructed to spatial 2D or 3D data representations.

In theory, 1D depth scans could be processed individually, for example, by exploiting tissue scattering properties for identifying plaque deposits. This strategy was used for early feature engineering approaches where attenuation coefficients or scattering properties were analyzed in individual depth scans in order to differentiate tissue types [557]. However, deep learning approaches have found that using just a 1D spatial data representation with a 1D CNN is not very effective [255]. Instead, using a 2D polar data representation worked well in multiple applications [3, 365, 560].

While it is reasonable that 2D polar images provide more spatial context, the image does not accurately reflect the morphological structure of the artery tissue. Therefore, medical practitioners usually rely on Cartesian data representations for image interpretation. In our work, we investigate this mismatch by considering both polar and Cartesian data representations for deep learning-based plaque detection.

In general, we find that using Cartesian images is advantageous over using polar data

representations, see Section 6.3, which is particularly interesting as previous work rarely considered Cartesian images [3]. Also, we find that this advantage is mostly tied to data augmentation techniques as we observe a substantial increase in performance for Cartesian images in combination with data augmentation. Another interesting insight is that both data representations can be fused in a single model, which leads to slightly improved performance. In a more general context, these insights indicate that data representations are also important in the same level of dimensionality, not just in terms of an increase or decrease.

Other approaches have primarily relied on polar representations [3], with few exceptions where Cartesian images were used [22]. What has been missing so far is the use of full 3D volume data representations. This might be tied to the difficulty of reconstructing an accurate 3D IVOCT volume [273]. Precise reconstruction requires knowledge of the exact path of the imaging catheter within the artery. As this path is often unknown, only approximations can be reconstructed, which might be an inaccurate representation of the actual patient's morphology. Thus, deep learning methods cannot exploit the 3D context effectively. Additional tracking of the catheter with another imaging modality could be an effective approach for addressing this problem [271].

A different type of data representations can be obtained with mirror-based OCT scanning devices. Here, a second and third data dimension is obtained by redirecting the OCT's laser beam laterally. As these scanning setups require more space, this acquisition mechanism is usually employed for external imaging, for example, in ophthalmology.

A first straight-forward comparison of data representations is, therefore, the use of individual 1D depth scans versus the use of a 2D cross-sectional images. Similar to IVOCT applications, classic feature engineering approaches relied on scattering features derived from individual depth scans [494]. In our work on retina layer segmentation, see Section 6.2, we find that using more spatial context with 2D scans is advantageous over processing individual 1D depth scans. This confirms findings in the literature, where 2D B-Scans are often employed for retina layer segmentation [195, 416, 545].

Additional spatial context can be considered with another lateral scanning direction, which results in full 3D OCT volumes. Although these images are 3D in nature, they often do not contain full 3D information. When imaging a surface of an object or tissue, the infrared lead can only penetrate the material by $1\,\mathrm{mm}$ to $2\,\mathrm{mm}$. Thus, in many cases, OCT volumes largely capture a surface. This surface can be encoded more efficiently as a depth map using a maximum intensity projection. Thus, if discarding subsurface image information is deemed acceptable, the 3D volume can be encoded as a 2D data representation.

For many applications in the field of computer-assisted interventions, this lower-dimensional representation is interesting as processing times can be reduced when using 2D instead of 3D data. 2D projections have therefore been employed frequently, both for OCT [276] and other modalities such as MRI [50]. We study this approach across the two applications pose estimation in Section 6.5 and force estimation in Section 6.8. While deep learning architectures using 2D depth maps or projections are more efficient, models using full 3D volumes significantly outperform their counterparts.

We find that this insight is consistent across different computer-assisted intervention application scenarios. For pose estimation, both intensity projections and depth maps lead to higher estimation errors compared to using 3D volumes. Here, subsurface features

appear to help in obtaining more accurate estimates. For force estimation, where we use depth maps that encode the tissue surface being deformed, similar observations can be made. Across different architectures, the use of 3D volumes significantly outperforms the use of depth maps. So far, using full OCT volumes for computer-assisted intervention applications has generally been avoided with the most cited reason being increased or even infeasible computational effort [275, 276]. Our research suggests that using full 3D OCT volumes is generally preferable over lower-dimensional representations. Still, the increase in performance needs to be traded off for longer processing times, which needs to be taken into account for different applications.

The high performance of models using full 3D volumes raises the question of whether the advantage stems from the additional subsurface information or from the higher-dimensional nature of the data itself. We explore this aspect by encoding the 2D depth maps in a 3D volume, see Section 6.8. While we find that using full 3D volumes performs significantly better, encoding 2D depth maps in a higher-dimensional space improves performance over using a 2D data representation.

Intuitively, this relates to the well-known kernel trick where the same fundamental principle is exploited. Features are encoded in a higher-dimensional space in order to achieve better separability between examples. Also, in the natural image domain, there have been similar approaches where point clouds have been encoded as volumes for deep learning applications [327]. Overall, this indicates that artificially increasing the dimensionality of a data representation could lead to improved performance. While the trade-off between performance and inference time remains, the availability of more computation resources could make this method useful for applications where high-performing models are needed.

Similar to needle-based force estimation, OCT devices' high acquisition speed allows for using temporal context, which increases the data dimensionality in exchange for potentially more consistent estimates. We study this aspect both for 2D depth maps and 3D volumes, leading to 3D and 4D spatio-temporal data representations. For both data representations, we find that using a temporal history leads to improved performance. In particular, we find that the extension to 4D data representations is effective across different regression tasks including force estimation in Section 6.8 and motion estimation in Section 6.6.

Using 4D spatio-temporal data also opens the possibility of forecasting. For computer-assisted interventions, this is particularly interesting. Forecasting could allow for compensation of lag or delay in a physical system. Also, safety features could be implemented that preemptively act on dangerous motion patterns, surgical tool positions, or force values that have been projected by a deep learning model. In the literature, temporal information as been consider for tasks such as force estimation [29, 321], but forecasting has not been studied so far. For both motion estimation in Section 6.6 and force estimation in Section 6.8, we find that short-term forecasting works will with only minor performance reduction compared to estimation. In particular, we observe increased importance of a longer temporal history for this problem.

So far, a temporal extension for more consistent estimates has hardly been used and for OCT, and there are no other deep learning applications using full 4D OCT data. Given our successful use of 4D OCT across several problems, we hypothesize that other OCT applications might benefit from higher-dimensional data in the future.

7.2.2 Magnetic Resonance Imaging

2D cross-sectional slices are the most common MRI data representation in practice [301], although MR images are usually acquired as a volume. There are multiple reasons why 2D MRI data representations have been popular. First, cross-sectional slices are the data representation that is typically used for visualization and examination by practitioners. Given that experts derive most of the relevant information from slices, deep learning models should be able to mimic that behavior. Second, deep learning models, in particular CNNs, are largely adopted from the natural image domain where 2D data representations are most common. This allows for easy adaptation of existing methods for MRI learning problems. Third, 2D data representations are a lot easier to handle than full 3D MRI volumes due to computational cost and difficult model design.

Recently, deep learning models using full 3D volumes have gained more traction [451]. The switch towards 3D data representations was largely enabled by efficient architecture design and usually low spatial resolution for saving computational cost [251]. Across multiple different applications and datasets, it was demonstrated that using 3D volumes is advantageous over using 2D data representations [84,231,258,374,405]. This suggests that exploiting inter-slice context is important for extracting relevant information for learning problems. Considering the workflow of clinicians, where 2D slices are usually used for interpretation, some inter-slice context is likely used as well since multiple slices can be assessed consecutively.

As a result, for MRI-based deep learning problems, spatial 3D data representations are becoming more popular. However, there are also learning problems where temporal information is relevant, as well. Often, capturing short-term temporal context is very difficult with MRI due to long acquisition times. For example, for cardiac imaging, acquisition needs to be performed while the patients hold their breath, or additional tracking is required to compensate breathing motion during imaging [322]. Thus, 3D spatio-temporal data representations with 2D slices are typical for cardiac imaging [563].

We find that exploiting temporal context with 3D spatio-temporal MRI data is useful for the task of left ventricle quantification, see Section 6.4. Similar observations have been made in the literature with different types of 3D spatio-temporal deep learning models [222,561]. Although geometric parameters of the heart should be quantifiable from an individual cross-sectional image, the additional temporal context appears to help in obtaining more consistent estimates.

More recently, full 4D MR imaging has been introduced and studied, for example, in the context of cardiac imaging [322]. So far, there are no deep learning approaches using full 4D spatio-temporal data for this problem. However, 4D spatio-temporal data representations have become more popular in the context of functional MR imaging [43,293,320]. A few deep learning approaches have made use of 4D data, which includes a recent approach employing the cGRU-CNN approach proposed in this thesis [43]. This suggests that 4D spatio-temporal data representations are also gaining traction for MRI applications. The successful use of 4D data for fMRI applications indicates that other areas, such as cardiac imaging might also benefit from processing the higher-dimensional data representation.

Temporal data can also be considered from a longitudinal perspective. Here, MRI's long acquisition time is not a bottleneck. Several applications require the consideration

of longitudinal data, for example, for tracking disease progression. In its straightforward form, a longitudinal comparison is made between two consecutive scans, where the goal is to find differences between the scanning points. For MRI volumes, this results in 3.5D data.

In our work, we demonstrate that this type of data can be processed effectively with deep learning methods, see Section 6.7. The task of MRI-based lesion activity segmentation has been mainly addressed using classical machine learning methods. The approaches usually employ lower-dimensional representations or features extracted from small regions [422, 429, 431]. We show that using the entire 3.5D image data improves performance over these classic methods, see Section 6.7. Also, we demonstrate that our Siamese approaches are well suited for processing this type of 3.5D data. In particular, our attention mechanism effectively transfers information between the two time points. In a more general context, other tasks that require tracking of disease progression, for example, Alzheimer's disease [218], or muscular dystrophy [138] could also benefit from our Siamese deep learning approach.

In terms of data representation, the next step is to make use of full 4D spatio-temporal longitudinal data. Given that diseases such as MS are tracked over a long period of time, several scans are accumulated for each patient over time. These additional scans could potentially provide more context and allow for more consistent estimates.

We study this concept for lesion activity segmentation. For this particular problem, a history of additional volumes can be seen as a more consistent baseline. Potential changes between the original baseline and follow-up scan might be more obvious when also considering scans that lie further in the past. We observe a moderate increase in performance, which suggests that the additional temporal information is helpful.

This matches observations for the closely related task of lesion segmentation from individual scans [50]. Here, Birenbaum et al. used 3D spatio-temporal longitudinal data with 2D spatial representations. Although change over time is not as important for individual scan segmentation, the authors also observed an increase in performance with an additional history.

For other longitudinal MRI-based problems such as Alzheimer's disease [218], there have been no approaches using full 4D longitudinal data. A major reason for this is the lack of 4D longitudinal datasets [74]. In terms of methodology, our 4D deep learning methods would be well suited for tracking disease progression in other applications.

8 Conclusions

In this thesis, we addressed deep learning with multi-dimensional medical image data. We adapted and proposed a plethora of deep learning methods for medical image analysis problems. We provided an extensive empirical evaluation of our methods and we studied the use of 1D to 4D OCT and MRI data representations in different medical applications. In this chapter, we summarize the primary contributions of our work. We discuss opportunities for future work on multi-dimensional deep learning and conclude with final thoughts.

8.1 Main Contributions

Throughout this thesis, we addressed two very fundamental research questions that are part of almost every medical deep learning problem but rarely put into focus. The first research question is concerned with the data representation used for deep learning methods. Medical imaging systems often provide multi-dimension 3D or 4D data; however, the full images are usually sliced up or projected into lower-dimensional representations. This raises the question of which type of data representation is suitable for which kind of task. We addressed this research question for multiple problems, including vision-based force estimation, pose and motion estimation, tissue characterization, and tissue segmentation using the multi-dimensional imaging modalities OCT and MRI.

The second research question is concerned with the choice of deep learning models for different representations. In the medical image analysis domain, a popular approach is to directly adopt 2D CNNs for RGB images and use them, for example, with 2D cross-sectional slices. When moving to higher-dimensional 3D and 4D image data, this method is difficult to apply. Therefore, we studied different ways of designing, adapting, and even learning CNN architectures. We also put a focus on spatio-temporal deep learning models, where the different data dimensions are treated differently with convolutions, recurrent gating mechanisms, or attention-based methods. We proposed multiple new deep learning methods that can be employed in a variety of applications and numerous imaging modalities.

8.1.1 Multi-Dimensional Deep Learning Methods

CNNs are intuitively well-suited for multi-dimensional deep learning. Their core processing unit, the convolutional operator, can be adapted to different dimensionalities. Therefore, we first adapted CNNs to different scenarios with different data representations, ranging from 1D to 4D CNNs. Starting with 1D CNNs, we studied the idea of neural architecture search due to the computationally cheap nature of lower-dimensional data. In particular, we proposed transfer of architectures found by NAS between data

dimensions, see Section 4.1.1. As a next step, we moved from 2D to 3D CNNs by handcrafting efficient architectures. High efficiency is required due to the higher-dimensional nature of 3D CNNs, and the real-time requirements associated with tasks in the field of image-guided interventions. We found that our 3D CNN designs proposed in Section 4.1.2, while being time-consuming to engineer, performed well and fulfilled real-time requirements. As an alternative to handcrafting architectures, we considered the direct transfer of existing 2D CNN models to 3D. This concept is very resource-intensive and comes with the risk of overfitting. We overcame this problem by proposing a new multi-dimensional transfer learning strategy, where we transferred pretrained 2D convolutional kernels into 3D CNNs with a weight scaling strategy, see Section 4.1.3. We found this method to be effective, given that resource requirements can be met. Finally, we explored 4D CNNs, which are even more challenging due to their high-dimensional nature. Therefore, we introduced different 4D CNNs in Section 4.1.4, where we employed different concepts for the efficient processing of a 4D tensor's different data dimensions. Here, we found that decomposed spatial and temporal processing leads to more efficient architectures while performing similar to or even better than plain 4D CNNs.

Many medical image analysis problems require the processing of two different states, for example, when studying changes between a patient's visit. For this type of problem, we proposed Siamese CNNs in Section 4.2, adapted from the natural image domain. Our Siamese CNNs use two processing paths, sharing parameters, followed by fusion and a joint processing path. Throughout several medical applications, we found that the fusion point and the method of fusion are key hyperparameters to consider. Also, we introduced a novel attention-guided interaction concept that allows for information exchange between the two paths before fusion. We found that this method improves the detection of disease progression and also provides interpretable attention maps.

Some two-state problems can be extended to a full spatio-temporal problem by considering more spatial images in a sequence. Besides CNNs, convolutional-recurrent deep learning models can be used for processing spatio-temporal data. Here, we adapted and proposed several different architectures that we employed for applications with different data dimensionality, see Section 4.3. First, we adapted CNN-RNN models, inspired by video processing methods in the natural image domain. Then, we proposed a modified convolutional RNN-CNN model that is particularly well-suited for spatio-temporal problems where temporal information is considered to achieve more consistency. We found this concept to outperform other methods for several different applications. We also proposed a modification of this architecture for segmentation problems, allowing for the processing of 4D spatio-temporal data for image volume segmentation.

8.1.2 Multi-Dimensional Deep Learning Applications

We applied the numerous deep learning architectures we adapted and proposed to a variety of biomedical image processing problems that come with different data representations. We started with lower-dimensional data representations and continued with increasing dimensionality. OCT's lowest dimensional data consists of 1D depth scans, acquired by a single optical fiber. While these depth scans cannot capture a lot of spatial context, they are well-suited for computer-assisted interventions where we integrated

an optical fiber into a needle for tissue insertion scenarios. Here, we learned needle tip forces from 1D depth scans and 2D spatio-temporal data, obtained from sequences of 1D scans. In general, we found that using additional temporal information is helpful and outperforms the use of 1D data consistently across different models. The higher-dimensional data is still processed in real-time, and we demonstrated our approaches' applicability in a prostate needle insertion scenario, see Section 6.1.

Another application where 1D OCT data can be used is retina layer segmentation. Here, external scanning devices can be used that also enable 2D and 3D data acquisition for additional spatial context. For this problem, we also found that higher-dimensional data processing leads to improved performance, see Section 6.2. When studying neural architecture search for this problem, we found that architectures searched on 1D data also work well when transferred to 2D data. This suggests that architectures transfer well across dimensions, even when the data dimensions are very different, as it is the case for 1D and 2D OCT data.

We also explored IVOCT-based plaque detection in coronary arteries, which is a problem with 2D data representations, see Section 6.3. We compared both 2D Cartesian and 2D polar data representations for the problem, where we demonstrated that Cartesian images appear to lead to a higher performance. We found that this is particularly linked to data representation-specific data augmentation strategies. Furthermore, we showed that combining both data representations with a late-fusion Siamese CNN can improve performance further.

Next, we considered LV quantification with 2D MRI slices and 3D spatio-temporal sequences, see Section 6.4. Deriving LVQ parameters works well from 2D slices but can be improved by also utilizing full 3D information. Also, we observed a more substantial improvement by using more advanced CNN architectures, indicating that both temporal information and more advanced architectures are valuable for this problem.

Furthermore, we proposed pose estimation with 3D OCT data, see Section 6.5, where 2D data representations can also be used by constructing lower-dimensional projections of the OCT volumes. Here, we made use of OCT's light-scattering properties and use maximum intensity projections to extract surfaces from the volumes, encoded as depth maps. While these are efficient representations that can be used for OCT-based pose estimation, we found that using 3D volumes performs significantly better. Furthermore, we demonstrated that 3D CNNs exploit subsurface features in 3D volumes effectively, indicating that higher-dimensional context is also useful for purely spatial dimensions.

We further considered full 4D spatio-temporal data representations for several deep learning problems. For OCT-based motion estimation in Section 6.6, OCT-based force estimation in Section 6.8, and MRI-based lesion activity segmentation in Section 6.7, we found that full 4D spatio-temporal data leads to the best performance. Also, temporal information leads to more consistent estimates, both for 3D and 4D spatio-temporal data. In addition, we demonstrated that there is an inherent advantage in using higher-dimensional data representations by encoding lower-dimensional data representations in a higher-dimensional space.

8.2 Future Work

Our two research questions are very broad in nature, and they can be asked for almost any medical image analysis problem. Therefore, there are still many problems to be addressed in future work.

8.2.1 Deep Learning Model Design

We considered several different ways of designing architectures for a multi-dimensional context in our work. This includes handcrafting efficient architectures, directly applying existing architectures in a higher-dimensional setting, and also learning architectures automatically through NAS.

The latter is a particularly promising approach, as it shifts machine learning to a higher level of abstraction. The design of new pipelines for new medical image analysis problems could be fully automatized with only little need for human interaction. While NAS has shown some initial success [302, 303, 603], its adoption in the medical image domain is still minimal [536]. Given the large number of potential medical image analysis problems, NAS could enable well-designed deep learning models, even for niche problems. Therefore, we encourage more work on NAS for medical image analysis problems.

A major downside of NAS remains its computational effort. While recent NAS approaches have found mechanisms to reduce search time significantly [377], ideas exploiting data properties are still rare. We proposed to perform the search on lower-dimensional data to find architectures that transfer well to higher dimensions. This idea could be extended further to higher image dimensions, for example, 3D or 4D. Also, it could be expanded to different types of deep learning models, for example, convolutional-recurrent models. To ensure high performance on higher-dimensional data, additional constraints or search space augmentations could be introduced that partially incorporate higher-dimensional context during the search phase.

We also considered the immediate transfer of successful 2D CNN architectures to 3D by using multi-dimensional transfer learning. A straight-forward approach is to attempt an extension of 2D CNNs to 4D. While this is unfeasible with today's deep learning hardware, future work might be able to exploit pretrained 2D CNNs in higher dimensions.

We also touched on a deep learning mechanism that is usually referred to as attention [509]. This type of processing method has gained popularity in the medical domain in recent years [418]. In our work, we use attention for information exchange between time points, within the temporal data dimension of 3.5D data. Taking a multi-dimensional perspective, future work could try to design attention mechanisms that model the relationship between different data dimensions, for example, spatial and temporal. This could allow for improved capturing of higher-dimensional context.

An orthogonal perspective that we did not consider throughout this thesis is the size of individual data dimensions. "Higher-dimensional" is often interpreted in terms of the size of individual data dimensions. While the size of data dimensions has often been neglected in deep learning applications, recent work has focused on designing CNNs that specifically work well for a certain size of spatial data dimensions [482].

This aspect of data dimensions will also be interesting for a lot of medical problems as data dimension sizes can also be expected to increase with improved high-resolution scanning devices [350]. For example, attention mechanisms could be explored to focus on the relevant part of each data dimension, which we already found to be effective for skin lesion classification [165].

8.2.2 Deep Learning Applications

In terms of deep learning applications, a straightforward open problem for future work is the application of our adapted and proposed methods for other medical image analysis problems. While we validated our methods across different problems, additional research could provide additional evidence for the effectiveness of our methods. For example, recent results on fMRI-based [43] and hyperspectral [42] image processing have demonstrated that our recurrent-convolutional architecture concept we proposed in Section 4.3 appears to be effective for a larger number of problems.

When extending our insights to other medical image analysis problems, a straightforward choice is the imaging modality US due to its similarity to OCT. Both modalities rely on wave scattering within tissue or material. Also, typical data representations include 2D, 3D spatial, and spatio-temporal, as well as 4D spatio-temporal data. In particular, our insights on 4D data representations could be useful for different US problems. The problem of pose and motion estimation is also frequently encountered in US image processing, for example, for radiation therapy [434]. Furthermore, tissue classification is also frequently addressed with US [209].

The corresponding counterpart for MRI is the imaging modality CT. Both modalities come with a similar spatial resolution and often address similar problems, such as tissue segmentation. In terms of data dimensionality, similar to MRI, deep learning for CT is currently undergoing the switch from 2D slices to full 3D volumes. In terms of the acquisition, 4D CT has already been researched for quite some time [535]. Still, 4D deep learning approaches are rare [210], which could be the focus of future work. Methods such as our cGRU-CNN-U architecture we proposed in Section 4.3 might be useful for spatio-temporal CT segmentation problems.

