Physics of Fluids

RESEARCH ARTICLE | OCTOBER 22 2026

Conditional neural field-based reduced order model for dynamic ditching load prediction

Henning Schwarz ; Pyei Phyo Lin ; Jens-Peter M. Zemke ; Thomas Rung

Check for updates

Physics of Fluids 38, 108124 (2026)
https://doi.org/10.1063/5.0342785

View Online Export Citation

Articles You May Be Interested In

Ditching characteristics of a seaplane under various wave conditions and effects of biomimetic floats
Physics of Fluids (September 2024)

One-way flow of a rarefied gas induced in a channel with a periodic temperature distribution
Physics of Fluids (August 1996)

Cavitation and ventilation modalities during ditching
Physics of Fluids (May 2019)

28 October 2026 11:51:29

AIP Publishing

Physics of Fluids

Special Topics Open for Submissions

Learn More

AIP Publishing

Physics of Fluids ARTICLE pubs.aip.org/aip/pof

Conditional neural field-based reduced order model for dynamic ditching load prediction

Cite as: Phys. Fluids **38**, 108124 (2026); doi: 10.1063/5.0342785
Submitted: 8 May 2026 · Accepted: 23 September 2026 ·
Published Online: 22 October 2026

View Online Export Citation CrossMark

Henning Schwarz,[1,a)] Pyei Phyo Lin,[1] Jens-Peter M. Zemke,[2] and Thomas Rung[1]

AFFILIATIONS

[1]Institute for Fluid Dynamics and Ship Theory, Hamburg University of Technology, Am Schwarzenberg-Campus 4, D-21073 Hamburg, Germany
[2]Institute of Mathematics, Hamburg University of Technology, Am Schwarzenberg-Campus 3, D-21073 Hamburg, Germany

[a)]Author to whom correspondence should be addressed: henning.schwarz@tuhh.de

ABSTRACT

Grid-based neural networks such as convolutional autoencoders are widely used in dimension reduction-based surrogate models for computational fluid dynamics. Recently, the use of coordinate-based approaches like conditional neural fields (CNFs) has emerged. Their independence of the spatial discretization is an advantageous feature and a key motivation for computational fluid dynamic applications. This paper discusses the spatiotemporal prediction of aircraft ditching loads using a CNF approach. The model employs a Fourier feature mapping of input coordinates to capture high frequency content. Based on feature-wise linear modulation, the output is conditioned on a latent vector, which is learned by autodecoding. The model is evaluated using two datasets for the dynamic loads on the fuselage of a German Aerospace Center (DLR) D150 aircraft. The first dataset relates to a single fixed spatial discretization, and the second dataset includes data from different discretizations. On the first dataset, the model is compared to a grid-dependent convolutional autoencoder-based model, and both approaches achieve similar reconstruction accuracy, but the CNF requires significantly fewer parameters. When paired with a long short-term memory network in the latent space, both approaches achieve similar spatiotemporal prediction accuracy. Results for the second dataset demonstrate the ability of the CNF-based approach to reconstruct ditching loads accurately for heterogeneous spatial discretizations. In particular, the model can generalize to discretizations not used for training. This allows for flexible use of training datasets generated for different geometries and/or discretizations and the use of the surrogate model to predict loads for different configurations.

© *2026 Author(s). All article content, except where otherwise noted, is licensed under a Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).* https://doi.org/10.1063/5.0342785

28 October 2026 11:51:29

I. INTRODUCTION

Machine learning (ML) has increasingly been applied in the fluid dynamics community over the last decade with great success in a variety of areas, with reduced order modeling being one of them. Next to classical approaches like the proper orthogonal decomposition (POD),[1] surrogate models based on (variational) autoencoders have been studied on a wide range of applications,[2–9] whereby models such as long short-term memory (LSTM)[10] or transformer networks[11] were applied in the latent space to even predict temporal dynamics.[3,7,12,13]

In many previous works, convolutional autoencoders (CAEs) are used as they can efficiently capture local dependencies inherent in the physical data. Although CAE-based models are effective, they require the same (spatial) discretization for all data examples in the dataset. Therefore, further data processing is required for grids that differ in the examples of the dataset. Meta grids can be employed in this regard[4,9,14] but involve additional effort and difficult-to-control errors. Furthermore, the possibility of flexibly using the ML model for different configurations would be appreciated in engineering practice. Consequently, there is strong motivation to train surrogate models directly on unstructured data, i.e., heterogeneous grids. Several works address this issue with graph neural networks (GNNs).[15–18] Although these might work well in combination with different grids, there are also known issues, e.g., the difficulties of generalizing to meshes not seen during training,[19] which presents a major hurdle in engineering practice.