Looking a step further into the future, the next data representation that could be considered is 5D. We considered both short-term 4D spatio-temporal data and long-term 4D spatio-temporal data. Combining both representations leads to a longitudinal sequence of short-term 4D sequences. While both data dimensions are temporal in nature, treating them separately is likely helpful as they cover very different contexts. Not too long ago, 3D deep learning has been considered "a nightmare" [327], and in this thesis we found 4D deep learning very challenging. Nevertheless, given that the improvements in terms of computational resources and imaging systems continue, 5D deep learning should become feasible in the future.

8.3 Closing Thoughts

Over the last few years, deep learning has made a significant impact on medical image analysis research. Improvements in computational resources have enabled the development of better deep learning models in the natural image domain that have been increasingly employed in the medical domain. The initial success of a few deep learning-based medical image analysis problems has spread across countless problems ever since. This results in a vast number of deep learning applications with different imaging modalities and data representations to be considered.

In our work, we have made a significant leap forward for a number of deep learning approaches. Most importantly, we provided a new, multi-dimensional view on medical image analysis. We highlighted the challenges and benefits of different data dimensions across problems. We pioneered 4D deep learning for medical image analysis, and our insights all point into the same direction: High-dimensional data processing is very promising and will play a significant role in the future of medical image analysis.

Bibliography

[1] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., et al., C.C.: TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems (2015), `http://tensorflow.org/`, software available from tensorflow.org

[2] Abdolmanafi, A., Cheriet, F., Duong, L., Ibrahim, R., Dahdah, N.: An automatic diagnostic system of coronary artery lesions in Kawasaki disease using intravascular optical coherence tomography imaging. Journal of Biophotonics 13(1), e201900112 (2020)

[3] Abdolmanafi, A., Duong, L., Dahdah, N., Adib, I.R., Cheriet, F.: Characterization of coronary artery pathological formations from OCT imaging using deep learning. Biomedical Optics Express 9(10), 4936–4960 (2018)

[4] Abdolmanafi, A., Duong, L., Dahdah, N., Cheriet, F.: Deep feature learning for automatic tissue classification of coronary artery using optical coherence tomography. Biomedical Optics Express 8(2), 1203–1220 (2017)

[5] Abdolmanafi, A., Prasad, A.S., Duong, L., Dahdah, N.: Classification of coronary artery tissues using optical coherence tomography imaging in kawasaki disease. In: Medical Imaging 2016: Image-Guided Procedures, Robotic Interventions, and Modeling. vol. 9786, p. 97862U. International Society for Optics and Photonics (2016)

[6] Abolhassani, N., Patel, R., Moallem, M.: Needle insertion into soft tissue: A survey. Medical Engineering and Physics 29(4), 413–431 (2007)

[7] Adhi, M., Duker, J.S.: Optical coherence tomography–current and future applications. Current Opinion in Ophthalmology 24(3), 213 (2013)

[8] Aït-Ali, L.S., Prima, S., Hellier, P., Carsin, B., Edan, G., Barillot, C.: STREM: a robust multidimensional parametric method to segment MS lesions in MRI. In: International Conference on Medical Image Computing and Computer-assisted Intervention. pp. 409–416. Springer (2005)

[9] Akselrod-Ballin, A., Galun, M., Gomori, J.M., Filippi, M., Valsasina, P., Basri, R., Brandt, A.: Automatic segmentation and classification of multiple sclerosis in multichannel MRI. IEEE Transactions on Biomedical Engineering 56(10), 2461–2469 (2009)

[10] Al Hajj, H., Lamard, M., Charrière, K., Cochener, B., Quellec, G.: Surgical tool detection in cataract surgery videos through multi-image fusion inside a convolutional neural network. In: International Conference of the IEEE Engineering in Medicine and Biology Society. pp. 2002–2005. IEEE (2017)

[11] Al Hajj, H., Lamard, M., Conze, P.H., Roychowdhury, S., Hu, X., Maršalkaitė, G., Zisimopoulos, O., Dedmari, M.A., Zhao, F., Prellberg, J., et al.: CATARACTS: Challenge on automatic tool annotation for cataRACT surgery. Medical Image Analysis 52, 24–41 (2019)

[12] Alibhai, A.Y., Or, C., Witkin, A.J.: Swept Source Optical Coherence Tomography: a Review. Current Ophthalmology Reports 6(1), 7–16 (2018)

[13] Allan, M., Thompson, S., Clarkson, M.J., Ourselin, S., Hawkes, D.J., Kelly, J., Stoyanov, D.: 2D-3D pose tracking of rigid instruments in minimally invasive surgery. In: International Conference on Information Processing in Computer-assisted Interventions. pp. 1–10. Springer (2014)

[14] Alom, M.Z., Yakopcic, C., Hasan, M., Taha, T.M., Asari, V.K.: Recurrent residual U-Net for medical image segmentation. Journal of Medical Imaging 6(1), 014006 (2019)

[15] Alpaydin, E.: Introduction to machine learning. MIT press (2020)

[16] Alsaih, K., Lemaitre, G., Rastgoo, M., Massich, J., Sidibé, D., Meriaudeau, F.: Machine learning techniques for diabetic macular edema (DME) classification on SD-OCT images. Biomedical Engineering Online 16(1), 68 (2017)

[17] An, G., Yokota, H., Motozawa, N., Takagi, S., Mandai, M., Kitahata, S., Hirami, Y., Takahashi, M., Kurimoto, Y., Akiba, M.: Deep Learning Classification Models Built with Two-step Transfer Learning for Age Related Macular Degeneration Diagnosis. In: International Conference of the IEEE Engineering in Medicine and Biology Society. pp. 2049–2052. IEEE (2019)

[18] Anthimopoulos, M., Christodoulidis, S., Ebner, L., Christe, A., Mougiakakou, S.: Lung pattern classification for interstitial lung diseases using a deep convolutional neural network. IEEE Transactions on Medical Imaging 35(5), 1207–1216 (2016)

[19] Asadi-Aghbolaghi, M., Clapes, A., Bellantonio, M., Escalante, H.J., Ponce-López, V., Baró et al., X.: A survey on deep learning based approaches for action and gesture recognition in image sequences. In: International Conference on Automatic Face & Gesture Recognition. pp. 476–483. IEEE (2017)

[20] Aslani, S., Dayan, M., Storelli, L., Filippi, M., Murino, V., Rocca, M.A., Sona, D.: Multi-branch convolutional neural network for multiple sclerosis lesion segmentation. NeuroImage 196, 1–15 (2019)

[21] Athanasiou, L.S., Bourantas, C.V., Rigas, G., Sakellarios, A.I., Exarchos, T.P., Siogkas, P.K., Ricciardi, A., Naka, K.K., Papafaklis, M.I., Michalis, L.K., et al.: Methodology for fully automated segmentation and plaque characterization in intracoronary optical coherence tomography images. Journal of Biomedical Optics 19(2), 026009 (2014)

[22] Athanasiou, L.S., Olender, M.L., José, M., Ben-Assa, E., Edelman, E.R.: A deep learning approach to classify atherosclerosis using intracoronary optical coherence tomography. In: Medical Imaging 2019: Computer-Aided Diagnosis. vol. 10950, p. 109500N. International Society for Optics and Photonics (2019)

[23] Atkinson, D.J., Edelman, R.: Cineangiography of the heart in a single breath hold with a segmented turboFLASH sequence. Radiology 178(2), 357–360 (1991)

[24] Atlason, H.E., Love, A., Sigurdsson, S., Gudnason, V., Ellingsen, L.M.: SegAE: Unsupervised white matter lesion segmentation from brain MRIs using a CNN autoencoder. NeuroImage: Clinical 24, 102085 (2019)

[25] Atlason, H.E., Love, A., Sigurdsson, S., Gudnason, V., Ellingsen, L.M.: Unsupervised brain lesion segmentation from MRI using a convolutional autoencoder. In: Medical Imaging 2019: Image Processing. vol. 10949, p. 109491H. International Society for Optics and Photonics (2019)

[26] Avendi, M., Kheradvar, A., Jafarkhani, H.: A combined deep-learning and deformable-model approach to fully automatic segmentation of the left ventricle in cardiac MRI. Medical Image Analysis 30, 108–119 (2016)

[27] Aviles, A.I., Alsaleh, S.M., Hahn, J.K., Casals, A.: Towards retrieving force feedback in robotic-assisted surgery: A supervised neuro-recurrent-vision approach. IEEE Transactions on Haptics 10(3), 431–443 (2017)

[28] Aviles, A.I., Alsaleh, S.M., Sobrevilla, P., Casals, A.: Force-feedback sensory substitution using supervised recurrent learning for robotic-assisted surgery. In: International Conference of the IEEE Engineering in Medicine and Biology Society. pp. 1–4. IEEE (2015)

[29] Aviles, A.I., Marban, A., Sobrevilla, P., Fernandez, J., Casals, A.: A recurrent neural network approach for 3d vision-based force estimation. In: 2014 4th International Conference on Image Processing Theory, Tools and Applications (IPTA). pp. 1–6. IEEE (2014)

[30] Azarmehr, N., Ye, X., Janan, F., Howard, J.P., Francis, D.P., Zolgharni, M., et al.: Automated Segmentation of Left Ventricle in 2D echocardiography using deep learning. In: Medical Imaging with Deep Learning (2019)

[31] Badrinarayanan, V., Kendall, A., Cipolla, R.: Segnet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence 39(12), 2481–2495 (2017)

[32] Ballas, N., Yao, L., Pal, C., Courville, A.: Delving deeper into convolutional networks for learning video representations. In: International Conference on Learning Representations (2016)

[33] Bash, S., Villablanca, J.P., Jahan, R., Duckwiler, G., Tillis, M., Kidwell, C., Saver, J., Sayre, J.: Intracranial vascular stenosis and occlusive disease: evaluation with

CT angiography, MR angiography, and digital subtraction angiography. American journal of neuroradiology 26(5), 1012–1021 (2005)

[34] Battaglini, M., et al.: Automated identification of brain new lesions in multiple sclerosis using subtraction images. Journal of Magnetic Resonance Imaging 39(6), 1543–1549 (2014)

[35] Baur, C., Wiestler, B., Albarqouni, S., Navab, N.: Deep autoencoding models for unsupervised anomaly segmentation in brain MR images. In: International MICCAI Brainlesion Workshop. pp. 161–169. Springer (2018)

[36] Baur, C., Wiestler, B., Albarqouni, S., Navab, N.: Fusing unsupervised and supervised deep learning for white matter lesion segmentation. In: Medical Imaging with Deep Learning. pp. 63–72 (2019)

[37] Bayle, B., Joinie-Maurin, M., Barbe, L., Gangloff, J., De Mathelin, M.: Robot interaction control in medicine and surgery: Original results and open problems. In: Computational Surgery and Dual Training, pp. 169–191. Springer (2014)

[38] Beekmans, S., Lembrechts, T., van den Dobbelsteen, J., van Gerwen, D.: Fiber-Optic Fabry-Pérot Interferometers for Axial Force Sensing on the Tip of a Needle. Sensors 17(1), 38 (2016)

[39] Bello, I., Zoph, B., Vasudevan, V., Le, Q.V.: Neural optimizer search with reinforcement learning. In: Proceedings of the 34th International Conference on Machine Learning. pp. 459–468 (2017)

[40] Ben-Cohen, A., Mark, D., Kovler, I., Zur, D., Barak, A., Iglicki, M., Soferman, R.: Retinal layers segmentation using fully convolutional network in OCT images. RSIP Vision (2017)

[41] Bengio, Y., Simard, P., Frasconi, P., et al.: Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks 5(2), 157–166 (1994)

[42] Bengs, M., Gessert, N., Laffers, W., Eggert, D., Westermann, S., Mueller, N.A., Gerstner, A.O., Betz, C., Schlaefer, A.: Spectral-Spatial Recurrent-Convolutional Networks for In-Vivo Hyperspectral Tumor Type Classification. In: International Conference on Medical Image Computing and Computer-assisted Intervention. pp. 1–10. Springer (2020)

[43] Bengs, M., Gessert, N., Schlaefer, A.: 4D Spatio-Temporal Deep Learning with 4D fMRI Data for Autism Spectrum Disorder Classification. In: Medical Imaging with Deep Learning (2019)

[44] Bengs, M., Gessert, N., Schlüter, M., Schlaefer, A.: Spatio-temporal deep learning methods for motion estimation using 4D OCT image data. International Journal of Computer Assisted Radiology and Surgery 15, 943–952 (2020)

[45] Bergmeier, J., Fitzpatrick, J.M., Daentzer, D., Majdani, O., Ortmaier, T., Kahrs, L.A.: Workflow and simulation of image-to-physical registration of holes inside spongy bone. International Journal of Computer Assisted Radiology and Surgery 12(8), 1425–1437 (2017)

[46] Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. Journal of Machine Learning Research 13(Feb), 281–305 (2012)

[47] Bernal, J., Tajkbaksh, N., Sánchez, F.J., Matuszewski, B.J., Chen, H., Yu, L., Angermann, Q., Romain, O., Rustad, B., Balasingham, I., et al.: Comparative validation of polyp detection methods in video colonoscopy: results from the MICCAI 2015 endoscopic vision challenge. IEEE Transactions on Medical Imaging 36(6), 1231–1249 (2017)

[48] Bibic, A., Knutsson, L., Ståhlberg, F., Wirestam, R.: Denoising of arterial spin labeling data: wavelet-domain filtering compared with Gaussian smoothing. Magnetic Resonance Materials in Physics, Biology and Medicine 23(3), 125–137 (2010)

[49] Binder, S., Falkner-Radler, C.I., Hauger, C., Matz, H., Glittenberg, C.: Feasibility of intrasurgical spectral-domain optical coherence tomography. Retina 31(7), 1332–1336 (2011)

[50] Birenbaum, A., Greenspan, H.: Longitudinal multiple sclerosis lesion segmentation using multi-view convolutional neural networks. In: Deep Learning and Data Labeling for Medical Applications, pp. 58–67. Springer (2016)

[51] Birenbaum, A., Greenspan, H.: Multi-view longitudinal CNN for multiple sclerosis lesion segmentation. Engineering Applications of Artificial Intelligence 65, 111–118 (2017)

[52] Bishop, C.M.: Regularization and complexity control in feed-forward networks, `http://publications.aston.ac.uk/524/`

[53] Bogaert, J., Dymarkowski, S., Taylor, A.M., Muthurangu, V.: Clinical cardiac MRI. Springer Science & Business Media (2012)

[54] Boldea, V., Sharp, G.C., Jiang, S.B., Sarrut, D.: 4D-CT lung motion estimation with deformable registration: quantification of motion nonlinearity and hysteresis. Medical Physics 35(3), 1008–1018 (2008)

[55] Borchani, H., Varando, G., Bielza, C., Larrañaga, P.: A survey on multi-output regression. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 5(5), 216–233 (2015)

[56] Bottou, L.: Large-scale machine learning with stochastic gradient descent. In: Proceedings of COMPSTAT'2010, pp. 177–186. Springer (2010)

[57] Bouget, D., Allan, M., Stoyanov, D., Jannin, P.: Vision-based and marker-less surgical tool detection and tracking: a review of the literature. Medical Image Analysis 35, 633–654 (2017)

[58] Bouma, B.E., Tearney, G.J.: Power-efficient nonreciprocal interferometer and linear-scanning fiber-optic catheter for optical coherence tomography. Optics Letters 24(8), 531–533 (1999)

[59] Brachmann, E., Krull, A., Michel, F., Gumhold, S., Shotton, J., Rother, C.: Learning 6d object pose estimation using 3d object coordinates. In: Proceedings of the European Conference on Computer Vision. pp. 536–551 (2014)

[60] Brex, P.A., Ciccarelli, O., O'Riordan, J.I., Sailer, M., Thompson, A.J., Miller, D.H.: A longitudinal study of abnormalities on MRI and disability from multiple sclerosis. New England Journal of Medicine 346(3), 158–164 (2002)

[61] Brezinski, M.E.: Optical coherence tomography: principles and applications. Academic press (2006)

[62] Bricq, S., Collet, C., Armspach, J.P.: Markovian segmentation of 3D brain MRI to detect multiple sclerosis lesions. In: 2008 15th IEEE International Conference on Image Processing. pp. 733–736. IEEE (2008)

[63] Brosch, T., Yoo, Y., Tang, L.Y., Li, D.K., Traboulsee, A., Tam, R.: Deep convolutional encoder networks for multiple sclerosis lesion segmentation. In: International Conference on Medical Image Computing and Computer-assisted Intervention. pp. 3–11. Springer (2015)

[64] Brosch, T., et al.: Deep 3D convolutional encoder networks with shortcuts for multiscale feature integration applied to multiple sclerosis lesion segmentation. IEEE Transactions on Medical Imaging 35(5), 1229–1239 (2016)

[65] Burlina, P.M., Joshi, N., Pekala, M., Pacheco, K.D., Freund, D.E., Bressler, N.M.: Automated grading of age-related macular degeneration from color fundus images using deep convolutional neural networks. JAMA Ophthalmology 135(11), 1170–1176 (2017)

[66] Bus, M.T., Muller, B.G., de Bruin, D.M., Faber, D.J., Kamphuis, G.M., van Leeuwen, T.G., de Reijke, T.M., de la Rosette, J.J.: Volumetric in vivo visualization of upper urinary tract tumors using optical coherence tomography: a pilot study. The Journal of Urology 190(6), 2236–2242 (2013)

[67] Bushberg, J.T.: The essential physics of medical imaging. Lippincott Williams & Wilkins (2002)

[68] Byra, M., Galperin, M., Ojeda-Fournier, H., Olson, L., O'Boyle, M., Comstock, C., Andre, M.: Breast mass classification in sonography with transfer learning using a deep convolutional neural network and color conversion. Medical Physics 46(2), 746–755 (2019)

[69] Cabezas, M., et al.: Improved automatic detection of new T2 lesions in multiple sclerosis using deformation fields. American Journal of Neuroradiology 37(10), 1816–1823 (2016)

[70] Caiani, E.G., Corsi, C., Zamorano, J., Sugeng, L., MacEneaney, P., Weinert, L., Battani, R., Gutierrez, J.L., Koch, R., de Isla, L.P., et al.: Improved semi-automated quantification of left ventricular volumes and ejection fraction using 3-dimensional echocardiography with a full matrix-array transducer: comparison with magnetic resonance imaging. Journal of the American Society of Echocardiography 18(8), 779–788 (2005)

[71] Callaghan, P.T.: Principles of nuclear magnetic resonance microscopy. Oxford University Press on Demand (1993)

[72] Camino, A., Zhang, M., Gao, S.S., Hwang, T.S., Sharma, U., Wilson, D.J., Huang, D., Jia, Y.: Evaluation of artifact reduction in optical coherence tomography angiography with real-time tracking and motion correction technology. Biomedical Optics Express 7(10), 3905–3915 (2016)

[73] Canny, J.: A computational approach to edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence 8(6), 679–698 (1986)

[74] Carass, A., Roy, S., Jog, A., Cuzzocreo, J.L., Magrath, E., Gherman, A., Button, J., Nguyen, J., Prados, F., Sudre, C.H., et al.: Longitudinal multiple sclerosis lesion segmentation: resource and challenge. NeuroImage 148, 77–102 (2017)

[75] Carr, J.C., Simonetti, O., Bundy, J., Li, D., Pereles, S., Finn, J.P.: Cine MR angiography of the heart with segmented true fast imaging with steady-state precession. Radiology 219(3), 828–834 (2001)

[76] Carrasco-Zevallos, O., Keller, B., Viehland, C., Shen, L., Waterman, G., Todorich, B., Shieh, C., Hahn, P., Farsiu, S., Kuo, A., et al.: Live volumetric (4D) visualization and guidance of in vivo human ophthalmic surgery with intraoperative optical coherence tomography. Scientific Reports 6, 31689 (2016)

[77] Celi, S., Berti, S.: In-vivo segmentation and quantification of coronary lesions by optical coherence tomography images for a lesion type definition and stenosis grading. Medical Image Analysis 18(7), 1157–1168 (2014)

[78] Chabat, F., Hansell, D.M., Yang, G.Z.: Computerized decision support in medical imaging. IEEE Engineering in medicine and Biology Magazine 19(5), 89–96 (2000)

[79] Cheimariotis, G.A., Riga, M., Toutouzas, K., Tousoulis, D., Katsaggelos, A., Maglaveras, N.: Deep Learning Method to Detect Plaques in IVOCT Images. In: International Conference on Biomedical and Health Informatics. pp. 389–395. Springer (2019)

[80] Chen, H., Dou, Q., Wang, X., Qin, J., Cheng, J.C., Heng, P.A.: 3D fully convolutional networks for intervertebral disc localization and segmentation. In: International Conference on Medical Imaging and Augmented Reality. pp. 375–382. Springer (2016)

[81] Chen, H., Dou, Q., Yu, L., Qin, J., Heng, P.A.: VoxResNet: Deep voxelwise residual networks for brain segmentation from 3D MR images. NeuroImage 170, 446–455 (2018)

[82] Chen, H., Zheng, Y., Park, J.H., Heng, P.A., Zhou, S.K.: Iterative multi-domain regularized deep learning for anatomical structure detection and segmentation from ultrasound images. In: International Conference on Medical Image Computing and Computer-assisted Intervention. pp. 487–495. Springer (2016)

[83] Chen, L.C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H.: Encoder-decoder with atrous separable convolution for semantic image segmentation. In: Proceedings of the European Conference on Computer Vision. pp. 801–818 (2018)

[84] Chen, Y., Xie, Y., Zhou, Z., Shi, F., Christodoulou, A.G., Li, D.: Brain MRI super resolution using 3D deep densely connected neural networks. In: IEEE International Symposium on Biomedical Imaging. pp. 739–742. IEEE (2018)

[85] Chen, Y., Sun, Q.L., Zhong, K.: Semi-supervised spatio-temporal CNN for recognition of surgical workflow. EURASIP Journal on Image and Video Processing 2018(1), 76 (2018)

[86] Cheng, M., et al.: A Multi-scale Multiple Sclerosis Lesion Change Detection in a Multi-sequence MRI. In: DLMIA 2018. pp. 353–360 (2018)

[87] Cho, K., Gulcehre, C., van Merrienboer, B., Bougares, F., Schwenk, H., Bengio, Y.: Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In: Conference on Empirical Methods in Natural Language Processing (2014)

[88] Choy, C., Gwak, J., Savarese, S.: 4d spatio-temporal convnets: Minkowski convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3075–3084 (2019)

[89] Christ, P.F., Elshaer, M.E.A., Ettlinger, F., Tatavarty, S., Bickel, M., Bilic, P., Rempfler, M., Armbruster, M., Hofmann, F., D'Anastasi, M., et al.: Automatic liver and lesion segmentation in CT using cascaded fully convolutional neural networks and 3D conditional random fields. In: International Conference on Medical Image Computing and Computer-assisted Intervention. pp. 415–423. Springer (2016)

[90] Ciernik, I.F., Dizendorf, E., Baumert, B.G., Reiner, B., Burger, C., Davis, J.B., Lütolf, U.M., Steinert, H.C., Von Schulthess, G.K.: Radiation treatment planning with an integrated positron emission and computer tomography (PET/CT): a feasibility study. International Journal of Radiation Oncology* Biology* Physics 57(3), 853–863 (2003)

[91] Ciresan, D.C., Meier, U., Masci, J., Gambardella, L.M., Schmidhuber, J.: Flexible, high performance convolutional neural networks for image classification. In: Twenty-Second International Joint Conference on Artificial Intelligence (2011)

[92] Clark, D., Badea, C.: Convolutional regularization methods for 4D, x-ray CT reconstruction. In: Medical Imaging 2019: Physics of Medical Imaging. vol. 10948, p. 109482A. International Society for Optics and Photonics (2019)

[93] Colleoni, E., Moccia, S., Du, X., De Momi, E., Stoyanov, D.: Deep learning based robotic tool detection and articulation estimation with spatio-temporal layers. IEEE Robotics and Automation Letters 4(3), 2714–2721 (2019)

[94] Cooijmans, T., Ballas, N., Laurent, C., Gülçehre, Ç., Courville, A.: Recurrent batch normalization. arXiv preprint arXiv:1603.09025 (2016)

[95] Costante, G., Mancini, M., Valigi, P., Ciarfuglia, T.A.: Exploring representation learning with cnns for frame-to-frame ego-motion estimation. IEEE Robotics and Automation Letters 1(1), 18–25 (2015)

[96] Crane, R.: A Simplified Approach to Image Processing. Prentice Hall, Upper Saddle River, NJ (1997)

[97] Crum, W.R., Camara, O., Hill, D.L.: Generalized overlap measures for evaluation and validation in medical image analysis. IEEE Transactions on Medical Imaging 25(11), 1451–1461 (2006)

[98] Das, V., Dandapat, S., Bora, P.K.: Multi-scale deep feature fusion for automated classification of macular pathologies from OCT images. Biomedical Signal Processing and Control 54, 101605 (2019)

[99] De Cock, D., Tu, S., Ughi, G.J., Adriaenssens, T.: Development of 3d ivoct imaging and co-registration of ivoct and angiography in the catheterization laboratory. Current Cardiovascular Imaging Reports 7(10), 9290 (2014)

[100] De Fauw, J., Ledsam, J.R., Romera-Paredes, B., Nikolov, S., Tomasev, N., Blackwell, S., Askham, H., Glorot, X., O'Donoghue, B., Visentin, D., et al.: Clinically applicable deep learning for diagnosis and referral in retinal disease. Nature Medicine 24(9), 1342–1350 (2018)

[101] De Lorenzo, D., De Momi, E., Dyagilev, I., Manganelli, R., Formaglio, A., Prattichizzo, D., Shoham, M., Ferrigno, G.: Force feedback in a piezoelectric linear actuator for neurosurgery. The International Journal of Medical Robotics and Computer Assisted Surgery 7(3), 268–275 (2011)

[102] Deng, J., Xie, X., Terry, L., Wood, A., White, N., Margrain, T.H., North, R.V.: Age-related macular degeneration detection and stage classification using choroidal oct images. In: International Conference on Image Analysis and Recognition. pp. 707–715. Springer (2016)

[103] Deshpande, G., Wang, P., Rangaprakash, D., Wilamowski, B.: Fully connected cascade artificial neural network architecture for attention deficit hyperactivity disorder classification from functional magnetic resonance imaging data. IEEE Transactions on Cybernetics 45(12), 2668–2679 (2015)

[104] Detre, J.A., Alsop, D., Vives, L., Maccotta, L., Teener, J., Raps, E.: Noninvasive MRI evaluation of cerebral blood flow in cerebrovascular disease. Neurology 50(3), 633–641 (1998)

[105] Diana, M., Marescaux, J.: Robotic surgery. British Journal of Surgery 102(2) (2015)

[106] Diba, A., Fayyaz, M., Sharma, V., Karami, A.H., Arzani, M.M., Yousefzadeh, R., Van Gool, L.: Temporal 3d convnets: New architecture and transfer learning for video classification. arXiv preprint arXiv:1711.08200 (2017)

[107] Dice, L.R.: Measures of the amount of ecologic association between species. Ecology 26(3), 297–302 (1945)

[108] Donahue, J., Anne Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko et al., K.: Long-term recurrent convolutional networks for visual recognition and description. In: Conference on Computer Vision and Pattern Recognition. pp. 2625–2634 (2015)

[109] Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., van der Smagt, P., Cremers, D., Brox, T.: Flownet: Learning optical flow with convolutional networks. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2758–2766 (2015)

[110] Dosovitskiy, A., Fischer, P., Ilg, E., Hausser, P., Hazirbas, C., Golkov, V., Van Der Smagt, P., Cremers, D., Brox, T.: Flownet: Learning optical flow with convolutional networks. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2758–2766 (2015)

[111] Dou, Q., Chen, H., Yu, L., Zhao, L., Qin, J., Wang, D., Mok, V.C.T., Shi, L., Heng, P.A.: Automatic detection of cerebral microbleeds from MR images via 3D convolutional neural networks. IEEE Transactions on Medical Imaging 35(5), 1182–1195 (2016)

[112] Dou, Q., Yu, L., Chen, H., Jin, Y., Yang, X., Qin, J., Heng, P.A.: 3D deeply supervised network for automated segmentation of volumetric medical images. Medical Image Analysis (2017)

[113] Dozat, T.: Incorporating nesterov momentum into adam. In: International Conference on Learning Representations 2016 Workshop (2016)

[114] Drexler, W., Morgner, U., Ghanta, R.K., Kärtner, F.X., Schuman, J.S., Fujimoto, J.G.: Ultrahigh-resolution ophthalmic optical coherence tomography. Nature Medicine 7(4), 502–507 (2001)

[115] Du, X., Kurmann, T., Chang, P.L., Allan, M., Ourselin, S., Sznitman, R., Kelly, J.D., Stoyanov, D.: Articulated multi-instrument 2-D pose estimation using fully convolutional networks. IEEE Transactions on Medical Imaging 37(5), 1276–1287 (2018)

[116] Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research 12(Jul), 2121–2159 (2011)

[117] Dvornek, N.C., Ventola, P., Pelphrey, K.A., Duncan, J.S.: Identifying autism from resting-state fMRI using long short-term memory networks. In: International Workshop on Machine Learning in Medical Imaging. pp. 362–370. Springer (2017)

[118] Egger, C., et al.: MRI FLAIR lesion segmentation in multiple sclerosis: does automated segmentation hold up with manual annotation? NeuroImage: Clinical 13, 264–270 (2017)

[119] Egmont-Petersen, M., de Ridder, D., Handels, H.: Image processing with neural networks—a review. Pattern recognition 35(10), 2279–2301 (2002)

[120] Ehlers, J.P., Kernstine, K., Farsiu, S., Sarin, N., Maldonado, R., Toth, C.A.: Analysis of pars plana vitrectomy for optic pit–related maculopathy with intraoperative optical coherence tomography: a possible connection with the vitreous cavity. Archives of Ophthalmology 129(11), 1483–1486 (2011)

[121] Ehlers, J.P., Srivastava, S.K., Feiler, D., Noonan, A.I., Rollins, A.M., Tao, Y.K.: Integrative advances for OCT-guided ophthalmic surgery and intraoperative OCT: microscope integration, surgical instrumentation, and heads-up display surgeon feedback. PloS one 9(8) (2014)

[122] Ehlers, J.P., Tam, T., Kaiser, P.K., Martin, D.F., Smith, G.M., Srivastava, S.K.: Utility of intraoperative optical coherence tomography during vitrectomy surgery for vitreomacular traction syndrome. Retina (Philadelphia, Pa.) 34(7), 1341 (2014)

[123] Ehlers, J.P., Uchida, A., Srivastava, S.K.: THE INTEGRATIVE SURGICAL THEATER: Combining Intraoperative Optical Coherence Tomography and 3D Digital Visualization for Vitreoretinal Surgery in the DISCOVER Study. Retina (Philadelphia, Pa.) 38(Suppl 1), S88–S96 (2018)

[124] El Sallab, A., Sobh, I., Zidan, M., Zahran, M., Abdelkarim, S.: Yolo4d: A spatio-temporal approach for real-time multi-object detection and classification from lidar point clouds. In: Conference on Neural Information Processing Systems 2018 Workshop MLITS (2018)

[125] Eladawi, N., Elmogy, M., Helmy, O., Aboelfetouh, A., Riad, A., Sandhu, H., Schaal, S., El-Baz, A.: Automatic blood vessels segmentation based on different retinal maps from OCTA scans. Computers in Biology and Medicine 89, 150–161 (2017)

[126] Elfring, R., de La Fuente, M., Radermacher, K.: Assessment of optical localizer accuracy for computer aided surgery systems. Computer Aided Surgery 15(1-3), 1–12 (2010)

[127] Ellebrecht, D.B., Latus, S., Schlaefer, A., Keck, T., Gessert, N.: Towards an optical biopsy during visceral surgical interventions. Visceral Medicine (2020)

[128] Elmore, J.G., Wells, C.K., Lee, C.H., Howard, D.H., Feinstein, A.R.: Variability in radiologists' interpretations of mammograms. New England Journal of Medicine 331(22), 1493–1499 (1994)

[129] Esteva, A., Kuprel, B., Novoa, R.A., Ko, J., Swetter, S.M., Blau, H.M., Thrun, S.: Dermatologist-level classification of skin cancer with deep neural networks. Nature 542(7639), 115–118 (2017)

[130] Fan, Z., Wei, J., Zhu, G., Mo, J., Li, W.: ENAS U-Net: Evolutionary Neural Architecture Search for Retinal Vessel Segmentation. arXiv preprint arXiv:2001.06678 (2020)

[131] Fang, L., Cunefare, D., Wang, C., Guymer, R.H., Li, S., Farsiu, S.: Automatic segmentation of nine retinal layer boundaries in OCT images of non-exudative AMD patients using deep learning and graph search. Biomedical Optics Express 8(5), 2732–2744 (2017)

[132] Fang, R., Huang, J., Luh, W.M.: A spatio-temporal low-rank total variation approach for denoising arterial spin labeling MRI data. In: IEEE International Symposium on Biomedical Imaging. pp. 498–502. IEEE (2015)

[133] Faragasso, A., Bimbo, J., Noh, Y., Jiang, A., Sareh, S., Liu, H., Nanayakkara, T., Wurdemann, H.A., Althoefer, K.: Novel uniaxial force sensor based on visual information for minimally invasive surgery. In: International Conference on Robotics and Automation. pp. 1405–1410 (2014)

[134] Farsiu, S., Chiu, S.J., O'Connell, R.V., Folgar, F.A., Yuan, E., Izatt, J.A., Toth, C.A., Group, A.R.E.D.S..A.S.D.O.C.T.S., et al.: Quantitative classification of eyes with and without intermediate age-related macular degeneration using optical coherence tomography. Ophthalmology 121(1), 162–172 (2014)

[135] Fenneteau, A., Bourdon, P., Helbert, D., Fernandez-Maloigne, C., Habas, C., Guillevin, R.: Learning a CNN on multiple sclerosis lesion segmentation with self-supervision. In: 3D Measurement and Data Processing, IS&T Electronic Imaging 2020 Symposium (2020)

[136] Fercher, A.F., Mengedoht, K., Werner, W.: Eye-length measurement by interferometry with partially coherent light. Optics Letters 13(3), 186–188 (1988)

[137] Fernandez, D.C.: Delineating fluid-filled region boundaries in optical coherence tomography images of the retina. IEEE Transactions on Medical Imaging 24(8), 929–945 (2005)

[138] Fischmann, A., Hafner, P., Fasler, S., Gloor, M., Bieri, O., Studler, U., Fischer, D.: Quantitative MRI can detect subclinical disease progression in muscular dystrophy. Journal of Neurology 259(8), 1648–1654 (2012)

[139] Franz, A.M., Haidegger, T., Birkfellner, W., Cleary, K., Peters, T.M., Maier-Hein, L.: Electromagnetic tracking in medicine—a review of technology, validation, and applications. IEEE Transactions on Medical Imaging 33(8), 1702–1725 (2014)

[140] Friston, K.J., Jezzard, P., Turner, R.: Analysis of functional MRI time-series. Human Brain Mapping 1(2), 153–171 (1994)

[141] Fu, J., Yang, Y., Singhrao, K., Ruan, D., Chu, F.I., Low, D.A., Lewis, J.H.: Deep learning approaches using 2D and 3D convolutional neural networks for generating male pelvic synthetic computed tomography from magnetic resonance imaging. Medical Physics 46(9), 3788–3798 (2019)

[142] Funke, I., Jenke, A., Mees, S.T., Weitz, J., Speidel, S., Bodenstedt, S.: Temporal coherence-based self-supervised learning for laparoscopic workflow analysis. In: OR 2.0 Context-Aware Operating Theaters, Computer Assisted Robotic Endoscopy, Clinical Image-Based Procedures, and Skin Image Analysis, pp. 85–93. Springer (2018)

[143] Gal, Y., Ghahramani, Z.: Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In: International Conference on Machine Learning. pp. 1050–1059 (2016)

[144] Gal, Y., Ghahramani, Z.: A theoretically grounded application of dropout in recurrent neural networks. In: Advances in Neural Information Processing Systems. pp. 1019–1027 (2016)

[145] Gambichler, T., Regeniter, P., Bechara, F.G., Orlikov, A., Vasa, R., Moussa, G., Stücker, M., Altmeyer, P., Hoffmann, K.: Characterization of benign and malignant melanocytic skin lesions using optical coherence tomography in vivo. Journal of the American Academy of Dermatology 57(4), 629–637 (2007)

[146] Ganiler, O., et al.: A subtraction pipeline for automatic detection of new appearing multiple sclerosis lesions in longitudinal studies. Neuroradiology 56(5), 363–374 (2014)

[147] Gao, C., Liu, X., Peven, M., Unberath, M., Reiter, A.: Learning to See Forces: Surgical Force Prediction with RGB-Point Cloud Temporal Convolutional Networks. In: OR 2.0 Context-Aware Operating Theaters, Computer Assisted Robotic Endoscopy, Clinical Image-Based Procedures, and Skin Image Analysis. pp. 118–127 (2018)

[148] García-Lorenzo, D., Francis, S., Narayanan, S., Arnold, D.L., Collins, D.L.: Review of automatic segmentation methods of multiple sclerosis white matter lesions on conventional magnetic resonance imaging. Medical Image Analysis 17(1), 1–18 (2013)

[149] García-Peraza-Herrera, L.C., Li, W., Gruijthuijsen, C., Devreker, A., Attilakos, G., Deprest, J., Vander Poorten, E., Stoyanov, D., Vercauteren, T., Ourselin, S.: Real-time segmentation of non-rigid surgical tools based on deep learning and tracking. In: International Workshop on Computer-Assisted and Robotic Endoscopy. pp. 84–95. Springer (2016)

[150] Gargesha, M., Shalev, R., Prabhu, D., Tanaka, K., Rollins, A.M., Costa, M., Bezerra, H.G., Wilson, D.L.: Parameter estimation of atherosclerotic tissue optical properties from three-dimensional intravascular optical coherence tomography. Journal of Medical Imaging 2(1), 016001 (2015)

[151] Garvin, M.K., Abràmoff, M.D., Kardon, R., Russell, S.R., Wu, X., Sonka, M.: Intraretinal layer segmentation of macular optical coherence tomography images using optimal 3-D graph search. IEEE Transactions on Medical Imaging 27(10), 1495–1505 (2008)

[152] Gerard, O., Billon, A.C., Rouet, J.M., Jacob, M., Fradkin, M., Allouche, C.: Efficient model-based quantification of left ventricular function in 3-D echocardiography. IEEE Transactions on Medical Imaging 21(9), 1059–1068 (2002)

[153] Gessert, N., Bengs, M., Krüger, J., Opfer, R., Ostwaldt, A.C., Manogaran, P., Schippling, S., Schlaefer, A.: 4D Deep Learning for Multiple-Sclerosis Lesion Activity Segmentation. In: Medical Imaging with Deep Learning (2020)

[154] Gessert, N., Bengs, M., Schlüter, M., Schlaefer, A.: Deep learning with 4D spatio-temporal data representations for OCT-based force estimation. Medical Image Analysis p. 101730 (2020)

[155] Gessert, N., Beringhoff, J., Otte, C., Schlaefer, A.: Force estimation from OCT volumes using 3D CNNs. International Journal of Computer Assisted Radiology and Surgery 13(7), 1073–1082 (2018)

[156] Gessert, N., Gromniak, M., Schlüter, M., Schlaefer, A.: Two-path 3D CNNs for calibration of system parameters for OCT-based motion compensation. In: Medical Imaging 2019: Image-Guided Procedures, Robotic Interventions, and Modeling. vol. 10951, p. 1095108. International Society for Optics and Photonics (2019)

[157] Gessert, N., Heyder, M., Latus, S., Lutz, M., Schlaefer, A.: Plaque classification in coronary arteries from ivoct images using convolutional neural networks and transfer learning. International Journal of Computer Assisted Radiology and Surgery 13(Suppl 1), 99–100 (2018)

[158] Gessert, N., Krüger, J., Opfer, R., Ostwaldt, A.C., Manogaran, P., Schippling, S., Schlaefer, A.: Multiple Sclerosis Lesion Activity Segmentation with Attention-Guided Two-Path CNNs. Computerized Medical Imaging and Graphics (2020)

[159] Gessert, N., Latus, S., Abdelwahed, Y.S., Leistner, D.M., Lutz, M., Schlaefer, A.: Bioresorbable scaffold visualization in IVOCT images using CNNs and weakly

supervised localization. In: Medical Imaging 2019: Image Processing. vol. 10949, p. 109492C. International Society for Optics and Photonics (2019)

[160] Gessert, N., Lutz, M., Heyder, M., Latus, S., Leistner, D.M., Abdelwahed, Y.S., Schlaefer, A.: Automatic plaque detection in IVOCT pullbacks using convolutional neural networks. IEEE Transactions on Medical Imaging 38(2), 426–434 (2018)

[161] Gessert, N., Priegnitz, T., Saathoff, T., Antoni, S.T., Meyer, D., Hamann, M.F., Jünemann, K.P., Otte, C., Schlaefer, A.: Needle tip force estimation using an oct fiber and a fused convgru-cnn architecture. In: International Conference on Medical Image Computing and Computer-assisted Intervention. pp. 222–229. Springer (2018)

[162] Gessert, N., Priegnitz, T., Saathoff, T., Antoni, S.T., Meyer, D., Hamann, M.F., Jünemann, K.P., Otte, C., Schlaefer, A.: Spatio-temporal deep learning models for tip force estimation during needle insertion. International Journal of Computer Assisted Radiology and Surgery 14(9), 1485–1493 (2019)

[163] Gessert, N., Schlaefer, A.: Left ventricle quantification using direct regression with segmentation regularization and ensembles of pretrained 2D and 3D CNNs. In: International Workshop on Statistical Atlases and Computational Models of the Heart. pp. 375–383. Springer (2019)

[164] Gessert, N., Schlüter, M., Schlaefer, A.: A deep learning approach for pose estimation from volumetric OCT data. Medical Image Analysis 46, 162–179 (2018)

[165] Gessert, N., Sentker, T., Madesta, F., Schmitz, R., Kniep, H., Baltruschat, I., Werner, R., Schlaefer, A.: Skin Lesion Classification Using CNNs with Patch-Based Attention and Diagnosis-Guided Loss Weighting. IEEE Transactions on Biomedical Engineering 67(2), 495–503 (2020)

[166] Gessert, N.T., Schlaefer, A.: Efficient neural architecture search on low-dimensional data for OCT image segmentation. In: Medical Imaging with Deep Learning. pp. 1–5 (2019)

[167] Geyer, H., Caracciolo, G., Abe, H., Wilansky, S., Carerj, S., Gentile, F., Nesser, H.J., Khandheria, B., Narula, J., Sengupta, P.P.: Assessment of myocardial mechanics using speckle tracking echocardiography: fundamentals and clinical applications. Journal of the American Society of Echocardiography 23(4), 351–369 (2010)

[168] Ghafoorian, M., Platel, B.: Convolutional neural networks for ms lesion segmentation, method description of diag team. Proceedings of the 2015 Longitudinal Multiple Sclerosis Lesion Segmentation Challenge pp. 1–2 (2015)

[169] Ghanem, A.M., Hamimi, A.H., Matta, J.R., Carass, A., Elgarf, R.M., Gharib, A.M., Abd-Elmoniem, K.Z.: Automatic coronary wall and atherosclerotic plaque

segmentation from 3D coronary CT angiography. Scientific Reports 9(1), 1–13 (2019)

[170] Gharaibeh, Y., Dong, P., Prabhu, D., Kolluru, C., Lee, J., Zimin, V., Mozafari, H., Bizzera, H., Gu, L., Wilson, D.: Deep learning segmentation of coronary calcified plaque from intravascular optical coherence tomography (IVOCT) images with application to finite element modeling of stent deployment. In: Medical Imaging 2019: Image-Guided Procedures, Robotic Interventions, and Modeling. vol. 10951, p. 109511C. International Society for Optics and Photonics (2019)