Recently, the use of neural fields for surrogate modeling has emerged. Neural fields are coordinate-based networks, i.e., they map the spatial (and potentially temporal) coordinates of one point to a specific output quantity at this point.[20] They are therefore independent of the discretization, which is a highly useful feature for computational fluid dynamics surrogate models. In neural field-based

© Author(s) 2026

surrogate models, the output is often conditioned to a provided or learned latent vector to enable considering different cases with a single model, similar to autoencoder-based models. An autodecoding approach[21] is usually employed to learn the latent vector, and the model is referred to as a conditional neural field (CNF). To address the challenge that neural networks struggle to capture high frequency content,[22] CNFs frequently make use of a sinusoidal representation network (SIREN).[23] Alternatively, the input coordinates can be transformed with a Fourier feature mapping (FFM), which extends the random Fourier features originally introduced by Rahimi and Recht.[24] Tancik *et al.*[25] theoretically showed their effectiveness for coordinate-based neural networks. CNFs have been applied in several works for fluid dynamics problems. For example, the DINo method[26] combines a CNF, which reduces the dimension of the input data, and a neural ODE, which acts in the learned lower-dimensional latent space, for spatiotemporal predictions of solutions to partial differential equations. The CORAL method[19] combines two CNFs, representing input and output functions, and a latent space processor for operator learning on general geometries. In the low-dimensional latent space, the processor maps the learned input representation to the learned output representation. The more recent CoNFiLD method employs a CNF combined with a latent diffusion model to generate spatiotemporal turbulence.[27] Guo *et al.* benchmark CNFs with different conditioning mechanisms against POD and CAEs for dimension reduction of turbulent flows.[28] Catalani *et al.* use CNFs in their proposed MARIO framework for aerodynamic surrogate modeling.[29] They use available operating and geometric conditions directly to define the latent vector. If the dataset has geometric variability without access to the corresponding true parameters, the part of the latent vector providing the geometric information is optionally learned.

Connections between CNFs and neural operators, which have achieved state of the art performance in operator learning,[30] have been reported by Wang *et al.*[31] They demonstrate that several neural operator architectures can in fact be seen as CNFs. Depending on the architecture, the conditioning is local and/or global. By using the CNF interpretation, they also suggest an alternative implementation for the frequently used Fourier neural operator (FNO) that does not rely on regular grids, as in the standard setting, but on arbitrary points. Moreover, Catalani *et al.*[29] note that their proposed MARIO framework is a generalization of the DeepONet architecture.[32] A similar observation has been made for the neural implicit flow framework,[33] which combines a neural field with a hypernetwork processing external conditions for dimensionality reduction of spatiotemporal data. Mind that physics-informed neural networks (PINNs)[34] are neural fields as well, since they process point coordinates and map them to a certain physical quantity. They are especially useful when training data are not available.

This paper is part of ongoing research for the development of data-driven surrogate models for the prediction of aircraft ditching loads and deformations. Ditching refers to emergency landing on water. Certification of a commercial aircraft requires ditching analysis, which can be based on simulations. A one-way coupling of the fluid–structure interaction is usually employed due to the high computational effort of the structural analysis.[8,35] To enable a two-way coupling while preserving computational speed, the long-term goal of this research is to include ML-based approximations of the deformations during the load analysis. First steps using ML to predict deformation-inducing ditching loads were investigated in previous works: Loads on the fuselage of a generic German Aerospace Center (DLR) D150 aircraft were predicted using CAE-LSTM combinations.[8] Subsequently, the interpretability was improved by disentangling the latent space with a probabilistic β-variational autoencoder and deterministic alternatives.[36] All examples of the datasets used in both previous studies are based on an identical discretization of the same fuselage. This discretization is allowed to be more flexible in this work. For this purpose, a CNF approach for dimensionality reduction is investigated, in which the latent variables are obtained using an autodecoding strategy. The model is initially compared to a CAE-based model on previously used datasets. Subsequently, the model is trained on a dataset containing results obtained from different discretizations. The ultimate goal of using a discretization independent approach is to consider different aircraft geometries with different discretizations within a single model. As a first step, we employ a single aircraft geometry with different discretizations in this work, which already offers considerably greater flexibility than the CAE-based dimensionality reduction employed in earlier studies.

The remainder of the paper is structured as follows: The CNF architecture and the training procedure are presented in Sec. II. In Sec. III, results on the two datasets are discussed, and finally, conclusions are drawn in Sec. IV.

II. CONDITIONAL NEURAL FIELD

Neural fields predict the output for one coordinate at a time.[20] This way, they learn a continuous function on the coordinates, which is parameterized by the network parameters. To consider multiple data samples, the network can be conditioned to a latent vector using an autodecoder architecture.[21] Different from an autoencoder, an autodecoder does not contain an encoder. The latent vectors are therefore considered to be trainable parameters that are optimized during training. The learned latent vectors can then be used for downstream tasks similar to those in autoencoder-based models, e.g., to predict the temporal load evolution.

While this encoder-less construction is independent of the discretization and the numbers of points used for each sample, it also requires the optimization of the latent vector of a test sample at inference time.[21] As a result, the time for encoding a test sample is usually several orders of magnitude longer compared to autoencoder architectures. In contrast, an advantage of this construction is that the output does not have to be calculated for the entire field if only individual points are of interest—although the related computational efforts are often marginal in practice.

A. Conditioning

Conditioning allows a neural field to handle multiple different snap shots that are not necessarily obtained for the same discrete geometry. There are different implementation strategies for conditioning.[20] A simple form of conditioning is the concatenation of the latent vector and the input coordinates. In the DeepSDF framework,[21] this concatenation is the input to the first hidden layer, and it is also connected to an intermediate layer with a skip connection. The conditioning can be performed in a more complex way with a hypernetwork[37] that predicts the parameters of the neural field based on the latent vector,[23,26,33,38] or with the feature-wise linear modulation

28 October 2026 11:51:29

© Author(s) 2026

(FiLM).[39] For the latter, the latent vector is transformed and fed into multiple layers of the network. Consider a layer output $\mathbf{x} \in \mathbb{R}^d$ and a latent vector $\mathbf{z} \in \mathbb{R}^m$. For functions $\boldsymbol{\gamma}, \boldsymbol{\beta} : \mathbb{R}^m \mapsto \mathbb{R}^d$, FiLM corresponds to an affine linear mapping,

$$\boldsymbol{\gamma}(\mathbf{z}) \odot \mathbf{x} + \boldsymbol{\beta}(\mathbf{z}), \tag{1}$$

where $\boldsymbol{\gamma}$ and $\boldsymbol{\beta}$ scale and shift $\mathbf{x}$, respectively.

Similar to,[19,27,29] we employ FiLM in this work. Note that these three works only use the shift modulation $\boldsymbol{\beta}$. When $\boldsymbol{\beta}$ is a linear layer, this also corresponds to concatenating the layer output $\mathbf{x}$ and $\mathbf{z}$.[40] For the CORAL framework, the authors report that using only the shift modulation led to more stable training and the same performance as using both shift and scale.[19] In our experiments, both options returned similar results as well. Therefore, we also only apply the shift modulation in this work.

B. Fourier feature mapping

Neural networks are known for having a spectral bias, i.e., difficulties to learn high frequency content.[22] This issue can be addressed by using a sinusoidal representation network (SIREN),[23] where periodic activation functions are applied. This approach is followed by the CORAL[19] and CoNFiLD[27] methods as well as the neural implicit flow framework.[33] Alternatively, a positional encoding of the coordinates via a Fourier feature mapping (FFM) can be employed, which allows the network to learn different frequency components independently. Using neural tangent kernel (NTK)[41] theory, Tancik *et al.*[25] have shown that using an FFM overcomes the spectral bias and makes learning high frequency content easier. The mapping is defined as

$$\boldsymbol{\zeta}(\mathbf{v}) = [\cos(2\pi \mathbf{B}\mathbf{v}),\ \sin(2\pi \mathbf{B}\mathbf{v})]^T, \tag{2}$$

where $\mathbf{v} = (x, y, z)^{\mathrm{T}}$ is the vector of input coordinates and the entries of $\mathbf{B} \in \mathbb{R}^{b \times 3}$ are sampled from $\mathcal{N}(0, \sigma^2)$ in this work. The dimension b and the standard deviation σ are hyperparameters whose values are given in Sec. II D.

Like the MARIO framework,[29] our model employs the FFM approach in this work. To illustrate the benefits of FFM, training and validation loss curves for the case where the FFM was not used have been added to the Fig. 4 in Sec. III.

C. Full model

The full model is illustrated in Fig. 1. For activation function α, the input coordinate vector $\mathbf{v}$ and latent vector $\mathbf{z}$ are mapped to the pressure load prediction $\tilde{p}(\mathbf{z}, \mathbf{v})$ via

$$\begin{aligned} \mathbf{h}_0 &= \boldsymbol{\zeta}(\mathbf{v}), \\ \mathbf{h}_i &= \alpha(\mathrm{Linear}(\mathbf{h}_{i-1}) + \boldsymbol{\beta}_i(\mathbf{z})), \quad 1 \le i \le N, \\ \tilde{p}(\mathbf{z}, \mathbf{v}) &= \mathrm{Linear}(\mathbf{h}_N). \end{aligned} \tag{3}$$

We implement the shift modulation $\boldsymbol{\beta}$ as a linear layer and use $\alpha =$ LeakyReLU with a slope parameter of 0.01 in this work. Similar to the DeepSDF model,[21] we apply weight normalization.[42] Specifically, we apply it to the layers of the main network, i.e., the Linear($\mathbf{h}_{i-1}$) layers. We use $N = 6$ FiLM layers. These are of dimension d, for which different values are investigated in Sec. III A. Our CNF computes latent representations for each time step of a simulation. Thus, only the spatial and not the temporal coordinates are used as input to the CNF. This is consistent with the previously employed CAE-based models.[8,36]

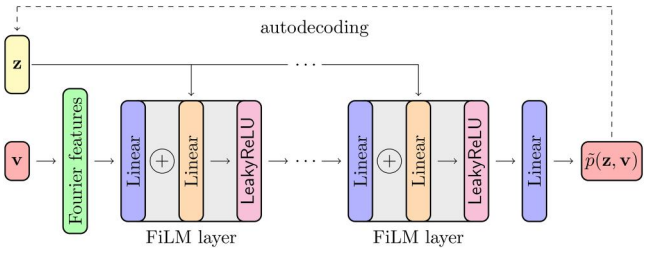


FIG. 1. Scheme of the conditional neural field architecture.

D. Training

The model is trained with the mean squared error (MSE),

$$\mathrm{MSE}(\mathbf{p}, \tilde{\mathbf{p}}) = \frac{1}{n} \|\mathbf{p} - \tilde{\mathbf{p}}\|_2^2, \quad \mathbf{p} \in \mathbb{R}^n, \tilde{\mathbf{p}} \in \mathbb{R}^n, \tag{4}$$

where $\mathbf{p}$ and $\tilde{\mathbf{p}}$ are the true and reconstructed pressure values of a minibatch containing a total of n spatial points.