[171] Gharaibeh, Y., Prabhu, D., Kolluru, C., Lee, J., Zimin, V., Bezerra, H., Wilson, D.: Coronary calcification segmentation in intravascular OCT images using deep learning: application to calcification scoring. Journal of Medical Imaging 6(4), 045002 (2019)

[172] Giger, M.L., Chan, H.P., Boone, J.: Anniversary paper: history and status of CAD and quantitative image analysis: the role of Medical Physics and AAPM. Medical Physics 35(12), 5799–5820 (2008)

[173] Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 580–587 (2014)

[174] Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the thirteenth international conference on artificial intelligence and statistics. pp. 249–256 (2010)

[175] Glorot, X., Bordes, A., Bengio, Y.: Deep Sparse Rectifier Neural Networks. In: Aistats. vol. 15, p. 275 (2011)

[176] Goodfellow, I., Bengio, Y., Courville, A.: Deep learning. MIT press (2016)

[177] Goodfellow, I.J., Bulatov, Y., Ibarz, J., Arnoud, S., Shet, V.: Multi-digit number recognition from street view imagery using deep convolutional neural networks. arXiv preprint arXiv:1312.6082 (2013)

[178] Greminger, M.A., Nelson, B.J.: Modeling elastic objects with neural networks for vision-based force measurement. In: International Conference on Intelligent Robots and Systems. vol. 2, pp. 1278–1283 (2003)

[179] Greminger, M.A., Nelson, B.J.: Vision-based force measurement. IEEE Transactions on Pattern Analysis and Machine Intelligence 26(3), 290–298 (2004)

[180] Guo, X., Tang, D., Molony, D., Yang, C., Samady, H., Zheng, J., Mintz, G.S., Maehara, A., Wang, L., Pei, X., et al.: A machine learning-based method for intracoronary oct segmentation and vulnerable coronary plaque cap thickness quantification. International Journal of Computational Methods 16(03), 1842008 (2019)

[181] Guo, Y., Camino, A., Wang, J., Huang, D., Hwang, T.S., Jia, Y.: MEDnet, a neural network for automated detection of avascular area in OCT angiography. Biomedical Optics Express 9(11), 5147–5158 (2018)

[182] Guo, Y., Hormel, T.T., Xiong, H., Wang, B., Camino, A., Wang, J., Huang, D., Hwang, T.S., Jia, Y.: Development and validation of a deep learning algorithm for distinguishing the nonperfusion area from signal reduction artifacts on oct angiography. Biomedical Optics Express 10(7), 3257–3268 (2019)

[183] Gupta, P.K., Jensen, P.S., de Juan, E.: Surgical forces and tactile perception during retinal microsurgery. In: International Conference on Medical Image Computing and Computer-assisted Intervention. pp. 1218–1225. Springer (1999)

[184] Haidegger, T., Benyó, B., Kovács, L., Benyó, Z.: Force sensing and force control for surgical robots. In: 7th IFAC Symposium on Modeling and Control in Biomedical Systems. pp. 413–418 (2009)

[185] Halder, A., Dey, D., Sadhu, A.K.: Lung Nodule Detection from Feature Engineering to Deep Learning in Thoracic CT Images: a Comprehensive Review. Journal of Digital Imaging pp. 1–23 (2020)

[186] Han, S., Kang, H.K., Jeong, J.Y., Park, M.H., Kim, W., Bang, W.C., Seong, Y.K.: A deep learning framework for supporting the classification of breast lesions in ultrasound images. Physics in Medicine & Biology 62(19), 7714 (2017)

[187] Handels, H.: Medizinische Bildverarbeitung. Springer (2000)

[188] Haouchine, N., Kuang, W., Cotin, S., Yip, M.C.: Vision-based Force Feedback Estimation for Robot-assisted Surgery using Instrument-constrained Biomechanical 3D Maps. IEEE Robotics and Automation Letters (2018)

[189] Haskins, G., Kruger, U., Yan, P.: Deep Learning in Medical Image Registration: A Survey. arXiv preprint arXiv:1903.02026 (2019)

[190] Hatzfeld, C., Wismath, S., Hessinger, M., Werthschützky, R., Schlaefer, A., Kupnik, M.: A miniaturized sensor for needle tip force measurements. Biomedical Engineering 62(1), 109—-115 (2017)

[191] Havaei, M., Davy, A., Warde-Farley, D., Biard, A., Courville, A., Bengio, Y., Pal, C., Jodoin, P.M., Larochelle, H.: Brain tumor segmentation with deep neural networks. Medical Image Analysis 35, 18–31 (2017)

[192] He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1026–1034 (2015)

[193] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 770–778 (2016)

[194] He, S., Zheng, J., Maehara, A., Mintz, G., Tang, D., Anastasio, M., Li, H.: Convolutional neural network based automatic plaque characterization for intracoronary optical coherence tomography images. In: Medical Imaging 2018: Image Processing. vol. 10574, p. 1057432 (2018)

[195] He, Y., Carass, A., Liu, Y., Jedynak, B.M., Solomon, S.D., Saidha, S., Calabresi, P.A., Prince, J.L.: Fully Convolutional Boundary Regression for Retina OCT Segmentation. In: International Conference on Medical Image Computing and Computer-assisted Intervention. pp. 120–128. Springer (2019)

[196] Healsmith, M., Bourke, J., Osborne, J., Graham-Brown, R.: An evaluation of the revised seven-point checklist for the early diagnosis of cutaneous malignant melanoma. British Journal of Dermatology 130(1), 48–50 (1994)

[197] Heinsfeld, A.S., Franco, A.R., Craddock, R.C., Buchweitz, A., Meneguzzi, F.: Identification of autism spectrum disorder using deep learning and the ABIDE dataset. NeuroImage: Clinical 17, 16–23 (2018)

[198] Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R.: Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580 (2012)

[199] Hochreiter, S.: Untersuchungen zu dynamischen neuronalen Netzen. Diploma, Technische Universität München 91(1) (1991)

[200] Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation 9(8), 1735–1780 (1997)

[201] Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural Networks 2(5), 359–366 (1989)

[202] Hosseini-Asl, E., Keynton, R., El-Baz, A.: Alzheimer's disease diagnostics by adaptation of 3D convolutional network. In: IEEE International Conference on Image Processing. pp. 126–130. IEEE (2016)

[203] Hounsfield, G.N.: Computerized transverse axial scanning (tomography): Part 1. Description of system. The British journal of radiology 46(552), 1016–1022 (1973)

[204] Hu, H., Pan, N., Wang, J., Yin, T., Ye, R.: Automatic segmentation of left ventricle from cardiac MRI via deep learning and region constrained dynamic programming. Neurocomputing 347, 139–148 (2019)

[205] Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 7132–7141 (2018)

[206] Hua, K.L., Hsu, C.H., Hidayati, S.C., Cheng, W.H., Chen, Y.J.: Computer-aided classification of lung nodules on computed tomography images via deep learning technique. OncoTargets and Therapy 8 (2015)

[207] Huang, D., Swanson, E.A., Lin, C.P., Schuman, J.S., Stinson, W.G., Chang, W., Hee, M.R., Flotte, T., Gregory, K., Puliafito, C.A., et al.: Optical Coherence Tomography. Science 254(5035), 1178–1181 (1991)

[208] Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4700–4708 (2017)

[209] Huang, Q., Zhang, F., Li, X.: Machine learning in ultrasound computer-aided diagnostic systems: a survey. BioMed Research International (2018)

[210] Ihsani, A., Doyle, S., Michalski, M., Tenenholtz, N., Roth, H.: 4D CNN for Semantic Segmentation of Cardiac Volumetric Sequences. In: International Workshop on Statistical Atlases and Computational Models of the Heart. vol. 12009, p. 72. Springer Nature (2020)

[211] Iidaka, T.: Resting state functional magnetic resonance imaging and neural network classified autism and control. Cortex 63, 55–67 (2015)

[212] Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., Brox, T.: Flownet 2.0: Evolution of optical flow estimation with deep networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2462–2470 (2017)

[213] Ioffe, S.: Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. In: Advances in Neural Information Processing Systems. pp. 1945–1953 (2017)

[214] Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)

[215] Irsch, K., Lee, S., Bose, S.N., Kang, J.U.: Motion-compensated optical coherence tomography using envelope-based surface detection and Kalman-based prediction. In: Advanced Biomedical and Clinical Diagnostic and Surgical Guidance Systems XVI. vol. 10484, p. 104840Q. International Society for Optics and Photonics (2018)

[216] Isgum, I., Prokop, M., Niemeijer, M., Viergever, M.A., Van Ginneken, B.: Automatic coronary calcium scoring in low-dose chest computed tomography. IEEE Transactions on Medical Imaging 31(12), 2322–2334 (2012)

[217] Ishikawa, H., Stein, D.M., Wollstein, G., Beaton, S., Fujimoto, J.G., Schuman, J.S.: Macular segmentation with optical coherence tomography. Investigative Ophthalmology & Visual Science 46(6), 2012–2017 (2005)

[218] Jack, C., Shiung, M., Gunter, J., O'brien, P., Weigand, S., Knopman, D.S., Boeve, B.F., Ivnik, R., Smith, G., Cha, R., et al.: Comparison of different MRI brain atrophy rate measures with clinical disease progression in AD. Neurology 62(4), 591–600 (2004)

[219] Jacobs, R.A.: Increased rates of convergence through learning rate adaptation. Neural Networks 1(4), 295–307 (1988)

[220] Jafari, M.H., Girgis, H., Liao, Z., Behnami, D., Abdi, A., Vaseli, H., Luong, C., Rohling, R., Gin, K., Tsang, T., et al.: A unified framework integrating recurrent fully-convolutional networks and optical flow for segmentation of the left ventricle in echocardiography data. In: Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support, pp. 29–37. Springer (2018)

[221] Jang, I.K., Bouma, B.E., Kang, D.H., Park, S.J., Park, S.W., Seung, K.B., Choi, K.B., Shishkov, M., Schlendorf, K., Pomerantsev, E., et al.: Visualization of coronary atherosclerotic plaques in patients using optical coherence tomography: comparison with intravascular ultrasound. Journal of the American College of Cardiology 39(4), 604–609 (2002)

[222] Jang, Y., Kim, S., Shim, H., Chang, H.J.: Full Quantification of Left Ventricle Using Deep Multitask Network with Combination of 2D and 3D Convolution on 2D+t Cine MRI. In: International Workshop on Statistical Atlases and Computational Models of the Heart. pp. 476–483. Springer (2018)

[223] Jarrett, K., Kavukcuoglu, K., LeCun, Y., et al.: What is the best multi-stage architecture for object recognition? In: IEEE International Conference on Computer Vision. pp. 2146–2153 (2009)

[224] Jerosch-Herold, M., Swingen, C., Seethamraju, R.T.: Myocardial blood flow quantification with MRI by model-independent deconvolution. Medical Physics 29(5), 886–897 (2002)

[225] Jia, Y., Morrison, J.C., Tokayer, J., Tan, O., Lombardi, L., Baumann, B., Lu, C.D., Choi, W., Fujimoto, J.G., Huang, D.: Quantitative OCT angiography of optic nerve head blood flow. Biomedical Optics Express 3(12), 3127–3137 (2012)

[226] Jiang, Z., Huang, Z., Qiu, B., Meng, X., You, Y., Liu, X., Liu, G., Zhou, C., Yang, K., Maier, A., et al.: Comparative study of deep learning models for optical coherence tomography angiography. Biomedical Optics Express 11(3), 1580–1597 (2020)

[227] Jin, A., Yeung, S., Jopling, J., Krause, J., Azagury, D., Milstein, A., Fei-Fei, L.: Tool detection and operative skill assessment in surgical videos using region-based convolutional neural networks. In: IEEE Winter Conference on Applications of Computer Vision. pp. 691–699. IEEE (2018)

[228] Jin, Y., Dou, Q., Chen, H., Yu, L., Qin, J., Fu, C.W., Heng, P.A.: SV-RCNet: workflow recognition from surgical videos using recurrent convolutional network. IEEE Transactions on Medical Imaging 37(5), 1114–1126 (2017)

[229] Jones, T.: The role of positron emission tomography within the spectrum of medical imaging. European journal of nuclear medicine 23(2), 207–211 (1996)

[230] Kak, A.C., Slaney, M., Wang, G.: Principles of computerized tomographic imaging. Medical Physics 29(1), 107–107 (2002)

[231] Kamnitsas, K., Ledig, C., Newcombe, V.F., Simpson, J.P., Kane, A.D., Menon, D.K., Rueckert, D., Glocker, B.: Efficient multi-scale 3D CNN with fully connected CRF for accurate brain lesion segmentation. Medical Image Analysis 36, 61–78 (2017)

[232] Kandasamy, K., Neiswanger, W., Schneider, J., Poczos, B., Xing, E.P.: Neural architecture search with bayesian optimisation and optimal transport. In: Advances in Neural Information Processing Systems. pp. 2016–2025 (2018)

[233] Kaneko, M., Nanayama, N., Tsuji, T.: Vision-based active sensor using a flexible beam. IEEE/ASME Transactions on Mechatronics 6(1), 7–16 (2001)

[234] Karamitsos, T.D., Francis, J.M., Myerson, S., Selvanayagam, J.B., Neubauer, S.: The role of cardiovascular magnetic resonance imaging in heart failure. Journal of the American College of Cardiology 54(15), 1407–1424 (2009)

[235] Karimirad, F., Chauhan, S., Shirinzadeh, B.: Vision-based force measurement using neural networks for biological cell microinjection. Journal of Biomechanics 47(5), 1157–1163 (2014)

[236] Kassahun, Y., Yu, B., Tibebu, A.T., Stoyanov, D., Giannarou, S., Metzen, J.H., Vander Poorten, E.: Surgical robotics beyond enhanced dexterity instrumentation: a survey of machine learning techniques and their role in intelligent and autonomous surgical actions. International Journal of Computer Assisted Radiology and Surgery 11(4), 553–568 (2016)

[237] Kataoka, H., Washio, T., Chinzei, K., Mizuhara, K., Simone, C., Okamura, A.M.: Measurement of the tip and friction force acting on a needle during penetration. In: International Conference on Medical Image Computing and Computer-assisted Intervention. pp. 216–223 (2002)

[238] Kaus, M.R., Von Berg, J., Weese, J., Niessen, W., Pekar, V.: Automated segmentation of the left ventricle in cardiac MRI. Medical Image Analysis 8(3), 245–254 (2004)

[239] Kaus, M.R., Warfield, S.K., Nabavi, A., Black, P.M., Jolesz, F.A., Kikinis, R.: Automated segmentation of MR images of brain tumors. Radiology 218(2), 586–591 (2001)

[240] Kawahara, J., Brown, C.J., Miller, S.P., Booth, B.G., Chau, V., Grunau, R.E., Zwicker, J.G., Hamarneh, G.: BrainNetCNN: Convolutional neural networks for brain networks; towards predicting neurodevelopment. NeuroImage 146, 1038–1049 (2017)

[241] Kehl, W., Milletari, F., Tombari, F., Ilic, S., Navab, N.: Deep learning of local RGB-D patches for 3D object detection and 6D pose estimation. In: Proceedings of the European Conference on Computer Vision. pp. 205–220. Springer (2016)

[242] Khazaee, A., Ebrahimzadeh, A., Babajani-Feremi, A.: Application of advanced machine learning methods on resting-state fMRI network for identification of mild cognitive impairment and Alzheimer's disease. Brain Imaging and Behavior 10(3), 799–817 (2016)

[243] Khazaee, A., Ebrahimzadeh, A., Babajani-Feremi, A., Initiative, A.D.N., et al.: Classification of patients with MCI and AD from healthy controls using directed graph measures of resting-state fMRI. Behavioural Brain Research 322, 339–350 (2017)

[244] Khened, M., Kollerathu, V.A., Krishnamurthi, G.: Fully convolutional multi-scale residual DenseNets for cardiac segmentation and automated cardiac diagnosis using ensemble of classifiers. Medical Image Analysis 51, 21–45 (2019)

[245] Kidwell, C.S., Chalela, J.A., Saver, J.L., Starkman, S., Hill, M.D., Demchuk, A.M., Butman, J.A., Patronas, N., Alger, J.R., Latour, L.L., et al.: Comparison of MRI and CT for detection of acute intracerebral hemorrhage. Journal of the American Medical Association 292(15), 1823–1830 (2004)

[246] Kim, D., Cho, H., Shin, H., Lim, S.C., Hwang, W.: An efficient three-dimensional convolutional neural network for inferring physical interaction force from video. Sensors 19(16), 3579 (2019)

[247] Kim, J., Janabi-Sharifi, F., Kim, J.: A haptic interaction method using visual information and physically based modeling. IEEE/ASME Transactions on Mechatronics 15(4), 636–645 (2010)

[248] Kim, K.H., Choi, S.H., Park, S.H.: Improving arterial spin labeling by using deep learning. Radiology 287(2), 658–666 (2018)

[249] Kim, W., Seung, S., Choi, H., Park, S., Ko, S.Y., Park, J.O.: Image-based force estimation of deformable tissue using depth map for single-port surgical robot. In: International Conference on Control, Automation and Systems. pp. 1716–1719. IEEE (2012)

[250] Kingma, D., Ba, J.: Adam: A method for stochastic optimization. In: International Conference on Learning Representations (2015)

[251] Kleesiek, J., Urban, G., Hubert, A., Schwarz, D., Maier-Hein, K., Bendszus, M., Biller, A.: Deep MRI brain extraction: A 3D convolutional neural network for skull stripping. NeuroImage 129, 460–469 (2016)

[252] Klein, T., Wieser, W., Reznicek, L., Neubauer, A., Kampik, A., Huber, R.: Multi-MHz retinal OCT. Biomedical Optics Express 4(10), 1890–1908 (2013)

[253] Ko, S.M., Kim, T.H., Chun, E.J., Kim, J.Y., Hwang, S.H.: Assessment of Left Ventricular Myocardial Diseases with Cardiac Computed Tomography. Korean Journal of Radiology 20(3), 333–351 (2019)

[254] Kohavi, R., Wolpert, D.H., et al.: Bias plus variance decomposition for zero-one loss functions. In: ICML. vol. 96, pp. 275–83 (1996)

[255] Kolluru, C., Prabhu, D., Gharaibeh, Y., Bezerra, H., Guagliumi, G., Wilson, D.: Deep neural networks for A-line-based plaque classification in coronary intravascular optical coherence tomography images. Journal of Medical Imaging 5(4), 044504 (2018)

[256] Koozekanani, D., Boyer, K., Roberts, C.: Retinal thickness measurements from optical coherence tomography using a Markov boundary model. IEEE Transactions on Medical Imaging 20(9), 900–916 (2001)

[257] Koretsky, A.P.: Early development of arterial spin labeling to measure regional brain blood flow by MRI. NeuroImage 62(2), 602–607 (2012)

[258] Korolev, S., Safiullin, A., Belyaev, M., Dodonova, Y.: Residual and plain convolutional neural networks for 3D brain MRI classification. In: IEEE International Symposium on Biomedical Imaging. pp. 835–838. IEEE (2017)

[259] Kozegar, E., Soryani, M., Behnam, H., Salamati, M., Tan, T.: Computer aided detection in automated 3-D breast ultrasound images: a survey. Artificial Intelligence Review pp. 1–23 (2019)

[260] Kral, F., Puschban, E.J., Riechelmann, H., Freysinger, W.: Comparison of optical and electromagnetic tracking for navigated lateral skull base surgery. The International Journal of Medical Robotics and Computer Assisted Surgery 9(2), 247–252 (2013)

[261] Kraus, M.F., Potsaid, B., Mayer, M.A., Bock, R., Baumann, B., Liu, J.J., Hornegger, J., Fujimoto, J.G.: Motion correction in optical coherence tomography volumes on a per A-scan basis using orthogonal scan patterns. Biomedical Optics Express 3(6), 1182–1199 (2012)

[262] Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems. pp. 1097–1105 (2012)

[263] Kroh, M., Chalikonda, S.: Essentials of robotic surgery. Springer (2015)

[264] Krull, A., Brachmann, E., Michel, F., Ying Yang, M., Gumhold, S., Rother, C.: Learning Analysis-by-Synthesis for 6D Pose Estimation in RGB-D Images. In: The IEEE International Conference on Computer Vision (2015)

[265] Kumar, S., Shrikanth, V., Amrutur, B., Asokan, S., Bobji, M.S.: Detecting stages of needle penetration into tissues through force estimation at needle tip using fiber Bragg grating sensors. Journal of Biomedical Optics 21(12), 127009 (2016)

[266] Laina, I., Rieke, N., Rupprecht, C., Vizcaíno, J.P., Eslami, A., Tombari, F., Navab, N.: Concurrent segmentation and localization for tracking of surgical instruments. In: International Conference on Medical Image Computing and Computer-assisted Intervention. pp. 664–672. Springer (2017)

[267] Lambin, P., Rios-Velazquez, E., Leijenaar, R., Carvalho, S., Van Stiphout, R.G., Granton, P., Zegers, C.M., Gillies, R., Boellard, R., Dekker, A., et al.: Radiomics: extracting more information from medical images using advanced feature analysis. European Journal of Cancer 48(4), 441–446 (2012)

[268] Lang, A., Carass, A., Hauser, M., Sotirchos, E.S., Calabresi, P.A., Ying, H.S., Prince, J.L.: Retinal layer segmentation of macular OCT images using boundary classification. Biomedical Optics Express 4(7), 1133–1152 (2013)

[269] Lang, R.M., Badano, L.P., Tsang, W., Adams, D.H., Agricola, E., Buck, T., Faletra, F.F., Franke, A., Hung, J., de Isla, L.P., et al.: EAE/ASE recommendations for image acquisition and display using three-dimensional echocardiography. European Heart Journal–Cardiovascular Imaging 13(1), 1–46 (2012)

[270] Lankenau, E., Klinger, D., Winter, C., Malik, A., Müller, H.H., Oelckers, S., Pau, H.W., Just, T., Hüttmann, G.: Combining optical coherence tomography (OCT) with an operating microscope. In: Advances in Medical Engineering, pp. 343–348. Springer (2007)