The output values as well as the input coordinates are each normalized to the interval [0, 1] using min–max normalization. The latent variables are initialized from $\mathcal{N}(0, 0.1^2)$. In line with the previous work,[8,36] we employ a latent space dimension of $m = 10$. Regarding the FFM hyperparameters, we set $b = 128$. Too small [large] values of the standard deviation σ can lead to underfitting [overfitting].[25] In this work, $\sigma = 10$ provided satisfactory results. The CNF is trained for 1500 epochs. For each of the two parameter groups, network parameters and latent vectors, an individual optimizer is defined, and both are updated per minibatch. To this end, we use the Adam optimizer[43] with a learning rate of $2 \cdot 10^{-3}$ for both parameter groups. The minibatch size is set to 32. At inference time, the latent vectors for test examples are optimized while keeping the network parameters fixed.

Different learning strategies are conceivable, e.g., the meta-learning strategies used in,[19,29] based on the work of Zintgraf *et al.*[44] In this approach, the latent vectors are updated in an inner loop during training, and the network parameters are updated subsequently in an outer loop using second-order gradients. At inference time, this has the advantage that the latent vectors only have to be optimized for the same number of steps that are used in the inner loop during training. However, the computational cost for training is higher due to the second-order gradients. We initially also performed experiments using the meta-learning approach but could not reach the same reconstruction quality as for the simpler first-order approach. Thus, we follow the first-order approach here, similar to of Park *et al.*[21] and Yin *et al.*,[26] in which the number of epochs for the optimization of latent vectors at inference time is on the order of hundreds.

Similar to,[21,26,29] we randomly sample a subset of spatial points for each example in a minibatch, containing 2000 spatial points. The error on the validation set cannot be computed with a forward pass like in ordinary neural networks, as the latent vectors have to be optimized. Optimizing the latent vector for all examples in the validation set in each epoch would therefore significantly prolong the training

28 October 2026 11:51:29

© Author(s) 2026

time. For that reason, the model and the latent vectors of the training set are stored at a few selected checkpoints throughout the training. The validation error is computed on all checkpoints after the training. Different from the training set, we use all available spatial points for the optimization of the latent vectors of the validation set, as this leads to a more stable optimization in our experiments. They are individually optimized for 200 epochs using Adam with a learning rate of 10^{-1}. In our experiments, smaller learning rates often resulted in larger reconstruction errors. The model is trained on two NVIDIA GeForce RTX 3090 graphics processing units (GPUs) in PyTorch.[45] For validation and testing, a single GPU is employed.

III. RESULTS

The CNF model was trained and tested using two datasets containing simulated emergency water landing loads on the fuselage of a D150 aircraft. The data were obtained from simulations using different horizontal and vertical approach velocities to the free water surface. For this purpose, the in-house 3-degrees-of-freedom method *ditch*[46] was used, which considers the dynamics of horizontal, vertical, and pitch motion. Further details on data generation and processing can be found in.[8] Figure 2 illustrates a view of the entire fuselage from below colored by contours of the normalized instantaneous pressure.

The aim of this study is twofold: first, to analyze the predictive capability of the CNF model in time-dependent dynamic processes (dataset A), and second, to analyze its predictive capability for heterogeneous spatial discretizations in the training and test data (dataset B). To reduce effort and complexity, the dynamic processes are analyzed using uniform spatial discretizations, and the influence of varying spatial discretizations is investigated without considering dynamic aspects.

For the first dataset, labeled dataset A, the uniform spatial discretization was already employed in two previous studies by the authors.[8,36] In these works, CAE-LSTM combinations were compared to Koopman-based autoencoders for the spatiotemporal prediction of ditching loads,[8] and the interpretability was improved by disentangling the latent space.[36] The reconstruction accuracy as well as the ability to perform spatiotemporal predictions when paired with an LSTM network in the latent space are assessed. To this end, the results are compared to those obtained with a CAE to verify that the CNF can achieve similarly accurate results.

For the second dataset, labeled dataset B, different discretizations are employed to demonstrate that the CNF can still reconstruct ditching loads accurately when trained on varying discretizations. This ability is the main benefit of using the CNF. As mentioned above, the second study is confined to static reconstructions, i.e., the temporal evolution is not considered.

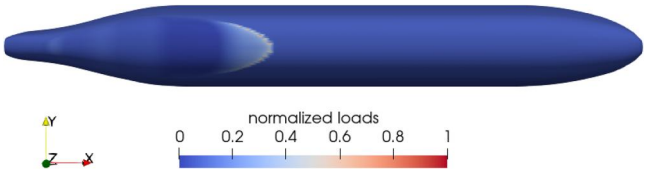


FIG. 2. Exemplary normalized pressure loads on the D150 fuselage simulated with *ditch*.

A. Dataset A

The dataset contains a total of 373 ditching simulations with a D150 aircraft fuselage. These simulations differ in the initial horizontal and vertical velocities at the time of the first water impact, which range between approximately 67 and 87 m/s for the horizontal velocity, and 0.6 and 3.9 m/s for the vertical velocity, respectively.

Each simulation consists of around 18–35 equidistant time steps of 0.1 s, which are represented by 128×128 sized images. These represent a portion of the fuselage, which in total is discretized using 150×171 spatial points. The two spatial dimensions refer to the longitudinal (x) and circumferential coordinates. The circumferential coordinates contain contributions of both y- and z-coordinates and therefore three input coordinates (x, y, z) to the CNF are considered.

The training set comprises 323 simulations, the distribution of the velocity pairs is shown in Fig. 3.

For validation, data from 20 simulations are used and a test set containing results from 30 simulations serves to assess the spatiotemporal prediction accuracy (cf. Fig. 8).

1. Dimension reduction

We compare results for different dimensions $d \in \{32, 64, 128\}$ of the CNF layers, cf. Sec. II C. Figure 4 depicts the reconstruction loss on the training and validation set for the different dimensions d.

The figure illustrates that a larger dimension d leads to lower errors on the training set. Although this is the case for the validation set as well, the differences between the validation errors are much smaller for the smaller dimensions d. This leads to the training and validation loss curves for $d = 32$ and $d = 64$ being much closer to each other compared to $d = 128$. For comparison, a CAE used in previous work[36] returns a validation error of $2.26 \cdot 10^{-6}$ when trained for 1500 epochs, cf. dashed horizontal line in Fig. 4. Note that in the previous work[36] the CAE-based models were only trained for 500 epochs, as satisfactory results were achieved at that point. To ensure consistency with the CNF, we extend the CAE training time to 1500 epochs in this work. Like the CNF, the CAE employs a latent space dimension of $m = 10$. The dashed horizontal line included for comparison is constant over the three panels and does not change for different CNF layer dimensions d. As indicated in the left panel, the validation

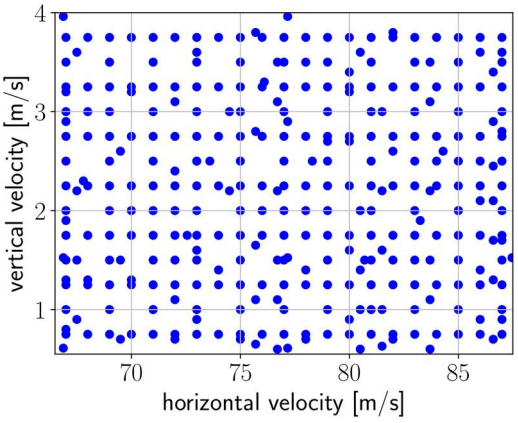


FIG. 3. Velocity pairs for the training set of dataset A.

28 October 2026 11:51:29

© Author(s) 2026

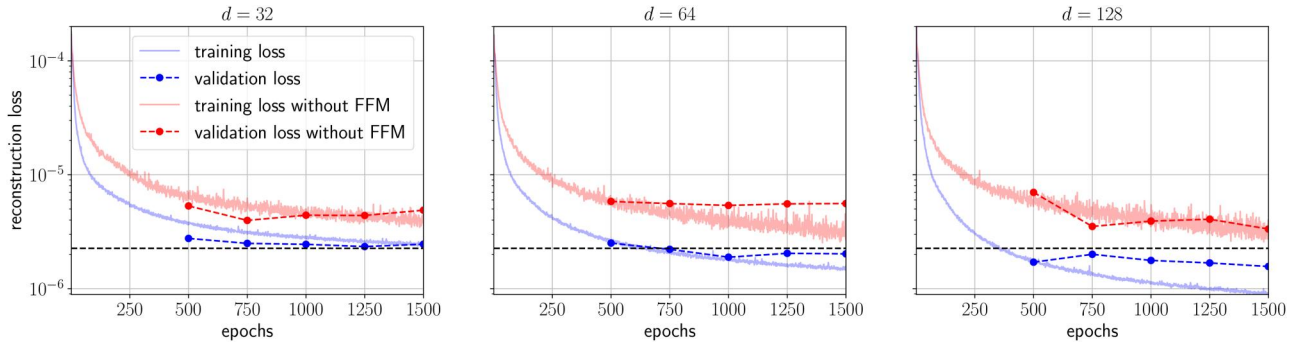


FIG. 4. Reconstruction loss on the training and validation sets of dataset A for different dimensions d of the CNF layers. The error on the validation set was computed on five checkpoints after the training. The dashed black line indicates the loss on the validation set for a CAE and is constant over all three panels. Furthermore, loss curves are shown for the case where the Fourier Feature Mapping (FFM) has not been used to outline the benefit of FFM.

error of the CAE is close to that of the CNF for $d = 32$, reaching a minimum value of $2.34 \cdot 10^{-6}$. Slightly lower validation errors than the CAE baseline are obtained for the two larger dimensions d, which return minimum validation errors of $1.89 \cdot 10^{-6}$ for $d = 64$ and $1.57 \cdot 10^{-6}$ for $d = 128$. To illustrate the benefit of FFM, Fig. 4 also includes results obtained without FFM that lead to a higher reconstruction loss under the same settings. As indicated by the dashed red lines, the validation error of the CNF without FFM is higher than that of the CAE for all tested dimensions d.