[271] Latus, S., Griese, F., Gräser, M., Möddel, M., Schlüter, M., Otte, C., Gessert, N., Saathoff, T., Knopp, T., Schlaefer, A.: Towards bimodal intravascular OCT MPI volumetric imaging. In: Medical Imaging 2018: Physics of Medical Imaging. vol. 10573, p. 105732E. International Society for Optics and Photonics (2018)

[272] Latus, S., Griese, F., Schlüter, M., Otte, C., Möddel, M., Graeser, M., Saathoff, T., Knopp, T., Schlaefer, A.: Bimodal intravascular volumetric imaging combining OCT and MPI. Medical Physics 46(3), 1371–1383 (2019)

[273] Latus, S., Neidhardt, M., Lutz, M., Gessert, N., Frey, N., Schlaefer, A.: Quantitative Analysis of 3D Artery Volume Reconstructions Using Biplane Angiography and Intravascular OCT Imaging. In: International Conference of the IEEE Engineering in Medicine and Biology Society. pp. 6004–6007. IEEE (2019)

[274] Lauermann, J., Treder, M., Alnawaiseh, M., Clemens, C., Eter, N., Alten, F.: Automated OCT angiography image quality assessment using a deep learning algorithm. Graefe's Archive for Clinical and Experimental Ophthalmology 257(8), 1641–1648 (2019)

[275] Laves, M.H., Ihler, S., Kahrs, L.A., Ortmaier, T.: Deep-learning-based 2.5 D flow field estimation for maximum intensity projections of 4D optical coherence tomography. In: Medical Imaging 2019: Image-Guided Procedures, Robotic Interventions, and Modeling. vol. 10951, p. 109510R. International Society for Optics and Photonics (2019)

[276] Laves, M.H., Schoob, A., Kahrs, L.A., Pfeiffer, T., Huber, R., Ortmaier, T.: Feature tracking for automated volume of interest stabilization on 4D-OCT images. In: Medical Imaging 2017: Image-Guided Procedures, Robotic Interventions, and Modeling. vol. 10135, p. 101350W. International Society for Optics and Photonics (2017)

[277] Leal-Taixé, L., Canton-Ferrer, C., Schindler, K.: Learning by tracking: Siamese CNN for robust target association. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 33–40 (2016)

[278] LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature 521(7553), 436–444 (2015)

[279] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al.: Gradient-based learning applied to document recognition. Proceedings of the IEEE 86(11), 2278–2324 (1998)

[280] Lee, C.S., Baughman, D.M., Lee, A.Y.: Deep learning is effective for classifying normal versus age-related macular degeneration OCT images. Ophthalmology Retina 1(4), 322–327 (2017)

[281] Lee, H.Y., Codella, N.C., Cham, M.D., Weinsaft, J.W., Wang, Y.: Automatic left ventricle segmentation using iterative thresholding and an active contour model with adaptation on short-axis cardiac MRI. IEEE Transactions on Biomedical Engineering 57(4), 905–913 (2009)

[282] Lee, J., Prabhu, D., Kolluru, C., Gharaibeh, Y., Zimin, V.N., Bezerra, H.G., Wilson, D.L.: Automated plaque characterization using deep learning on coronary intravascular optical coherence tomographic images. Biomedical Optics Express 10(12), 6497–6515 (2019)

[283] Lee, J., Prabhu, D., Kolluru, C., Gharaibeh, Y., Zimin, V.N., Dallan, L.A., Bezerra, H.G., Wilson, D.L.: Fully automated plaque characterization in intravascular OCT images using hybrid convolutional and lumen morphology features. Scientific Reports 10(1), 1–13 (2020)

[284] van de Leemput, S.C., Prokop, M., van Ginneken, B., Manniesing, R.: Stacked Bidirectional Convolutional LSTMs for Deriving 3D Non-contrast CT from Spatiotemporal 4D CT. IEEE Transactions on Medical Imaging (2019)

[285] Leitgeb, R., Hitzenberger, C.K., Fercher, A.F.: Performance of fourier domain vs. time domain optical coherence tomography. Optics Express 11(8), 889–894 (2003)

[286] Lemaître, G., Rastgoo, M., Massich, J., Cheung, C.Y., Wong, T.Y., Lamoureux, E., Milea, D., Mériaudeau, F., Sidibé, D.: Classification of SD-OCT volumes using local binary patterns: experimental validation for DME detection. Journal of Ophthalmology 2016 (2016)

[287] Lesjak, Ž., et al.: Validation of white-matter lesion change detection methods on a novel publicly available MRI image database. Neuroinformatics 14(4), 403–420 (2016)

[288] Lesjak, Ž., et al.: A novel public MR image dataset of multiple sclerosis patients with lesion segmentations based on multi-rater consensus. Neuroinformatics 16(1), 51–63 (2018)

[289] Lessmann, N., van Ginneken, B., Zreik, M., de Jong, P.A., de Vos, B.D., Viergever, M.A., Išgum, I.: Automatic calcium scoring in low-dose chest CT using deep neural networks with dilated convolutions. IEEE Transactions on Medical Imaging 37(2), 615–625 (2017)

[290] Lessmann, N., Išgum, I., Setio, A.A., de Vos, B.D., Ciompi, F., de Jong, P.A., Oudkerk, M., Willem, P.T.M., Viergever, M.A., van Ginneken, B.: Deep convolutional neural networks for automatic coronary calcium scoring in a screening study with low-dose chest CT. In: Medical Imaging 2016: Computer-Aided Diagnosis. vol. 9785, p. 978511. International Society for Optics and Photonics (2016)

[291] Li, G., Citrin, D., Camphausen, K., Mueller, B., Burman, C., Mychalczak, B., Miller, R.W., Song, Y.: Advances in 4D medical imaging and 4D radiation therapy. Technology in Cancer Research & Treatment 7(1), 67–81 (2008)

[292] Li, J., Hu, Z.: Left Ventricle Full Quantification using Deep Layer Aggregation based Multitask Relationship Learning. In: International Workshop on Statistical Atlases and Computational Models of the Heart. pp. 381–388. Springer (2018)

[293] Li, X., Dvornek, N.C., Papademetris, X., Zhuang, J., Staib, L.H., Ventola, P., Duncan, J.S.: 2-channel convolutional 3D deep neural network (2CC3D) for fMRI analysis: ASD classification and feature learning. In: IEEE International Symposium on Biomedical Imaging. pp. 1252–1255. IEEE (2018)

[294] Li, X., Chudoba, C., Ko, T., Pitris, C., Fujimoto, J.G.: Imaging needle for optical coherence tomography. Optics Letters 25(20), 1520–1522 (2000)

[295] Li, Y., Wang, G., Ji, X., Xiang, Y., Fox, D.: Deepim: Deep iterative matching for 6d pose estimation. In: Proceedings of the European Conference on Computer Vision. pp. 683–698 (2018)

[296] Liang, X., Connelly, A., Calamante, F.: Voxel-wise functional connectomics using arterial spin labeling functional magnetic resonance imaging: the role of denoising. Brain Connectivity 5(9), 543–553 (2015)

[297] Liang, Z.P., Lauterbur, P.C.: Principles of magnetic resonance imaging: a signal processing perspective. SPIE Optical Engineering Press (2000)

[298] Lim, G., Cheng, Y., Hsu, W., Lee, M.L.: Integrated optic disc and cup segmentation with deep learning. In: IEEE International Conference on Tools with Artificial Intelligence. pp. 162–169. IEEE (2015)

[299] Lin, M., Chen, Q., Yan, S.: Network in network. In: International Conference on Learning Representations (2014)

[300] Lin, X., Cowan, B.R., Young, A.A.: Automated detection of left ventricle in 4D MR images: experience from a large study. In: International Conference on Medical Image Computing and Computer-assisted Intervention. pp. 728–735. Springer (2006)

[301] Litjens, G., Kooi, T., Bejnordi, B.E., Setio, A.A.A., Ciompi, F., Ghafoorian, M., Van Der Laak, J.A., Van Ginneken, B., Sánchez, C.I.: A survey on deep learning in medical image analysis. Medical Image Analysis 42, 60–88 (2017)

[302] Liu, C., Chen, L.C., Schroff, F., Adam, H., Hua, W., Yuille, A.L., Fei-Fei, L.: Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 82–92 (2019)

[303] Liu, C., Zoph, B., Neumann, M., Shlens, J., Hua, W., Li, L.J., Fei-Fei, L., Yuille, A., Huang, J., Murphy, K.: Progressive neural architecture search. In: Proceedings of the European Conference on Computer Vision. pp. 19–34 (2018)

[304] Liu, L., Jiang, H., He, P., Chen, W., Liu, X., Gao, J., Han, J.: On the variance of the adaptive learning rate and beyond. arXiv preprint arXiv:1908.03265 (2019)

[305] Liu, R., Zhang, Y., Zheng, Y., Liu, Y., Zhao, Y., Yi, L.: Automated Detection of Vulnerable Plaque for Intravascular Optical Coherence Tomography Images. Cardiovascular Engineering and Technology 10(4), 590–603 (2019)

[306] Liu, S., Wang, Y., Yang, X., Lei, B., Liu, L., Li, S.X., Ni, D., Wang, T.: Deep learning in medical ultrasound analysis: a review. Engineering (2019)

[307] Liu, S., Shamonin, D.P., Zahnd, G., van der Steen, A., van Walsum, T., van Soest, G.: Healthy Vessel Wall Detection Using U-Net in Optical Coherence Tomography. In: Machine Learning and Medical Engineering for Cardiovascular Health and Intravascular Imaging and Computer Assisted Stenting, pp. 184–192. Springer (2019)

[308] Liu, S., Sotomi, Y., Eggermont, J., Nakazawa, G., Torii, S., Ijichi, T., Onuma, Y., Serruys, P.W., Lelieveldt, B.P., Dijkstra, J.: Tissue characterization with depth-resolved attenuation coefficient and backscatter term in intravascular optical coherence tomography images. Journal of Biomedical Optics 22(9), 096004 (2017)

[309] Liu, X., Huang, Z., Wang, Z., Wen, C., Jiang, Z., Yu, Z., Liu, J., Liu, G., Huang, X., Maier, A., et al.: A deep learning based pipeline for optical coherence tomography angiography. Journal of Biophotonics 12(10), e201900008 (2019)

[310] Liu, X., Wang, W., Lansdorp, B.M., Sun, Y.: Vision-based cellular force measurement using an elastic microfabricated device. In: 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 1378–1383. IEEE (2006)

[311] Lo, S.C., Lou, S.L., Lin, J.S., Freedman, M.T., Chien, M.V., Mun, S.K.: Artificial convolution neural network techniques and applications for lung nodule detection. IEEE Transactions on Medical Imaging 14(4), 711–718 (1995)

[312] Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3431–3440 (2015)

[313] Looney, P., Stevenson, G.N., Nicolaides, K.H., Plasencia, W., Molloholli, M., Natsis, S., Collins, S.L.: Automatic 3D ultrasound segmentation of the first trimester placenta using deep learning. In: IEEE International Symposium on Biomedical Imaging. pp. 279–282. IEEE (2017)

[314] Lowe, D.G.: Object recognition from local scale-invariant features. In: Proceedings of the seventh IEEE international conference on computer vision. vol. 2, pp. 1150–1157. Ieee (1999)

[315] Lu, H., Gargesha, M., Wang, Z., Chamie, D., Attizzani, G.F., Kanaya, T., Ray, S., Costa, M.A., Rollins, A.M., Bezerra, H.G., et al.: Automatic stent detection in intravascular OCT images using bagged decision trees. Biomedical Optics Express 3(11), 2809–2824 (2012)

[316] Lu, S., Cheung, C.Y.l., Liu, J., Lim, J.H., Leung, C.K.s., Wong, T.Y.: Automated layer segmentation of optical coherence tomography images. IEEE Transactions on Biomedical Engineering 57(10), 2605–2608 (2010)

[317] Lucas, B.D., Kanade, T., et al.: An iterative image registration technique with an application to stereo vision. In: Proceedings DARPA Image Understanding Workshop. Vancouver, British Columbia (1981)

[318] Luo, G., Dong, S., Wang, K., Zhang, D., Gao, Y., Chen, X., Zhang, H., Li, S.: A Deep Reinforcement Learning Framework for Frame-by-Frame Plaque Tracking on Intravascular Optical Coherence Tomography Image. In: International Conference on Medical Image Computing and Computer-assisted Intervention. pp. 12–20. Springer (2019)

[319] Macedo, M.M.G.d., Takimura, C.K., Lemos, P.A., Gutierrez, M.A.: A robust fully automatic lumen segmentation method for in vivo intracoronary optical coherence tomography. Research on Biomedical Engineering 32(1), 35–43 (2016)

[320] Mao, Z., Su, Y., Xu, G., Wang, X., Huang, Y., Yue, W., Sun, L., Xiong, N.: Spatio-temporal deep learning method for ADHD fMRI classification. Information Sciences 499, 1–11 (2019)

[321] Marban, A., Srinivasan, V., Samek, W., Fernández, J., Casals, A.: A recurrent convolutional neural network approach for sensorless force estimation in robotic surgery. Biomedical Signal Processing and Control 50, 134–150 (2019)

[322] Markl, M., Frydrychowicz, A., Kozerke, S., Hope, M., Wieben, O.: 4D flow MRI. Journal of Magnetic Resonance Imaging 36(5), 1015–1036 (2012)

[323] Marr, D., Hildreth, E.: Theory of edge detection. Proceedings of the Royal Society of London. Series B. Biological Sciences 207(1167), 187–217 (1980)

[324] Marsland, S.: Machine learning: an algorithmic perspective. CRC Press (2015)

[325] Masood, S., Fang, R., Li, P., Li, H., Sheng, B., Mathavan, A., Wang, X., Yang, P., Wu, Q., Qin, J., et al.: Automatic choroid layer segmentation from optical coherence tomography images using deep learning. Scientific Reports 9(1), 1–18 (2019)

[326] Matovinovic, I.Z., Loncaric, S., Lo, J., Heisler, M., Sarunic, M.: Transfer Learning with U-Net type model for Automatic Segmentation of Three Retinal Layers In Optical Coherence Tomography Images. In: IEEE International Symposium on Image and Signal Processing and Analysis. pp. 49–53. IEEE (2019)

[327] Maturana, D., Scherer, S.: Voxnet: A 3d convolutional neural network for real-time object recognition. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 922–928. IEEE (2015)

[328] Mayer, A., Greenspan, H.: An adaptive mean-shift framework for MRI brain segmentation. IEEE Transactions on Medical Imaging 28(8), 1238–1250 (2009)

[329] McCreery, G.L., Trejos, A.L., Naish, M.D., Patel, R.V., Malthaner, R.A.: Feasibility of locating tumours in lung via kinaesthetic feedback. The International Journal of Medical Robotics and Computer Assisted Surgery 4(1), 58–68 (2008)

[330] McFarland, H.F., et al.: Using gadolinium-enhanced magnetic resonance imaging lesions to monitor disease activity in multiple sclerosis. Annals of Neurology 32(6), 758–766 (1992)

[331] Mehanna, E., Bezerra, H.G., Prabhu, D., Brandt, E., Chamié, D., Yamamoto, H., Attizzani, G.F., Tahara, S., Van Ditzhuijzen, N., Fujino, Y., et al.: Volumetric characterization of human coronary calcification by frequency-domain optical coherence tomography. Circulation Journal 77(9), 2334–2340 (2013)

[332] Meier-Schroers, M., Homsi, R., Skowasch, D., Buermann, J., Zipfel, M., Schild, H.H., Thomas, D.: Lung cancer screening with MRI: results of the first screening round. Journal of Cancer Research and Clinical Oncology 144(1), 117–125 (2018)

[333] Meli, L., Pacchierotti, C., Prattichizzo, D.: Experimental evaluation of magnified haptic feedback for robot-assisted needle insertion and palpation. The International Journal of Medical Robotics and Computer Assisted Surgery 13(4) (2017)

[334] Mendes-Moreira, J., Soares, C., Jorge, A.M., Sousa, J.F.D.: Ensemble approaches for regression: A survey. ACM Computing Surveys 45(1), 10 (2012)

[335] Mendizabal, A., Sznitman, R., Cotin, S.: Force classification during robotic interventions through simulation-trained neural networks. International Journal of Computer Assisted Radiology and Surgery 14(9), 1601–1610 (2019)

[336] Messay, T., Hardie, R.C., Rogers, S.K.: A new computationally efficient CAD system for pulmonary nodule detection in CT imagery. Medical Image Analysis 14(3), 390–406 (2010)

[337] Michel, R.G., Kinasewitz, G.T., Fung, K.M., Keddissi, J.I.: Optical coherence tomography as an adjunct to flexible bronchoscopy in the diagnosis of lung cancer: a pilot study. Chest 138(4), 984–988 (2010)

[338] Mihalef, V., Ionasec, R., Wang, Y., Zheng, Y., Georgescu, B., Comaniciu, D.: Patient-specific modeling of left heart anatomy, dynamics and hemodynamics from high resolution 4D CT. In: IEEE International Symposium on Biomedical Imaging. pp. 504–507. IEEE (2010)

[339] Milletari, F., Navab, N., Ahmadi, S.A.: V-net: Fully convolutional neural networks for volumetric medical image segmentation. In: International Conference on 3D Vision. pp. 565–571. IEEE (2016)

[340] Mintz, G.S.: Intravascular imaging of coronary calcification and its clinical implications. JACC: Cardiovascular Imaging 8(4), 461–471 (2015)

[341] Miotto, R., Wang, F., Wang, S., Jiang, X., Dudley, J.T.: Deep learning for healthcare: review, opportunities and challenges. Briefings in bioinformatics 19(6), 1236–1246 (2018)

[342] Mishra, A., Wong, A., Bizheva, K., Clausi, D.A.: Intra-retinal layer segmentation in optical coherence tomography images. Optics Express 17(26), 23719–23728 (2009)

[343] Mitchell, T.M.: Machine learning. 1997. Burr Ridge, IL: McGraw Hill 45(37), 870–877 (1997)

[344] Miyagawa, M., Costa, M.G., Gutierrez, M.A., Costa, J., Costa Filho, C.F.: Lumen segmentation in optical coherence tomography images using convolutional neural network. In: International Conference of the IEEE Engineering in Medicine and Biology Society. pp. 600–603. IEEE (2018)

[345] Mo, Y., Liu, F., McIlwraith, D., Yang, G., Zhang, J., He, T., Guo, Y.: The deep Poincaré map: A novel approach for left ventricle segmentation. In: International Conference on Medical Image Computing and Computer-assisted Intervention. pp. 561–568. Springer (2018)

[346] Moeskops, P., Viergever, M.A., Mendrik, A.M., De Vries, L.S., Benders, M.J., Išgum, I.: Automatic segmentation of MR brain images with a convolutional neural network. IEEE Transactions on Medical Imaging 35(5), 1252–1261 (2016)

[347] Moraal, B., Wattjes, M.P., Geurts, J.J., Knol, D.L., van Schijndel, R.A., Pouwels, P.J., Vrenken, H., Barkhof, F.: Improved detection of active multiple sclerosis lesions: 3D subtraction imaging. Radiology 255(1), 154–163 (2010)

[348] Moraes, M.C., Cardenas, D.A.C., Furuie, S.S.: Automatic lumen segmentation in IVOCT images using binary morphological reconstruction. Biomedical Engineering Online 12(1), 78 (2013)

[349] Mortazi, A., Burt, J., Bagci, U.: Multi-planar deep segmentation networks for cardiac substructures from MRI and CT. In: International Workshop on Statistical Atlases and Computational Models of the Heart. pp. 199–206. Springer (2017)

[350] Moult, E., Choi, W., Waheed, N.K., Adhi, M., Lee, B., Lu, C.D., Jayaraman, V., Potsaid, B., Rosenfeld, P.J., Duker, J.S., et al.: Ultrahigh-speed swept-source OCT angiography in exudative AMD. Ophthalmic Surgery, Lasers and Imaging Retina 45(6), 496–505 (2014)

[351] Mozaffari, A., Behzadipour, S., Kohani, M.: Identifying the tool-tissue force in robotic laparoscopic surgery using neuro-evolutionary fuzzy systems and a synchronous self-learning hyper level supervisor. Applied Soft Computing 14, 12–30 (2014)

[352] Murphy, K., van Ginneken, B., Schilham, A.M., De Hoop, B., Gietema, H., Prokop, M.: A large-scale evaluation of automatic pulmonary nodule detection in chest CT using local image features and k-nearest-neighbour classification. Medical Image Analysis 13(5), 757–770 (2009)

[353] Naeini, F.B., Alali, A., Al-Husari, R., Rigi, A., AlSharman, M.K., Makris, D., Zweiri, Y.: A novel dynamic-vision-based approach for tactile sensing applications. IEEE Transactions on Instrumentation and Measurement (2019)

[354] Nair, T., Precup, D., Arnold, D.L., Arbel, T.: Exploring uncertainty measures in deep networks for multiple sclerosis lesion detection and segmentation. Medical Image Analysis 59, 101557 (2020)

[355] Nesterov, Y.: A method of solving a convex programming problem with convergence rate O(1k2). Soviet Nathematics Doklady 27(2), 372–376 (1983)

[356] Ng, A.Y.: Feature selection, L 1 vs. L 2 regularization, and rotational invariance. In: Proceedings of the International Conference on Machine Learning. p. 78 (2004)

[357] Ngo, L., Cha, J., Han, J.H.: Deep Neural Network Regression for Automated Retinal Layer Segmentation in Optical Coherence Tomography Images. IEEE Transactions on Image Processing 29, 303–312 (2019)

[358] Ngo, T.A., Lu, Z., Carneiro, G.: Combining deep learning and level set for the automated segmentation of the left ventricle of the heart from cardiac cine magnetic resonance. Medical Image Analysis 35, 159–171 (2017)

[359] Nie, D., Zhang, H., Adeli, E., Liu, L., Shen, D.: 3D deep learning for multi-modal imaging-guided survival time prediction of brain tumor patients. In: International Conference on Medical Image Computing and Computer-assisted Intervention. pp. 212–220. Springer (2016)