The training times, inference times for encoding and decoding, as well as the number of trainable parameters excluding latent vectors, are listed in Table I. The different dimensions d lead to fairly similar training times, although the largest value $d = 128$ takes around 8% more time than $d = 32$ and $d = 64$. In comparison to the CAE method, the training times for the CNF models are about eight times longer, with the CAE only being trained on one GPU and, like most neural networks, calculating the validation error already during the training process. The times for encoding test samples are close to 2 s for the three CNF models, with $d = 128$ leading to around 7% longer time than $d = 32$ and $d = 64$. The validation set includes a total of 494 time steps, computing the validation error therefore requires around 945 s to 1015 s for each checkpoint, depending on d. The CAE, however, takes around four orders of magnitude less encoding inference time. For decoding, the inference times are in the same order of magnitude for all models, with the CAE again requiring the least time. Note that when temporal predictions are performed in the latent space, typically only a few initial time steps need to be encoded, but many more (predicted) time steps need to be decoded. The inference speed of the decoder is therefore usually more relevant for the use of a surrogate model. Considering the number of trainable parameters, all CNF models have fewer than the CAE. While for $d = 128$, the CNF has only around 7% fewer parameters than the CAE, $d = 64$ [$d = 32$] leads to around 69% [88%] fewer parameters than the CAE. The memory efficiency of CNFs compared to grid-based methods has also been pointed out by Catalani *et al.*[29] When the CNF has a similar amount of parameters as the previous CAE[36] for $d = 128$, a lower reconstruction error is obtained. Similarly, significantly fewer parameters are required to reach the same reconstruction quality. This comes, however, at the cost of longer training and inference times.

2. Spatiotemporal predictions

Unlike most previous engineering studies, this section aims to evaluate the performance of CNF in combination with an LSTM network in the latent space for predicting temporal evolutions. The LSTM structure is based on the previous works,[8,36] and comprises two LSTM layers with hidden state dimensions of 100, a linear output layer with $m = 10$ neurons, and uses a time delay of three input time steps. The network is again trained using the Adam optimizer with a learning rate of 10^{-3} and a minibatch size of 512 for 500 epochs.

To assess the accuracy of the spatiotemporal predictions, we use the same average error as in our previous studies.[8,36] To this end, the models are trained five times to account for the randomness in the training process. From each trained CNF version, we use the latent vectors from the checkpoint with the lowest validation error to train

TABLE I. Training times and inference times for encoding and decoding for different dimensions d of the CNF in comparison to a previously used CAE.[36] The CNF models are trained on two NVIDIA GeForce RTX 3090 GPUs. The CAE training time refers to 1500 epochs on a single NVIDIA GeForce RTX 3090 GPU and includes the calculation of the error on the validation set. Inference times are reported for a single GPU in all cases.

Model	Training time (s)	Encoding inference time (s)	Decoding inference time (s)	Trainable parameters
CNF(32)	6595	1.91	0.000 69	15 841
CNF(64)	6525	1.92	0.000 69	41 921
CNF(128)	7030	2.05	0.000 72	124 801
CAE	870	0.000 41	0.000 39	134 731

28 October 2026 11:51:29

© Author(s) 2026

the LSTM network, e.g., for the first version, the epochs 1250, 1000, and 1500 of the results shown in Fig. 4 for CNF layer dimensions of 32, 64, and 128, respectively. We note that the validation errors obtained on the five checkpoints are not exactly the same over all five runs. As the reconstruction errors on the validation set shown in Fig. 4 for the stationary analyses no longer decrease significantly after epoch 500, especially for $d = 32$ and $d = 128$, we also report the results obtained from training the models for only 500 epochs for comparison. To this end, predictions are made on all 30 test cases with each of the three trained CNF variants and the trained CAE. Normalized root mean squared errors (RMSEs) are computed for each test case, and the RMSEs are subsequently averaged over the five trained versions to obtain time-dependent errors for each test case. These are then time-averaged for each test case, and finally, an overall mean is calculated across all test cases. The overall average errors are shown in Table II.

Although the reconstruction error of the CNF models is similar or lower than the CAE depending on the dimension d, the CNF-LSTM combination returns a slightly higher error level than the CAE-LSTM combination. A smaller CNF layer dimension d generally leads to a lower average error in this setting, while obtaining higher reconstruction errors in Fig. 4 when the temporal evolution is not considered. For $d = 64$, the performance after 1500 epochs is similar to the uncorrelated autoencoder approach shown in our previous study,[36] which corresponds to a CAE that aims to provide disentangled latent variables with a modified loss function. It is important to note that the CNF for $d = 128$ still performs better than the worst autoencoder models tested in previous studies.[8,36]

When the LSTM is trained with the CNF latent vectors corresponding to epoch 500, the average errors reported in Table II remain nearly unchanged for $d = 32$ and $d = 128$, suggesting that the small training improvements observed after epoch 500 in Fig. 4 do not have a significant influence on the time series prediction anymore. The training time could therefore be reduced to around $1/3$ of the time reported in Table I to achieve approximately the same accuracy for time series prediction. For $d = 64$, however, the value increases slightly to 0.023. In the previous study, the CAE-LSTM combination also yielded a slightly higher average error value of 0.020 after 500 epochs of CAE training.[36]

Setting the standard deviation σ in the FFM in Eq. (2) too small [large] can lead to oversmoothed [noisy] results.[25] It can be expected that this also affects the obtained latent space. To investigate the related influence on the spatiotemporal predictions, we also performed experiments with a smaller value of $\sigma = 1$ instead of $\sigma = 10$, cf. Table III. This led to a small reduction of the average errors for $d = 32$ and $d = 128$, i.e., to 0.020 and 0.022 after 500 epochs of the CNF, respectively. For $d = 32$, this approximately corresponds to the results obtained with a CAE after 500 epochs. Values of 0.021 and 0.022 were obtained using the same procedure as for Table II, i.e., using the epochs, where the lowest validation error is obtained over 1500 epochs. For $d = 64$, however, this also resulted in small increments to 0.024 after 500 epochs and to 0.023 using the epochs where the lowest validation error is obtained over 1500 epochs. The experiments therefore did not provide a clear indication on the influence of σ on the accuracy of the spatiotemporal predictions, at least not in the parameter range studied.

The fact that the larger CNF layer dimensions tested, $d = 64$ and $d = 128$, yield smaller static reconstruction errors than the smaller layer dimension tested, $d = 32$, but exhibit larger spatiotemporal prediction errors when combined with an LSTM, could be attributed to the different learned latent spaces. It is conceivable that the latent space learned with $d = 32$ is smoother and therefore enables more accurate predictions by the LSTM, while $d = 64$ and $d = 128$ offer greater capacity, allowing finer details to be captured, which reduces the reconstruction error but may also diminish the regularity of the latent space. However, a detailed assessment of the CNF latent space would go beyond the scope of this work and could be investigated in future studies.

To assess the performance of the CNF- and CAE-based models on the 30 individual test cases, Fig. 5 compares the average errors obtained for the CNF-LSTM and CAE-LSTM combinations, achieving the lowest overall average error, i.e., the CNF with $d = 32$ and $\sigma = 1$ trained for 500 epochs, and the CAE trained for 1500 epochs. While the figure confirms that the CAE model delivers (slightly) better performance, it also shows that the CNF model yields lower average errors in several cases. Moreover, the results from both models follow a trend that was already observed in our previous work:[8] Test cases with lower vertical velocities are generally more difficult to predict. Without going into further detail, we note that the trend can also be observed with the other CNF settings. For the CNF setting considered in the figure, particularly large errors are returned for the test case with horizontal and vertical initial velocities of 70.4 and 1.0 m/s, respectively.

B. Dataset B

The CNF is now trained using a dataset containing simulations with three different spatial discretizations of the same D150 hull. The training data of the second dataset are therefore not uniformly based on 150×171 discrete points, as is the case with dataset A, cf.

TABLE II. Average errors obtained by the different models combined with the same LSTM network for spatiotemporal predictions on 30 test cases. Results are obtained using $\sigma = 10$ in the FFM for the CNF. Second column refers to CNF/CAE checkpoints with the lowest validation error over 1500 epochs. Third column refers to CNF/CAE after 500 epochs.

Model	1500 epochs	500 epochs
CNF(32) – LSTM	0.021	0.021
CNF(64) – LSTM	0.022	0.023
CNF(128) – LSTM	0.023	0.023
CAE – LSTM	0.019	0.020

TABLE III. Average errors obtained by the different models combined with the same LSTM network for spatiotemporal predictions on 30 test cases. Results are obtained using $\sigma = 1$ in the FFM. Second column refers to CNF checkpoints with the lowest validation error over 1500 epochs. Third column refers to CNF after 500 epochs.

Model	1500 epochs	500 epochs
CNF(32) – LSTM	0.021	0.020
CNF(64) – LSTM	0.023	0.024
CNF(128) – LSTM	0.022	0.022

28 October 2026 11:51:29

© Author(s) 2026

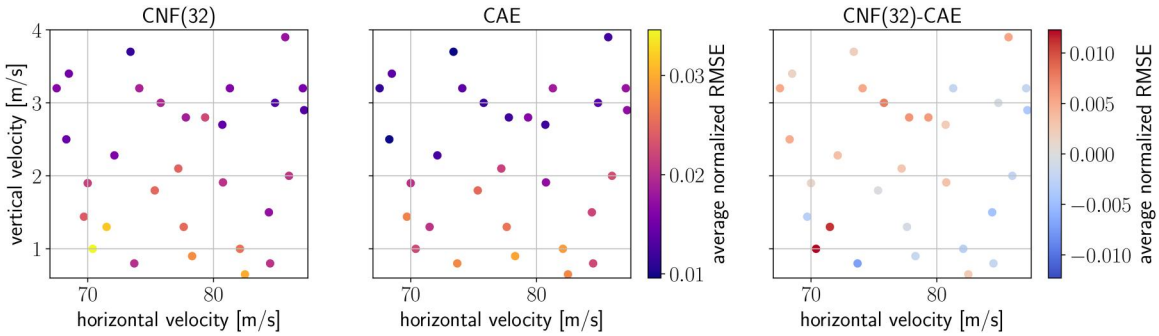


FIG. 5. Velocity pairs for the test set colored by obtained best accuracy. Color refers to average errors to compare the best performing CNF ($d = 32$, $\sigma = 1$, after 500 epochs, left) and the CAE (1500 epochs, center) combined with the same LSTM network. The right panel displays the difference of the average errors, where red [blue] colors indicate larger errors by the CNF [CAE].