[360] Noohi, E., Parastegari, S., Žefran, M.: Using monocular images to estimate interaction forces during minimally invasive surgery. In: International Conference on Intelligent Robots and Systems. pp. 4297–4302 (2014)

[361] Okamura, A.M.: Haptic feedback in robot-assisted minimally invasive surgery. Current Opinion in Urology 19(1), 102 (2009)

[362] Okamura, A.M., Simone, C., O'leary, M.D.: Force modeling for needle insertion into soft tissue. IEEE Transactions on Biomedical Engineering 51(10), 1707–1716 (2004)

[363] Oktay, O., Ferrante, E., Kamnitsas, K., Heinrich, M., Bai, W., Caballero, J., Cook, S.A., De Marvao, A., Dawes, T., O'Regan, D.P., et al.: Anatomically constrained neural networks (ACNNs): application to cardiac image enhancement and segmentation. IEEE Transactions on Medical Imaging 37(2), 384–395 (2017)

[364] Oktay, O., Schlemper, J., Folgoc, L.L., Lee, M., Heinrich, M., Misawa, K., Mori, K., McDonagh, S., Hammerla, N.Y., Kainz, B., et al.: Attention u-net: Learning where to look for the pancreas. arXiv preprint arXiv:1804.03999 (2018)

[365] Oliveira, D.A., Macedo, M.M., Nicz, P., Campos, C., Lemos, P., Gutierrez, M.A.: Coronary calcification identification in optical coherence tomography using convolutional neural networks. In: Medical Imaging 2018: Biomedical Applications in Molecular, Structural, and Functional Imaging. vol. 10578, p. 105781Y (2018)

[366] Orlando, J.I., Seeböck, P., Bogunović, H., Klimscha, S., Grechenig, C., Waldstein, S., Gerendas, B.S., Schmidt-Erfurth, U.: U2-net: A bayesian u-net model with epistemic uncertainty feedback for photoreceptor layer segmentation in pathological oct scans. In: IEEE International Symposium on Biomedical Imaging. pp. 1441–1445. IEEE (2019)

[367] Otte, C., Beringhoff, J., Latus, S., Antoni, S.T., Rajput, O., Schlaefer, A.: Towards force sensing based on instrument-tissue interaction. In: IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems. pp. 180–185. IEEE (2016)

[368] Otte, S., Otte, C., Schlaefer, A., Wittig, L., Hüttmann, G., Drömann, D., Zell, A.: OCT A-Scan based lung tumor tissue classification with Bidirectional Long Short Term Memory networks. In: IEEE International Workshop on Machine Learning for Signal Processing. pp. 1–6. IEEE (2013)

[369] Pacchierotti, C., Meli, L., Chinello, F., Malvezzi, M., Prattichizzo, D.: Cutaneous haptic feedback to ensure the stability of robotic teleoperation systems. The International Journal of Robotics Research 34(14), 1773–1787 (2015)

[370] Paragios, N.: A level set approach for shape-driven segmentation and tracking of the left ventricle. IEEE Transactions on Medical Imaging 22(6), 773–776 (2003)

[371] Park, B.y., Kim, M., Seo, J., Lee, J.m., Park, H.: Connectivity analysis and feature classification in attention deficit hyperactivity disorder sub-types: a task functional magnetic resonance imaging study. Brain Topography 29(3), 429–439 (2016)

[372] Patti, F., et al.: Lesion load may predict long-term cognitive dysfunction in multiple sclerosis patients. PloS one 10(3), e0120754 (2015)

[373] Pelc, N.J., Bernstein, M.A., Shimakawa, A., Glover, G.H.: Encoding strategies for three-direction phase-contrast MR imaging of flow. Journal of Magnetic Resonance Imaging 1(4), 405–413 (1991)

[374] Peng, S., Chen, W., Sun, J., Liu, B.: Multi-Scale 3D U-Nets: An approach to automatic segmentation of brain tumor. International Journal of Imaging Systems and Technology (2019)

[375] Pereira, S., Pinto, A., Alves, V., Silva, C.A.: Brain tumor segmentation using convolutional neural networks in MRI images. IEEE Transactions on Medical Imaging 35(5), 1240–1251 (2016)

[376] Pham, C.H., Ducournau, A., Fablet, R., Rousseau, F.: Brain MRI super-resolution using deep 3D convolutional networks. In: IEEE International Symposium on Biomedical Imaging. pp. 197–200. IEEE (2017)

[377] Pham, H., Guan, M.Y., Zoph, B., Le, Q.V., Dean, J.: Efficient Neural Architecture Search via Parameter Sharing. In: International Conference on Machine Learning (2018)

[378] Pham, V., Bluche, T., Kermorvant, C., Louradour, J.: Dropout improves recurrent neural networks for handwriting recognition. In: 2014 14th International Conference on Frontiers in Handwriting Recognition. pp. 285–290. IEEE (2014)

[379] Philip, M.E., Sowmya, A., Avnet, H., Ferreira, A., Stevenson, G., Welsh, A.: Convolutional Neural Networks for Automated Fetal Cardiac Assessment using 4D B-Mode Ultrasound. In: IEEE International Symposium on Biomedical Imaging. pp. 824–828. IEEE (2019)

[380] Pigou, L., Van Den Oord, A., Dieleman, S., Van Herreweghe, M., Dambre, J.: Beyond temporal pooling: Recurrence and temporal convolutions for gesture recognition in video. International Journal of Computer Vision 126(2-4), 430–439 (2018)

[381] Pinto, A., Pereira, S., Meier, R., Alves, V., Wiest, R., Silva, C.A., Reyes, M.: Enhancing clinical MRI perfusion maps with data-driven maps of complementary nature for lesion outcome prediction. In: International Conference on Medical Image Computing and Computer-assisted Intervention. pp. 107–115. Springer (2018)

[382] Plitt, M., Barnes, K.A., Martin, A.: Functional connectivity classification of autism identifies highly predictive brain features but falls short of biomarker standards. NeuroImage: Clinical 7, 359–366 (2015)

[383] Polat, H., Danaei Mehr, H.: Classification of pulmonary CT images by using hybrid 3D-deep convolutional neural network architecture. Applied Sciences 9(5), 940 (2019)

[384] Poudel, R.P., Lamata, P., Montana, G.: Recurrent fully convolutional neural networks for multi-slice MRI cardiac segmentation. In: Reconstruction, Segmentation, and Analysis of Medical Images, pp. 83–94. Springer (2016)

[385] Prasad, K., Atherton, J., Smith, G.C., McKenna, W.J., Frenneaux, M.P., Nihoyannopoulos, P.: Echocardiographic pitfalls in the diagnosis of hypertrophic cardiomyopathy. Heart 82(suppl 3), III8–III15 (1999)

[386] Pratt, H., Coenen, F., Broadbent, D.M., Harding, S.P., Zheng, Y.: Convolutional neural networks for diabetic retinopathy. Procedia Computer Science 90, 200–205 (2016)

[387] Prechelt, L.: Automatic early stopping using cross validation: quantifying the criteria. Neural Networks 11(4), 761–767 (1998)

[388] Puangmali, P., Liu, H., Seneviratne, L.D., Dasgupta, P., Althoefer, K.: Miniature 3-axis distal force sensor for minimally invasive surgical palpation. IEEE/ASME Transactions on Mechatronics 17(4), 646–656 (2012)

[389] Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3D classification and segmentation. In: Conference on Computer Vision and Pattern Recognition (2017)

[390] Qiu, J., Sun, Y.: Self-supervised iterative refinement learning for macular OCT volumetric data classification. Computers in Biology and Medicine 111, 103327 (2019)

[391] Qiu, Z., Yao, T., Mei, T.: Learning spatio-temporal representation with pseudo-3d residual networks. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 5533–5541 (2017)

[392] Qureshi, M.N.I., Ryu, S., Song, J., Lee, K.H., Lee, B.: Evaluation of functional decline in alzheimer's dementia using 3d deep learning and group ica for rs-fmri measurements. Frontiers in Aging Neuroscience 11, 8 (2019)

[393] Rahimy, E.: Deep learning applications in ophthalmology. Current Opinion in Ophthalmology 29(3), 254–260 (2018)

[394] Ranjan, A., Black, M.J.: Optical flow estimation using a spatial pyramid network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4161–4170 (2017)

[395] Rao, Y.J.: In-fibre Bragg grating sensors. Measurement Science and Technology 8(4), 355 (1997)

[396] Rasmussen, C.E.: Gaussian processes in machine learning. In: Advanced Lectures on Machine Learning. pp. 63–71 (2004)

[397] Rasti, R., Rabbani, H., Mehridehnavi, A., Hajizadeh, F.: Macular OCT classification using a multi-scale convolutional neural network ensemble. IEEE Transactions on Medical Imaging 37(4), 1024–1034 (2017)

[398] Ravenscroft, D., Deng, J., Xie, X., Terry, L., Margrain, T.H., North, R.V., Wood, A.: Learning feature extractors for AMD classification in OCT using convolutional neural networks. In: European Signal Processing Conference. pp. 51–55. IEEE (2017)

[399] Redmon, J., Farhadi, A.: Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767 (2018)

[400] Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. In: Advances in Neural Information Processing Systems. pp. 91–99 (2015)

[401] Ren, X., Wu, H., Chen, Q., Kubo, T., Akasaka, T.: A tissue classification method of IVOCT images using rectangle region cropped along the circumferential direction based on deep learning. In: International Forum on Medical Imaging in Asia 2019. vol. 11050, p. 1105015. International Society for Optics and Photonics (2019)

[402] Ren, Z., Yan, J., Ni, B., Liu, B., Yang, X., Zha, H.: Unsupervised deep learning for optical flow estimation. In: AAAI Conference on Artificial Intelligence (2017)

[403] Rey, D., Subsol, G., Delingette, H., Ayache, N.: Automatic detection and segmentation of evolving processes in 3D medical images: Application to multiple sclerosis. Medical Image Analysis 6(2), 163–179 (2002)

[404] Richter, L., Trillenberg, P., Schweikard, A., Schlaefer, A.: Stimulus intensity for hand held and robotic transcranial magnetic stimulation. Brain Stimulation 6(3), 315–321 (2013)

[405] Roa-Barco, L., Serradilla-Casado, O., de Velasco-Vázquez, M., López-Zorrilla, A., Graña, M., Chyzhyk, D., Price, C.: A 2d/3d convolutional neural network for brain white matter lesion detection in multimodal mri. In: International Conference on Computer Recognition Systems. pp. 377–385. Springer (2017)

[406] Rodrigues, S., Horeman, T., Sam, P., Dankelman, J., van den Dobbelsteen, J., Jansen, F.W.: Influence of visual force feedback on tissue handling in minimally invasive surgery. British Journal of Surgery 101(13), 1766–1773 (2014)

[407] Romaguera, L.V., Costa, M.G.F., Romero, F.P., Costa Filho, C.F.F.: Left ventricle segmentation in cardiac MRI images using fully convolutional neural networks. In: Medical Imaging 2017: Computer-Aided Diagnosis. vol. 10134, p. 101342Z. International Society for Optics and Photonics (2017)

[408] Rong, Y., Xiang, D., Zhu, W., Yu, K., Shi, F., Fan, Z., Chen, X.: Surrogate-assisted retinal OCT image classification based on convolutional neural networks. IEEE Journal of Biomedical and Health Informatics 23(1), 253–263 (2018)

[409] Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical Image Computing and Computer-assisted Intervention. pp. 234–241. Springer (2015)

[410] Roriz, P., Carvalho, L., Frazão, O., Santos, J.L., Simões, J.A.: From conventional sensors to fibre optic sensors for strain and force measurements in biomechanics applications: A review. Journal of Biomechanics 47(6), 1251–1261 (2014)

[411] Rosenblatt, F.: The perceptron: a probabilistic model for information storage and organization in the brain. Psychological review 65(6), 386 (1958)

[412] Roth, H.R., Le Lu, Liu, J., Yao, J., Seff, A., Cherry, K., Kim, L., Summers, R.M.: Improving Computer-Aided Detection Using Convolutional Neural Networks and Random View Aggregation. IEEE Transactions on Medical Imaging 35(5), 1170–1181 (2016)

[413] Roura, E., et al.: A toolbox for multiple sclerosis lesion segmentation. Neuroradiology 57(10), 1031–1043 (2015)

[414] Rovira, À., et al.: Evidence-based guidelines: MAGNIMS consensus guidelines on the use of MRI in multiple sclerosis—clinical implementation in the diagnostic process. Nature Reviews Neurology 11(8), 471 (2015)

[415] Roy, A.G., Conjeti, S., Carlier, S.G., Dutta, P.K., Kastrati, A., Laine, A.F., Navab, N., Katouzian, A., Sheet, D.: Lumen segmentation in intravascular optical coherence tomography using backscattering tracked and initialized random walks. IEEE Journal of Biomedical and Health Informatics 20(2), 606–614 (2016)

[416] Roy, A.G., Conjeti, S., Karri, S.P.K., Sheet, D., Katouzian, A., Wachinger, C., Navab, N.: ReLayNet: retinal layer and fluid segmentation of macular optical coherence tomography using fully convolutional networks. Biomedical Optics Express 8(8), 3627–3642 (2017)

[417] Roy, A.G., Navab, N., Wachinger, C.: Concurrent spatial and channel 'squeeze & excitation'in fully convolutional networks. In: International Conference on Medical Image Computing and Computer-assisted Intervention. pp. 421–429. Springer (2018)

[418] Roy, A.G., Navab, N., Wachinger, C.: Recalibrating fully convolutional networks with spatial and channel "squeeze and excitation" blocks. IEEE Transactions on Medical Imaging 38(2), 540–549 (2018)

[419] Roy, S., Butman, J.A., Reich, D.S., Calabresi, P.A., Pham, D.L.: Multiple sclerosis lesion segmentation from brain MRI via fully convolutional neural networks. arXiv preprint arXiv:1803.09172 (2018)

[420] Rumelhart, D.E., Durbin, R., Golden, R., Chauvin, Y.: Backpropagation: The basic theory. Backpropagation: Theory, Architectures and Applications pp. 1–34 (1995)

[421] Saha, S., Nassisi, M., Wang, M., Lindenberg, S., Sadda, S., Hu, Z.J., et al.: Automated detection and classification of early AMD biomarkers using deep learning. Scientific Reports 9(1), 1–9 (2019)

[422] Salem, M., et al.: A supervised framework with intensity subtraction and deformation field features for the detection of new T2-w lesions in multiple sclerosis. NeuroImage: Clinical 17, 607–615 (2018)

[423] Sarafianos, N., Boteanu, B., Ionescu, B., Kakadiaris, I.A.: 3d human pose estimation: A review of the literature and analysis of covariates. Computer Vision and Image Understanding 152, 1–20 (2016)

[424] Sarikaya, D., Corso, J.J., Guru, K.A.: Detection and localization of robotic tools in robot-assisted surgery videos using deep neural networks for region proposal and detection. IEEE Transactions on Medical Imaging 36(7), 1542–1549 (2017)

[425] Sarraf, S., Tofighi, G.: Deep learning-based pipeline to recognize Alzheimer's disease using fMRI data. In: Future Technologies Conference. pp. 816–820. IEEE (2016)

[426] Schlegl, T., Glodan, A.M., Podkowinski, D., Waldstein, S.M., Gerendas, B.S., Schmidt-Erfurth, U., Langs, G.: Automatic segmentation and classification of intraretinal cystoid fluid and subretinal fluid in 3D-OCT using convolutional neural networks. Investigative Ophthalmology & Visual Science 56(7), 5920 (2015)

[427] Schlegl, T., Waldstein, S.M., Bogunovic, H., Endstraßer, F., Sadeghipour, A., Philip, A.M., Podkowinski, D., Gerendas, B.S., Langs, G., Schmidt-Erfurth, U.: Fully automated detection and quantification of macular fluid in OCT using deep learning. Ophthalmology 125(4), 549–558 (2018)

[428] Schlüter, M., Otte, C., Saathoff, T., Gessert, N., Schlaefer, A.: Feasibility of a markerless tracking system based on optical coherence tomography. In: Medical Imaging 2019: Image-Guided Procedures, Robotic Interventions, and Modeling. vol. 10951, p. 1095107. International Society for Optics and Photonics (2019)

[429] Schmidt, P.: LST: A lesion segmentation tool for SPM (2017)

[430] Schmidt, P., et al.: An automated tool for detection of FLAIR-hyperintense white-matter lesions in multiple sclerosis. NeuroImage 59(4), 3774–3783 (2012)

[431] Schmidt, P., et al.: Automated segmentation of changes in FLAIR-hyperintense white matter lesions in multiple sclerosis on serial magnetic resonance imaging. NeuroImage: Clinical 23, 101849 (2019)

[432] Schmidt-Richberg, A., Brosch, T., Schadewaldt, N., Klinder, T., Cavallaro, A., Salim, I., Roundhill, D., Papageorghiou, A., Lorenz, C.: Abdomen segmentation in 3D fetal ultrasound using CNN-powered deformable models. In: Fetal, Infant and Ophthalmic Medical Image Analysis, pp. 52–61. Springer (2017)

[433] Schmitt, J.M., Xiang, S.H., Yung, K.M.: Speckle in optical coherence tomography: an overview. In: Saratov Fall Meeting'98: Light Scattering Technologies for Mechanics, Biomedicine, and Material Science. pp. 450–461 (1999)

[434] Schwaab, J., Prall, M., Sarti, C., Kaderka, R., Bert, C., Kurz, C., Parodi, K., Günther, M., Jenne, J.: Ultrasound tracking for intra-fractional motion compensation in radiation therapy. Physica Medica 30(5), 578–582 (2014)

[435] Sedai, S., Antony, B., Mahapatra, D., Garnavi, R.: Joint segmentation and uncertainty visualization of retinal layers in optical coherence tomography images using Bayesian deep learning. In: Computational Pathology and Ophthalmic Medical Image Analysis, pp. 219–227. Springer (2018)

[436] Sentker, T., Madesta, F., Werner, R.: GDL-FIRE4D: Deep Learning-Based Fast 4D CT Image Registration. In: International Conference on Medical Image Computing and Computer-assisted Intervention. pp. 765–773. Springer (2018)

[437] Serener, A., Serte, S.: Dry and wet age-related macular degeneration classification using oct images and deep learning. In: Scientific Meeting on Electrical-Electronics & Biomedical Engineering and Computer Science. pp. 1–4. IEEE (2019)

[438] Setio, A.A.A., Ciompi, F., Litjens, G., Gerke, P., Jacobs, C., Van Riel, S.J., Wille, M.M.W., Naqibullah, M., Sánchez, C.I., van Ginneken, B.: Pulmonary nodule detection in CT images: false positive reduction using multi-view convolutional networks. IEEE Transactions on Medical Imaging 35(5), 1160–1169 (2016)

[439] Shah, A., Zhou, L., Abrámoff, M.D., Wu, X.: Multiple surface segmentation using convolution neural nets: application to retinal layer segmentation in OCT images. Biomedical Optics Express 9(9), 4509–4526 (2018)

[440] Shahidi, M., Wang, Z., Zelkha, R.: Quantitative thickness measurement of retinal layers imaged by optical coherence tomography. American Journal of Ophthalmology 139(6), 1056–1061 (2005)

[441] Shan, H., Zhang, Y., Yang, Q., Kruger, U., Kalra, M.K., Sun, L., Cong, W., Wang, G.: 3-D convolutional encoder-decoder network for low-dose CT via transfer learning from a 2-D trained network. IEEE Transactions on Medical Imaging 37(6), 1522–1534 (2018)

[442] Shen, T., Li, X., Zhong, Z., Wu, J., Lin, Z.: R 2-Net: Recurrent and Recursive Network for Sparse-View CT Artifacts Removal. In: International Conference on Medical Image Computing and Computer-assisted Intervention. pp. 319–327. Springer (2019)

[443] Shi, F., Chen, X., Zhao, H., Zhu, W., Xiang, D., Gao, E., Sonka, M., Chen, H.: Automated 3-D retinal layer segmentation of macular optical coherence tomography images with serous pigment epithelial detachments. IEEE Transactions on Medical Imaging 34(2), 441–452 (2014)

[444] Shiee, N., et al.: A topology-preserving approach to the segmentation of brain images with multiple sclerosis lesions. NeuroImage 49(2), 1524–1535 (2010)

[445] Shin, H., Cho, H., Kim, D., Ko, D.K., Lim, S.C., Hwang, W.: Sequential Image-based Attention Network for Inferring Force Estimation without Haptic Sensor. IEEE Access 7, 150237–150246 (2019)

[446] Shin, H.C., Roberts, K., Lu, L., Demner-Fushman, D., Yao, J., Summers, R.M.: Learning to read chest x-rays: Recurrent neural cascade model for automated image annotation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2497–2506 (2016)

[447] Shin, H.C., Roth, H.R., Gao, M., Lu, L., Xu, Z., Nogues, I., Yao, J., Mollura, D., Summers, R.M.: Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning. IEEE Transactions on Medical Imaging 35(5), 1285–1298 (2016)

[448] Shrivastava, A., Sukthankar, R., Malik, J., Gupta, A.: Beyond skip connections: Top-down modulation for object detection. arXiv preprint arXiv:1612.06851 (2016)

[449] Siddiqui, M., Nam, A.S., Tozburun, S., Lippok, N., Blatter, C., Vakoc, B.J.: High-speed optical coherence tomography by circular interferometric ranging. Nature Photonics 12(2), 111 (2018)

[450] Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)

[451] Singh, S.P., Wang, L., Gupta, S., Goli, H., Padmanabhan, P., Gulyás, B.: 3D Deep Learning on Medical Images: A Review. arXiv preprint arXiv:2004.00218 (2020)

[452] Slichter, C.P.: Principles of magnetic resonance, vol. 1. Springer Science & Business Media (2013)

[453] Smistad, E., Lindseth, F.: Real-time tracking of the left ventricle in 3D ultrasound using Kalman filter and mean value coordinates. Medical Image Segmentation for Improved Surgical Navigation 189 (2014)

[454] Smistad, E., Østvik, A., Salte, I.M., Leclerc, S., Bernard, O., Lovstakken, L.: Fully automatic real-time ejection fraction and MAPSE measurements in 2D echocardiography using deep neural networks. In: IEEE International Ultrasonics Symposium. pp. 1–4. IEEE (2018)

[455] Smistad, E., Østvik, A., et al.: 2D left ventricle segmentation using deep learning. In: IEEE International Ultrasonics Symposium. pp. 1–4. IEEE (2017)

[456] Snoek, J., Larochelle, H., Adams, R.P.: Practical bayesian optimization of machine learning algorithms. In: Advances in Neural Information Processing Systems. pp. 2951–2959 (2012)

[457] Sokhanvar, S., Dargahi, J., Najarian, S., Arbatani, S.: Clinical and regulatory challenges for medical devices tactile sensing and displays. Haptic Feedback for Minimally Invasive Surgery and Robotics (2012)

[458] Song, H., Wang, W., Zhao, S., Shen, J., Lam, K.M.: Pyramid dilated deeper convlstm for video salient object detection. In: Proceedings of the European conference on computer vision. pp. 715–731 (2018)

[459] Song, S., Lan, C., Xing, J., Zeng, W., Liu, J.: An end-to-end spatio-temporal attention model for human action recognition from skeleton data. In: Thirty-first AAAI conference on artificial intelligence (2017)

[460] Song, Y., Cai, W., Zhou, Y., Feng, D.D.: Feature-based image patch approximation for lung tissue classification. IEEE Transactions on Medical Imaging 32(4), 797–808 (2013)

[461] Sørensen, T.: A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyses of the vegetation on Danish commons. Kongelige Danske Videnskabernes Selskab 5(4), 1–34 (1948)

[462] Sprawls, P.: Magnetic resonance imaging: principles, methods, and techniques. Medical Physics Publishing (2000)

[463] Springenberg, J.T., Dosovitskiy, A., Brox, T., Riedmiller, M.: Striving for simplicity: The all convolutional net. In: International Conference on Learning Representations (2015)

[464] Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research 15(1), 1929–1958 (2014)

[465] Steven, P., Le Blanc, C., Velten, K., Lankenau, E., Krug, M., Oelckers, S., Heindl, L.M., Gehlsen, U., Hüttmann, G., Cursiefen, C.: Optimizing descemet membrane endothelial keratoplasty using intraoperative optical coherence tomography. JAMA Ophthalmology 131(9), 1135–1142 (2013)

[466] Su, H., Zervas, M., Furlong, C., Fischer, G.S.: A miniature MRI-compatible fiber-optic force sensor utilizing fabry-perot interferometer. In: MEMS and Nanotechnology, Volume 4. pp. 131–136 (2011)

[467] Sudre, C.H., Li, W., Vercauteren, T., Ourselin, S., Cardoso, M.J.: Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. In: Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support, pp. 240–248. Springer (2017)

[468] Sugmk, J., Kiattisin, S., Leelasantitham, A.: Automated classification between age-related macular degeneration and diabetic macular edema in OCT image using image segmentation. In: The Biomedical Engineering International Conference. pp. 1–4. IEEE (2014)

[469] Suinesiaputra, A., Bluemke, D.A., Cowan, B.R., Friedrich, M.G., Kramer, C.M., Kwong, R., Plein, S., Schulz-Menger, J., Westenberg, J.J., Young, A.A., et al.:

Quantification of LV function and mass by cardiovascular magnetic resonance: multi-center variability and consensus contours. Journal of Cardiovascular Magnetic Resonance 17(1), 63 (2015)

[470] Suk, H.I., Shen, D.: Deep learning-based feature representation for AD/MCI classification. In: International Conference on Medical Image Computing and Computer-assisted Intervention. pp. 583–590. Springer (2013)

[471] Suk, H.I., Wee, C.Y., Lee, S.W., Shen, D.: State-space model with deep learning for functional dynamics estimation in resting-state fMRI. NeuroImage 129, 292–307 (2016)

[472] Sun, L., Jia, K., Yeung, D.Y., Shi, B.E.: Human action recognition using factorized spatio-temporal convolutional networks. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 4597–4605 (2015)

[473] Sutskever, I., Martens, J., Dahl, G., Hinton, G.: On the importance of initialization and momentum in deep learning. In: Proceedings of the International Conference on Machine Learning. pp. 1139–1147 (2013)

[474] Sweeney, E., Shinohara, R., Shea, C., Reich, D., Crainiceanu, C.M.: Automatic lesion incidence estimation and detection in multiple sclerosis using multisequence longitudinal MRI. American Journal of Neuroradiology 34(1), 68–73 (2013)

[475] Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.A.: Inception-v4, inception-resnet and the impact of residual connections on learning. In: AAAI Conference on Artificial Intelligence (2017)

[476] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1–9 (2015)

[477] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2818–2826 (2016)

[478] Szkulmowski, M., Wojtkowski, M.: Averaging techniques for OCT imaging. Optics Express 21(8), 9757–9773 (2013)

[479] Tan, A.C., Tan, G.S., Denniston, A.K., Keane, P.A., Ang, M., Milea, D., Chakravarthy, U., Cheung, C.M.G.: An overview of the clinical applications of optical coherence tomography angiography. Eye 32(2), 262 (2018)

[480] Tan, L.K., Liew, Y.M., Lim, E., McLaughlin, R.A.: Convolutional neural network regression for short-axis left ventricle segmentation in cardiac cine MR sequences. Medical Image Analysis 39, 78–86 (2017)

[481] Tan, L.K., McLaughlin, R.A., Lim, E., Abdul Aziz, Y.F., Liew, Y.M.: Fully automated segmentation of the left ventricle in cine cardiac MRI using neural network regression. Journal of Magnetic Resonance Imaging 48(1), 140–152 (2018)

[482] Tan, M., Le, Q.: EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In: International Conference on Machine Learning. pp. 6105–6114 (2019)

[483] Tao, Q., Yan, W., Wang, Y., Paiman, E.H., Shamonin, D.P., Garg, P., Plein, S., Huang, L., Xia, L., Sramko, M., et al.: Deep learning–based method for fully automatic quantification of left ventricle function from cine MR images: a multivendor, multicenter study. Radiology 290(1), 81–88 (2019)

[484] Taylor, R.H., Menciassi, A., Fichtinger, G., Fiorini, P., Dario, P.: Medical robotics and computer-integrated surgery. In: Springer Handbook of Robotics. pp. 1657–1684 (2016)

[485] Team, N.L.S.T.R.: Reduced lung-cancer mortality with low-dose computed tomographic screening. New England Journal of Medicine 365(5), 395–409 (2011)

[486] Tearney, G.J., Regar, E., Akasaka, T., Adriaenssens, T., Barlis, P., Bezerra, H.G., Bouma, B., Bruining, N., Cho, J.m., Chowdhary, S., et al.: Consensus standards for acquisition, measurement, and reporting of intravascular optical coherence tomography studies: a report from the International Working Group for Intravascular Optical Coherence Tomography Standardization and Validation. Journal of the American College of Cardiology 59(12), 1058–1072 (2012)

[487] Tekin, B., Sinha, S.N., Fua, P.: Real-time seamless single shot 6d object pose prediction. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 292–301 (2018)

[488] Thong, W., Kadoury, S., Piché, N., Pal, C.J.: Convolutional networks for kidney segmentation in contrast-enhanced CT scans. Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization 6(3), 277–282 (2018)

[489] Tieleman, T., Hinton, G.: Lecture 6.5-RMSProp, COURSERA: Neural networks for machine learning. University of Toronto, Tech. Rep (2012)

[490] Tompson, J., Goroshin, R., Jain, A., LeCun, Y., Bregler, C.: Efficient object localization using convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 648–656 (2015)

[491] Tran, D., Bourdev, L., Fergus, R., Torresani, L., Paluri, M.: Learning spatiotemporal features with 3d convolutional networks. In: The IEEE International Conference on Computer Vision. pp. 4489–4497 (2015)

[492] Trejos, A., Patel, R., Naish, M.: Force sensing and its application in minimally invasive surgery and therapy: a survey. Proceedings of the Institution of Mechanical

Engineers, Part C: Journal of Mechanical Engineering Science 224(7), 1435–1454 (2010)

[493] Tsantis, S., Kagadis, G.C., Katsanos, K., Karnabatidis, D., Bourantas, G., Nikiforidis, G.C.: Automatic vessel lumen segmentation and stent strut detection in intravascular optical coherence tomography. Medical Physics 39(1), 503–513 (2012)

[494] Tuchin, V.: Tissue optics: light scattering methods and instruments for medical diagnosis. SPIE Press Bellingham (2007)

[495] Ughi, G.J., Van Dyck, C.J., Adriaenssens, T., Hoymans, V.Y., Sinnaeve, P., Timmermans, J.P., Desmet, W., Vrints, C.J., D'hooge, J.: Automatic assessment of stent neointimal coverage by intravascular optical coherence tomography. European Heart Journal–Cardiovascular Imaging 15(2), 195–200 (2014)

[496] Ughi, G.J., Adriaenssens, T., Sinnaeve, P., Desmet, W., D'hooge, J.: Automated tissue characterization of in vivo atherosclerotic plaques by intravascular optical coherence tomography images. Biomedical Optics Express 4(7), 1014–1030 (2013)

[497] Ullah, A., Ahmad, J., Muhammad, K., Sajjad, M., Baik, S.W.: Action recognition in video sequences using deep bi-directional LSTM with CNN features. IEEE Access 6, 1155–1166 (2017)

[498] Ulyanov, D., Vedaldi, A., Lempitsky, V.: Instance normalization: The missing ingredient for fast stylization. arXiv preprint arXiv:1607.08022 (2016)

[499] Uppaluri, R., Hoffman, E.A., Sonka, M., Hartley, P.G., Hunninghake, G.W., McLennan, G.: Computer recognition of regional lung disease patterns. American Journal of Respiratory and Critical Care Medicine 160(2), 648–654 (1999)

[500] Vaidya, S., Chunduru, A., Muthuganapathy, R., Krishnamurthi, G.: Longitudinal multiple sclerosis lesion segmentation using 3D convolutional neural networks. Proceedings of the 2015 Longitudinal Multiple Sclerosis Lesion Segmentation Challenge pp. 1–2 (2015)

[501] Valverde, S., Salem, M., Cabezas, M., Pareto, D., Vilanova, J.C., Ramió-Torrentà, L., Rovira, À., Salvi, J., Oliver, A., Lladó, X.: One-shot domain adaptation in multiple sclerosis lesion segmentation using convolutional neural networks. NeuroImage: Clinical 21, 101638 (2019)

[502] Valverde, S., et al.: Improving automated multiple sclerosis lesion segmentation with a cascaded 3D convolutional neural network approach. NeuroImage 155, 159–168 (2017)

[503] Van Ginneken, B., Setio, A.A., Jacobs, C., Ciompi, F.: Off-the-shelf convolutional neural network features for pulmonary nodule detection in computed tomography scans. In: IEEE International Symposium on Biomedical Imaging. pp. 286–289. IEEE (2015)

[504] Van Leemput, K., et al.: Automated segmentation of multiple sclerosis lesions by model outlier detection. IEEE Transactions on Medical Imaging 20(8), 677–688 (2001)

[505] Van Pelt, R., Bescos, J.O., Breeuwer, M., Clough, R.E., Groller, M.E., ter Haar Romenij, B., Vilanova, A.: Exploration of 4D MRI blood flow using stylistic visualization. IEEE Transactions on Visualization and Computer Graphics 16(6), 1339–1347 (2010)

[506] Van Soest, G., Goderie, T.P., Regar, E., Koljenovic, S., van Leenders, A.G.J., Gonzalo, N., van Noorden, S., Okamura, T., Bouma, B.E., Tearney, G.J., et al.: Atherosclerotic tissue characterization in vivo by optical coherence tomography attenuation imaging. Journal of Biomedical Optics 15(1), 011105 (2010)

[507] Vapnik, V.: The nature of statistical learning theory. Springer Science & Business Media (2013)

[508] Varior, R.R., Haloi, M., Wang, G.: Gated siamese convolutional neural network architecture for human re-identification. In: Proceedings of the European Conference on Computer Vision. pp. 791–808. Springer (2016)

[509] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems. pp. 5998–6008 (2017)

[510] Venhuizen, F.G., Grinsven, M., Hoyng, C.B.: Vendor Independent Cyst Segmentation in Retinal SD-OCT Volumes using a Combination of Multiple Scale Convolutional Neural Networks. In: Medical Image Computing and Computer-assisted Intervention - Challenge on Retinal Cyst Segmentation (2015)

[511] Venhuizen, F.G., van Ginneken, B., Bloemen, B., van Grinsven, M.J., Philipsen, R., Hoyng, C., Theelen, T., Sánchez, C.I.: Automated age-related macular degeneration classification in OCT using unsupervised feature learning. In: Medical Imaging 2015: Computer-Aided Diagnosis. vol. 9414, p. 94141I. International Society for Optics and Photonics (2015)

[512] Vermeer, K., Mo, J., Weda, J., Lemij, H., De Boer, J.: Depth-resolved model-based reconstruction of attenuation coefficients in optical coherence tomography. Biomedical Optics Express 5(1), 322–337 (2014)

[513] Viehland, C., Keller, B., Carrasco-Zevallos, O.M., Nankivil, D., Shen, L., Mangalesh, S., Kuo, A.N., Toth, C.A., Izatt, J.A., et al.: Enhanced volumetric visualization for real time 4D intraoperative ophthalmic swept-source OCT. Biomedical Optics Express 7(5), 1815–1829 (2016)

[514] de Vos, B.D., Berendsen, F.F., Viergever, M.A., Sokooti, H., Staring, M., Išgum, I.: A deep learning framework for unsupervised affine and deformable image registration. Medical Image Analysis 52, 128–143 (2019)

[515] Voulodimos, A., Doulamis, N., Doulamis, A., Protopapadakis, E.: Deep learning for computer vision: A brief review. Computational intelligence and neuroscience 2018 (2018)

[516] Walsh, S.L., Calandriello, L., Silva, M., Sverzellati, N.: Deep learning for classifying fibrotic lung disease on high-resolution computed tomography: a case-cohort study. The Lancet Respiratory Medicine 6(11), 837–845 (2018)

[517] Wan, L., Zeiler, M., Zhang, S., Le Cun, Y., Fergus, R.: Regularization of neural networks using dropconnect. In: Proceedings of the International Conference on Machine Learning. pp. 1058–1066 (2013)

[518] Wang, A., Eggermont, J., Dekker, N., Garcia-Garcia, H.M., Pawar, R., Reiber, J.H., Dijkstra, J.: Automatic stent strut detection in intravascular optical coherence tomographic pullback runs. The International Journal of Cardiovascular Imaging 29(1), 29–38 (2013)

[519] Wang, A., Nakatani, S., Eggermont, J., Onuma, Y., Garcia-Garcia, H.M., Serruys, P.W., Reiber, J.H., Dijkstra, J.: Automatic detection of bioresorbable vascular scaffold struts in intravascular optical coherence tomography pullback runs. Biomedical Optics Express 5(10), 3589–3602 (2014)

[520] Wang, C., Xu, D., Zhu, Y., Martín-Martín, R., Lu, C., Fei-Fei, L., Savarese, S.: Densefusion: 6d object pose estimation by iterative dense fusion. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3343–3352 (2019)

[521] Wang, D., Wang, L.: On OCT image classification via deep learning. IEEE Photonics Journal 11(5), 1–14 (2019)

[522] Wang, F., Jiang, M., Qian, C., Yang, S., Li, C., Zhang, H., Wang, X., Tang, X.: Residual attention network for image classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3156–3164 (2017)

[523] Wang, F., Biswal, B.: Neural Architecture Search for Gliomas Segmentation on Multimodal Magnetic Resonance Imaging. arXiv preprint arXiv:2005.06338 (2020)

[524] Wang, F., Tax, D.M.: Survey on the attention based RNN model and its applications in computer vision. arXiv preprint arXiv:1601.06823 (2016)

[525] Wang, P., Li, W., Ogunbona, P., Wan, J., Escalera, S.: RGB-D-based human motion recognition with deep learning: A survey. Computer Vision and Image Understanding 171, 118–139 (2018)

[526] Wang, T., Chen, Y., Qiao, M., Snoussi, H.: A fast and robust convolutional neural network-based defect detection model in product quality control. The International Journal of Advanced Manufacturing Technology 94(9-12), 3465–3471 (2018)

[527] Wang, T., Pfeiffer, T., Regar, E., Wieser, W., van Beusekom, H., Lancee, C.T., Springeling, G., Krabbendam-Peters, I., van der Steen, A.F., Huber, R., et al.: Heartbeat OCT and motion-free 3D in vivo coronary artery microscopy. JACC: Cardiovascular Imaging 9(5), 622–623 (2016)

[528] Wang, W., Wanga, Y., Wu, Y., Lin, T., Li, S., Chen, B.: Quantification of Full Left Ventricular metrics via Deep Regression Learning with Contour-guidance. IEEE Access (2019)

[529] Wang, Y., Bower, B.A., Izatt, J.A., Tan, O., Huang, D.: Retinal blood flow measurement by circumpapillary Fourier domain Doppler optical coherence tomography. Journal of Biomedical Optics 13(6), 064003 (2008)

[530] Wang, Y., Fawzi, A., Tan, O., Gil-Flamer, J., Huang, D.: Retinal blood flow detection in diabetic patients by Doppler Fourier domain optical coherence tomography. Optics Express 17(5), 4061 (2009)

[531] Wang, Y., Zhang, Y., Yao, Z., Zhao, R., Zhou, F.: Machine learning based detection of age-related macular degeneration (AMD) and diabetic macular edema (DME) from optical coherence tomography (OCT) images. Biomedical Optics Express 7(12), 4928–4940 (2016)

[532] Wang, Z., Chamie, D., Bezerra, H.G., Yamamoto, H., Kanovsky, J., Wilson, D.L., Costa, M.A., Rollins, A.M.: Volumetric quantification of fibrous caps using intravascular optical coherence tomography. Biomedical Optics Express 3(6), 1413–1426 (2012)

[533] Wang, Z., Jenkins, M.W., Linderman, G.C., Bezerra, H.G., Fujino, Y., Costa, M.A., Wilson, D.L., Rollins, A.M.: 3-D stent detection in intravascular OCT using a Bayesian network and graph search. IEEE Transactions on Medical Imaging 34(7), 1549–1561 (2015)

[534] Warfield, S.K., Kaus, M., Jolesz, F.A., Kikinis, R.: Adaptive, template moderated, spatially varying statistical classification. Medical Image Analysis 4(1), 43–55 (2000)

[535] Weiss, E., Wijesooriya, K., Dill, S.V., Keall, P.J.: Tumor and normal tissue motion in the thorax during respiration: Analysis of volumetric and positional variations using 4D CT. International Journal of Radiation Oncology* Biology* Physics 67(1), 296–307 (2007)

[536] Weng, Y., Zhou, T., Li, Y., Qiu, X.: NAS-Unet: Neural architecture search for medical image segmentation. IEEE Access 7, 44247–44257 (2019)

[537] Wieser, W., Draxinger, W., Klein, T., Karpf, S., Pfeiffer, T., Huber, R.: High definition live 3D-OCT in vivo: design and evaluation of a 4D OCT engine with 1 GVoxel/s. Biomedical Optics Express 5(9), 2963–2977 (2014)

[538] Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning 8(3-4), 229–256 (1992)

[539] Wilson, D.R., Martinez, T.R.: The general inefficiency of batch training for gradient descent learning. Neural Networks 16(10), 1429–1451 (2003)

[540] Wilson, E.B., Bagshahi, H., Woodruff, V.D.: Overview of general advantages, limitations, and strategies. In: Robotics in General Surgery, pp. 17–22. Springer (2014)

[541] Wohlhart, P., Lepetit, V.: Learning descriptors for object recognition and 3d pose estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3109–3118 (2015)

[542] Wolterink, J.M., Leiner, T., de Vos, B.D., van Hamersvelt, R.W., Viergever, M.A., Išgum, I.: Automatic coronary artery calcium scoring in cardiac CT angiography using paired convolutional neural networks. Medical Image Analysis 34, 123–136 (2016)

[543] Wu, B., Iandola, F., Jin, P.H., Keutzer, K.: Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 129–137 (2017)

[544] Wu, G., Kim, M., Wang, Q., Munsell, B.C., Shen, D.: Scalable high-performance image registration framework by unsupervised deep feature representations learning. IEEE Transactions on Biomedical Engineering 63(7), 1505–1516 (2015)

[545] Wu, J., Zhang, Y., Wang, J., Zhao, J., Ding, D., Chen, N., Wang, L., Chen, X., Jiang, C., Zou, X., et al.: AttenNet: Deep Attention Based Retinal Disease Classification in OCT Images. In: International Conference on Multimedia Modeling. pp. 565–576. Springer (2020)

[546] Wu, L., Fernandez-Loaiza, P., Sauma, J., Hernandez-Bogantes, E., Masis, M.: Classification of diabetic retinopathy and diabetic macular edema. World Journal of Diabetes 4(6), 290 (2013)

[547] Wu, Y., Warfield, S.K., Tan, I.L., Wells III, W.M., Meier, D.S., van Schijndel, R.A., Barkhof, F., Guttmann, C.R.: Automated segmentation of multiple sclerosis lesion subtypes with multichannel MRI. NeuroImage 32(3), 1205–1215 (2006)

[548] Wu, Y., He, K.: Group normalization. In: Proceedings of the European Conference on Computer Vision. pp. 3–19 (2018)

[549] Xiang, Y., Schmidt, T., Narayanan, V., Fox, D.: PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes. In: Robotics: Science and Systems (RSS) (2018)

[550] Xiao, T., Li, H., Ouyang, W., Wang, X.: Learning deep feature representations with domain guided dropout for person re-identification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1249–1258 (2016)

[551] Xie, D., Bai, L., Wang, Z.: Denoising arterial spin labeling cerebral blood flow images using deep learning. arXiv preprint arXiv:1801.09672 (2018)

[552] Xie, H., Yang, D., Sun, N., Chen, Z., Zhang, Y.: Automated pulmonary nodule detection in CT images using deep convolutional neural networks. Pattern Recognition 85, 109–119 (2019)

[553] Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1492–1500 (2017)

[554] Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: IEEE Conference on Computer Vision and Pattern Recognition. pp. 5987–5995. IEEE (2017)

[555] Xie, Y., Cham, M.D., Henschke, C., Yankelevitz, D., Reeves, A.P.: Automated coronary artery calcification detection on low-dose chest CT images. In: Medical Imaging 2014: Computer-Aided Diagnosis. vol. 9035, p. 90350F. International Society for Optics and Photonics (2014)

[556] Xingjian, S., Chen, Z., Wang, H., Yeung, D.Y., Wong, W.K., Woo, W.c.: Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In: Advances in Neural Information Processing Systems. pp. 802–810 (2015)

[557] Xu, C., Schmitt, J.M., Carlier, S.G., Virmani, R.: Characterization of atherosclerosis plaques by measuring both backscattering and attenuation coefficients in optical coherence tomography. Journal of Biomedical Optics 13(3), 034003 (2008)

[558] Xu, H., Schneider, J.E., Grau, V.: Calculation of Anatomical and Functional Metrics Using Deep Learning in Cardiac MRI: Comparison Between Direct and Segmentation-Based Estimation. In: International Workshop on Statistical Atlases and Computational Models of the Heart. pp. 402–411. Springer (2018)

[559] Xu, J., Ishikawa, H., Wollstein, G., Kagemann, L., Schuman, J.S.: Alignment of 3-D optical coherence tomography scans to correct eye movement using a particle filtering. IEEE Transactions on Medical Imaging 31(7), 1337–1345 (2012)

[560] Xu, M., Cheng, J., Li, A., Lee, J.A., Wong, D.W.K., Taruya, A., Tanaka, A., Foin, N., Wong, P.: Fibroatheroma identification in intravascular optical coherence tomography images using deep features. In: International Conference of the IEEE Engineering in Medicine and Biology Society. pp. 1501–1504. IEEE (2017)

[561] Xue, W., Brahm, G., Pandey, S., Leung, S., Li, S.: Full left ventricle quantification via deep multitask relationships learning. Medical Image Analysis 43, 54–65 (2018)

[562] Xue, W., Islam, A., Bhaduri, M., Li, S.: Direct multitype cardiac indices estimation via joint representation and regression learning. IEEE Transactions on Medical Imaging 36(10), 2057–2067 (2017)

[563] Xue, W., Lum, A., Mercado, A., Landis, M., Warrington, J., Li, S.: Full quantification of left ventricle via deep multitask learning network respecting intra-and inter-task relatedness. In: International Conference on Medical Image Computing and Computer-assisted Intervention. pp. 276–284. Springer (2017)

[564] Yabushita, H., Bouma, B.E., Houser, S.L., Aretz, H.T., Jang, I.K., Schlendorf, K.H., Kauffman, C.R., Shishkov, M., Kang, D.H., Halpern, E.F., et al.: Characterization of human atherosclerosis by optical coherence tomography. Circulation 106(13), 1640–1645 (2002)

[565] Yang, X., Kwitt, R., Styner, M., Niethammer, M.: Quicksilver: Fast predictive image registration–a deep learning approach. NeuroImage 158, 378–396 (2017)

[566] Yang, X., Zeng, Z., Yi, S.: Deep convolutional neural networks for automatic segmentation of left ventricle cavity from cardiac magnetic resonance images. IET Computer Vision 11(8), 643–649 (2017)

[567] Yazdanpanah, A., Hamarneh, G., Smith, B., Sarunic, M.: Intra-retinal layer segmentation in optical coherence tomography using an active contour approach. In: International Conference on Medical Image Computing and Computer-assisted Intervention. pp. 649–656. Springer (2009)

[568] Yin, Z., Shi, J.: Geonet: Unsupervised learning of dense depth, optical flow and camera pose. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1983–1992 (2018)

[569] Yokoi, K., Kamiya, N., Matsuguma, H., Machida, S., Hirose, T., Mori, K., Tominaga, K.: Detection of brain metastasis in potentially operable non-small cell lung cancer: a comparison of CT and MRI. Chest 115(3), 714–719 (1999)

[570] Yong, Y.L., Tan, L.K., McLaughlin, R.A., Chee, K.H., Liew, Y.M.: Linear-regression convolutional neural network for fully automated coronary lumen segmentation in intravascular optical coherence tomography. Journal of Biomedical Optics 22(12), 126005 (2017)

[571] Yoo, T.K., Choi, J.Y., Seo, J.G., Ramasubramanian, B., Selvaperumal, S., Kim, D.W.: The possibility of the combination of OCT and fundus images for improving the diagnostic accuracy of deep learning for age-related macular degeneration: a preliminary experiment. Medical & Biological Engineering & Computing 57(3), 677–687 (2019)

[572] Yoo, Y., Brosch, T., Traboulsee, A., Li, D.K., Tam, R.: Deep learning of image features from unlabeled data for multiple sclerosis lesion segmentation. In: International Workshop on Machine Learning in Medical Imaging. pp. 117–124. Springer (2014)

[573] Yosinski, J., Clune, J., Bengio, Y., Lipson, H.: How transferable are features in deep neural networks? In: Advances in Neural Information Processing Systems. pp. 3320–3328 (2014)

[574] Yu, L., Chen, H., Dou, Q., Qin, J., Heng, P.A.: Automated melanoma recognition in dermoscopy images via very deep residual networks. IEEE Transactions on Medical Imaging 36(4), 994–1004 (2017)

[575] Yu, L., Yang, X., Chen, H., Qin, J., Heng, P.A.: Volumetric ConvNets with mixed residual connections for automated prostate segmentation from 3D MR images. In: AAAI Conference on Artificial Intelligence (2017)

[576] Zagoruyko, S., Komodakis, N.: Wide residual networks. arXiv preprint arXiv:1605.07146 (2016)

[577] Zahnd, G., Karanasos, A., van Soest, G., Regar, E., Niessen, W., Gijsen, F., van Walsum, T.: Quantification of fibrous cap thickness in intracoronary optical coherence tomography with a contour segmentation method based on dynamic programming. International Journal of Computer Assisted Radiology and Surgery 10(9), 1383–1394 (2015)

[578] Zaremba, W., Sutskever, I., Vinyals, O.: Recurrent neural network regularization. arXiv preprint arXiv:1409.2329 (2014)

[579] Zawadzki, R.J., Fuller, A.R., Choi, S.S., Wiley, D.F., Hamann, B., Werner, J.S.: Correction of motion artifacts and scanning beam distortions in 3D ophthalmic optical coherence tomography imaging. In: Ophthalmic Technologies XVII. vol. 6426, p. 642607. International Society for Optics and Photonics (2007)

[580] Zbontar, J., LeCun, Y.: Computing the stereo matching cost with a convolutional neural network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1592–1599 (2015)

[581] Zeiler, M.D.: ADADELTA: an adaptive learning rate method. arXiv preprint arXiv:1212.5701 (2012)

[582] Zeiler, M.D., Fergus, R.: Stochastic pooling for regularization of deep convolutional neural networks. arXiv preprint arXiv:1301.3557 (2013)

[583] Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: Proceedings of the European Conference on Computer Vision. pp. 818–833 (2014)

[584] Zeng, A., Yu, K.T., Song, S., Suo, D., Walker, E., Rodriguez, A., Xiao, J.: Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge. In: 2017 IEEE International Conference on Robotics and Automation. pp. 1386–1383. IEEE (2017)

[585] Zeng, L.L., Wang, H., Hu, P., Yang, B., Pu, W., Shen, H., Chen, X., Liu, Z., Yin, H., Tan, Q., et al.: Multi-site diagnostic classification of schizophrenia using discriminant deep learning with functional connectivity MRI. EBioMedicine 30, 74–85 (2018)

[586] Zhang, C., Li, H., Guo, X., Molony, D., Guo, X., Samady, H., Giddens, D., Athanasiou, L., Nie, R., Cao, J., et al.: Convolution Neural Networks and Support Vector Machines for Automatic Segmentation of Intracoronary Optical Coherence Tomography. Molecular & Cellular Biomechanics 16(2), 153–161 (2019)

[587] Zhang, K., Wang, W., Han, J., Kang, J.U.: A surface topology and motion compensation system for microsurgery guidance and intervention based on common-path optical coherence tomography. IEEE Transactions on Biomedical Engineering 56(9), 2318–2321 (2009)

[588] Zhang, S., Guo, S., Huang, W., Scott, M.R., Wang, L.: V4D: 4D Convolutional Neural Networks for Video-level Representation Learning. arXiv preprint arXiv:2002.07442 (2020)

[589] Zhang, Y., Pfeiffer, T., Weller, M., Wieser, W., Huber, R., Raczkowsky, J., Schipper, J., Wörn, H., Klenzner, T.: Optical coherence tomography guided laser cochleostomy: Towards the accuracy on tens of micrometer scale. BioMed Research International (2014)

[590] Zhang, Y., Wörn, H.: Optical coherence tomography as highly accurate optical tracking system. In: IEEE/ASME International Conference on Advanced Intelligent Mechatronics. pp. 1145–1150. IEEE, Piscataway, NJ (2014)

[591] Zhang, Z., Yang, L., Zheng, Y.: Translating and segmenting multimodal medical volumes with cycle-and shape-consistency generative adversarial network. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 9242–9251 (2018)

[592] Zhou, T., Brown, M., Snavely, N., Lowe, D.G.: Unsupervised learning of depth and ego-motion from video. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1851–1858 (2017)

[593] Zhou, X., Ito, T., Takayama, R., Wang, S., Hara, T., Fujita, H.: Three-dimensional CT image segmentation by combining 2D fully convolutional network with 3D majority voting. In: Deep Learning and Data Labeling for Medical Applications, pp. 111–120. Springer (2016)

[594] Zhou, Y.T., Chellappa, R., Vaid, A., Jenkins, B.K.: Image restoration using a neural network. IEEE Transactions on Acoustics, Speech, and Signal Processing 36(7), 1141–1151 (1988)

[595] Zhou, Y., Xu, J., Liu, Q., Li, C., Liu, Z., Wang, M., Zheng, H., Wang, S.: A radiomics approach with CNN for shear-wave elastography breast tumor classification. IEEE Transactions on Biomedical Engineering 65(9), 1935–1942 (2018)

[596] Zhu, G., Zhang, L., Shen, P., Song, J.: Multimodal gesture recognition using 3-D convolution and convolutional LSTM. IEEE Access 5, 4517–4524 (2017)

[597] Zhu, W., Liu, C., Fan, W., Xie, X.: Deeplung: Deep 3d dual path nets for automated pulmonary nodule detection and classification. In: 2018 IEEE Winter Conference on Applications of Computer Vision. pp. 673–681. IEEE (2018)

[598] Zijdenbos, A.P., Forghani, R., Evans, A.C.: Automatic" pipeline" analysis of 3-D MRI data for clinical trials: application to multiple sclerosis. IEEE Transactions on Medical Imaging 21(10), 1280–1291 (2002)

[599] Zilly, J., Buhmann, J.M., Mahapatra, D.: Glaucoma detection using entropy sampling and ensemble learning for automatic optic cup and disc segmentation. Computerized Medical Imaging and Graphics 55, 28–41 (2017)

[600] Zilly, J.G., Srivastava, R.K., Koutník, J., Schmidhuber, J.: Recurrent highway networks. In: Proceedings of the International Conference on Machine Learning. pp. 4189–4198 (2017)

[601] Zitova, B., Flusser, J.: Image registration methods: a survey. Image and Vision Computing 21(11), 977–1000 (2003)

[602] Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning. arXiv preprint arXiv:1611.01578 (2016)

[603] Zoph, B., Vasudevan, V., Shlens, J., Le, Q.V.: Learning transferable architectures for scalable image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 8697–8710 (2018)

[604] Zou, L., Zheng, J., Miao, C., Mckeown, M.J., Wang, Z.J.: 3D CNN based automatic diagnosis of attention deficit hyperactivity disorder using functional and structural MRI. IEEE Access 5, 23626–23636 (2017)

[605] Zreik, M., Leiner, T., De Vos, B.D., van Hamersvelt, R.W., Viergever, M.A., Išgum, I.: Automatic segmentation of the left ventricle in cardiac CT angiography using convolutional neural networks. In: IEEE International Symposium on Biomedical Imaging. pp. 40–43. IEEE (2016)

List of Figures

List of Tables

List of Abbreviations

AD Alzheimer's disease

ADHD attention deficit hyperactivity disorder

AMD age-related macular degeneration

ASD autism spectrum disorder

BM Bruch's membrane

cGRU convolutional gated recurrent units

CNN convolutional neural network

CT computed tomography

CV cross-validation

DN DenseNet

DR diabetic retinopathy

DSA digital subtraction angiography

ECG electrocardiography

EM electromagnetic

ENAS efficient neural architecture search

ER error rate

FC-NN fully-connected neural networks

FD-OCT Fourier-domain OCT

FLAIR fluid-attenuated inversion recovery

fMRI functional magnetic resonance imaging

FN false negatives

FOV field of view

FP false positives

GAP global average pooling

GMM Gaussian Mixture Models

GPM Gaussian process model

GRU gated recurrent units

ILM inner limiting membrane

IN Inception

IT inference time

IVOCT intravascular OCT

LFPR lesion-wise false positive rate

LSTM long short-term memory cell

LTPR lesion-wise true positive rate

LV left ventricle

MAE mean absolute error

MCI mild cognitive impairment

MIP maximum intensity projections

MIS minimally invasive surgery

MPI magnetic particle imaging

MRI magnetic resonance imaging

MSE mean squared error

NAS neural architecture search

NMR nuclear magnetic resonance

OCT optical coherence tomography

OCTA optical coherence tomography angiography

PC phase-contrast

PCC Pearson's correlation coefficient

PET positron emission tomography

POM polyoxymethylene

ReLU rectified linear units

rMAE relative mean absolute error

RN ResNet

RNN recurrent neural networks

ROI region of interest

RPEDC retinal pigment epithelium drusen complex

RWT regional wall thickness

RX ResNeXt

S spatial

SD-OCT spectral-domain OCT

SE Squeeze-and-Excitation Networks

SGD stochastic gradient descent

SP single-path

SR segmentation-based regularization

SS-OCT swept-source OCT

ST spatio-temporal

TD-OCT time-domain OCT

TN true negatives

TP true positives

TP two-path

US ultrasound

List of Symbols

a An attention map

$a_{1:T}$ A set of actions a controller can chose from

$A(\alpha_{lr}, j)$ Function that decays the learning rate α_{lr} at iteration j

a_p Image processing point operator for multiplication

α_{ReLU} Learnable parameter of parametric ReLU

A_{t_i} An A-Scan (1D profile) at time t_i

α Significance level for statistical significance tests

α_{lr} Learning rate of gradient descent-based optimization algorithm

α_{lr_0} Base learning rate of gradient descent-based optimization algorithm

α_{rot} Rotation angle along an image's first spatial dimension

B_0 External magnetic field of an MRI scanner

b_e Exponential moving average of previous rewards for NAS

b_M A neural network's learnable biases

b_p Image processing point operator for addition

B_{t_i} A B-Scan (2D slice) at time t_i

$\mathcal{B}$ Mini-batch that is a subset of a dataset $\mathcal{X}$

β_1 Moving average decay for first order statistical moments

β_2 Moving average decay for second order statistical moments

β_M Learnable biases of batch normalization layer

β_{rot} Rotation angle along an image's second spatial dimension

c_I Cumulative intensity histogram

c_{t_i} Cell state of an LSTM at time point t_i

C_{t_i} A C-Scan (3D volume) at time t_i

CV A set of CV splits

$\mathcal{D}$ A dataset of depth maps

d Depth coordinate in a polar image representation

d_{dc} Learning rate decay rate

d_i Size of a data tensor's dimension i

Δl_a OCT's axial resolution

d_{NN} Number of neurons in a neural network layer

D_{reg} Distance metric for image registration

ΔE_l Difference in energy levels between nuclei with parallel and antiparallel spin

$d_{p_{max}}$ Deformation depth in a needle experiment

E_{en} An ensembling function

E_M The experience a machine learning model learns from

E_{rad} Energy of an excitation pulse

ϵ Scalar for numerical stability

$f_I(x, y)$ Discrete image function of a raw 2D image with coordinates x and y

f_m A feature map in a neural network

$\mathcal{F}_M$ A set of machine learning models f_M

fg_{t_i} Output of an LSTM's forget gate at time point t_i

G Magnetic field of the gradient coils

g_a A neural network's activation function

$g_I(x, y)$ Discrete image function of a processed 2D image with coordinates x and y

g_k Growth rate of the DenseNet architecture

γ_g Nucleus-specific gyromagnetic ratio

γ_M Learnable weights of batch normalization layer

γ_{rot} Rotation angle along an image's third spatial dimension

H Entropy

h_I Intensity histogram

$\mathcal{H}$ An arbitrary transform performed by a neural network layer

H_M A set of hyperparameter configurations h_M

h_M Hyperparameters of a machine learning model

h_p Planck's constant

h_{t_i} Internal state of a recurrent neural network at time point t_i

i_I Image intensity value

$\hat{i}_{v_j}$ A voxel's intensity value at index j

in_{t_i} Output of an LSTM's input gate at time point t_i

J_P A performance measure for a machine learning model

K Kernel tensor of a convolutional layer

k_B Boltzmann constant

k_c Number of filters in a kernel tensor

k_{CV} Number of cross-validation folds

k_d Kernel size for the third spatial dimension

k_h Kernel size for the first spatial dimension

k_i Size of a kernel tensor's dimension i

k_{sup} Number of superimposed relaxation processes

k_t Kernel size for the temporal dimension

k_w Kernel size for the second spatial dimension

λ_b OCT's central wavelength

λ_P Weighting factor for the phase classification loss for LVQ

λ_{reg} Weighting factor for regularization methods

λ_S Weighting factor for the segmentation loss for LVQ

λ_w Weighting parameter that trades off regularization and parameter fitting

M Magnetization

m_{D_x} Filter mask of a difference operator along the x direction

$\mathcal{M}$ A dataset of intensity projections

m_G Mask of a Gaussian filter

M_G Gaussian distribution function

M_I Mutual information

m_I Image processing function

M_∞ Longitudinal magnetization in steady-state

m_L Filter mask of a Laplace operator

m_M Mask of a mean filter

m_{med} Mask region of a median filter

m_{P_x} Filter mask of a Prewitt operator along the x direction

m_{S_x} Filter mask of a Sobel operator along the x direction

m_{SD_x} Filter mask of a symmetric difference operator along the x direction

M_{xy} Transversal magnetization

M_z Longitudinal magnetization

μ Mean

μ_d Magnetic dipole moments

n_a The number of A-Scans in a B-Scan

n_{ap} Antiparallel spin occupation number

N_b Batch size

N_c Number of classes for a classification problem

$n_{c_{red}}$ Reduction factor of compression layers in a DenseNet architecture

n_{cd} Cardinality of the ResNeXt architecture

N_{Cells} The number of cells in a block for NAS

N_{col} Number of columns in a 2D MR image

N_{crops} Number of crops for multi-crop evaluation

N_d Number of dimensions of a multi-dimensional image

n_d Size of an image's third spatial dimension (depth)

N_{dc} Number of decay steps for learning rate decay

N_e Number of training epochs

N_{en} Number of models in an ensemble

n_f The number of time steps predicted by forecasting

n_h Size of an image's first spatial dimension (height)

N_λ Number of pixels for sampling an A-Scan

n_m Number of pixels taken into account by a local operator

n_p Parallel spin occupation number

n_{params} Number of trainable parameters in a deep learning model

N_r The number of regression targets

N_{sample} Number of architectures to sample by a controller for NAS

n_t Size of an image's temporal dimension

n_θ The size of the angle dimension in a polar image representation

n_w Size of an image's second spatial dimension (width)

$\mathcal{N}$ Normal distribution

o_{NN} A neural network's output

o_{t_i} Output of a recurrent neural network

ω Frequency of an excitation pulse

ω_0 Magnetic resonance frequency

$\Omega(w_M)$ A regularizer that penalizes trainable parameters

p_a Probability of sampling an architecture for NAS

p_d Dropout probability

P_M The performance of a machine learning model

p_r Freezing point inside a model for transfer learning

p_{shift} Probability of B-Scan being shifted due to simulated motion

p_{d_j} Zero padding in dimension j

ϕ A data augmentation function

Q_s Magnetic spin of a nucleus

R Reward function for reinforcement learning

r Stride of a convolution

R_{rot} Rotation matrix

r_{t_i} Output of a GRU's reset gate at time point t_i

$Regl$ Regularization method for image registration

s_c^i Spatial scale in a CNN architecture

σ Standard deviation

SSD_{reg} Sum of squared distances metric for image registration

$T1$ Longitudinal relaxation time

$T2$ Transversal relaxation time

T_B Temperature

t_i Time point i in a sequence of time points

T_M A machine learning task

t_{reg} Registration transformation

t_s Sampling time point

T_{t_i} A sequence of volumes (4D tensor) at time t_i

TE Echo time

θ Angle coordinate in a polar image representation

θ_E Angle of the excitation pulse

TR Repetition time

u_d Displacement field

u_{v_j} Maximum number of receptive fields v in dimension j

$\mathcal{V}$ A dataset of image volumes

v Local receptive field of a neural network

v_m Smoothing factor for mean filtering

V_{t_i} A sequence of 2D slices at time t_i

var Variance

W Weight matrix of a neural network

W_0 Gaussian beam's waist size

λ_{i-1} Weighting factor for temporal regularization

w_M A neural network's learnable weights

x Cartesian coordinate along the first image dimension

x_c Coordinate vector

$\mathcal{X}$ A dataset containing data and target examples

y Cartesian coordinate along the second image dimension

y Target example from a dataset $\mathcal{X}$

$\hat{\text{y}}$ Prediction of machine learning model

Y_{t_i} A target matrix at time point t_i from a dataset $\mathcal{X}$

z_{max} OCT's maximum imaging depth

z_o Rayleigh range

z_{t_i} Output of a GRU's input gate at time point t_i