Sec. III A. As in the first dataset, the number of points in the circumferential direction remains constant at 171. However, in the longitudinal direction, three different discretizations are used, based on 129, 150, and 170 discrete points, respectively. Each of these three longitudinal discretizations contributes to the training set with 100 simulations. The simulations again differ in the initial horizontal and vertical velocities. The training data space essentially corresponds to the first dataset, and the distribution of the velocity pairs used for the simulations is depicted in Fig. 6, where the different colored shapes represent different subsets. Mind that the three subsets do not correspond to exactly the same distribution of velocity pairs as the dataset used in Sec. III A, and the graph does not account the differences between the spatial discretizations. However, the goal of this section is not to compare the accuracies of the trained models on both datasets but rather to demonstrate that the CNF can naturally handle the second dataset and obtain satisfactory reconstruction quality. The validation set contains another 10 simulations for each discretization, and the input parameters are included in the operating range of the training set. We consider the same portion of around 85% of the total longitudinal coordinates as in Sec. III A, referring to 110, 128, and 145 considered points, respectively. To reduce training and inference time on the second dataset, we consider a smaller portion of the circumferential coordinates than in Sec. III A, i.e., the first 70 circumferential coordinates. Typically, circumferential coordinate positions beyond the 70th entry remain unwetted during the first impact and therefore very rarely contribute to the peak loads. In line with that, we also reduce the number of considered spatial points per example in a minibatch to 1000 during training.

Since the CAE algorithm cannot be applied to this dataset without further data processing, we restrict ourselves to the CNF and use the lowest tested hidden dimension $d = 32$ from Sec. III A and $\sigma = 10$ for the FFM. Although $d = 64$ and $d = 128$ returned lower static reconstruction errors in Sec. III A, we use $d = 32$, because the reconstruction accuracy, in combination with $\sigma = 10$, was close to that of the CAE and thus satisfactory, while at the same time having the smallest number of trainable parameters.

Figure 7 displays the loss on the training and validation sets. As seen in Sec. III A, the training and validation errors are again close together, suggesting that CNF can also generalize to unseen data for this dataset.

1. Generalization to unseen discretizations

We now test the model on discretizations that were not used during training. For this purpose, a test set composed of loads from

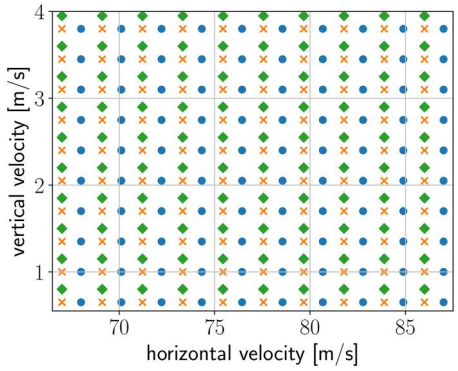


FIG. 6. Velocity pairs for the training set of dataset B. Blue circles, orange crosses, and green diamonds refer to subsets with 129, 150, and 170 points in longitudinal direction, respectively.

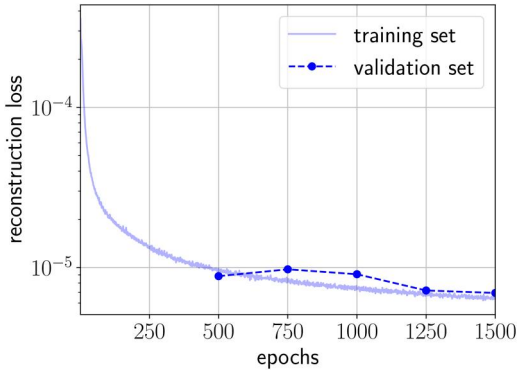


FIG. 7. Reconstruction loss obtained for the CNF with dimension $d = 32$ on the training and validation sets of dataset B. The error on the validation set was computed on five checkpoints after the training.

28 October 2026 11:51:29

© Author(s) 2026

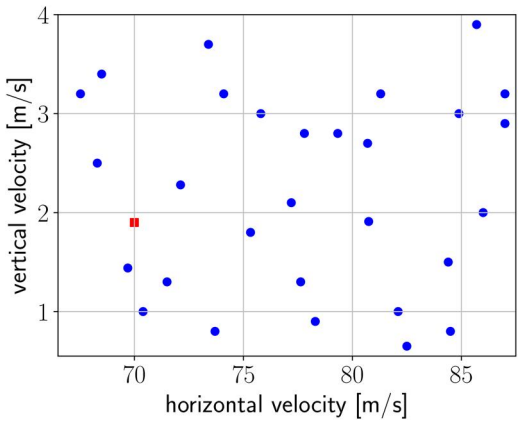


FIG. 8. Velocity pairs for the test sets. Red square refers to the configuration, for which reconstructions of dataset B are shown in Fig. 10.

four different discretizations is used. Specifically, 121, 139, 159, and 179 points in longitudinal direction are tested. Note that 121 and 179 points lie outside the range used for training while 139 and 159 lie inside. To evaluate the performance of the CNF, we again consider around 85% of the longitudinal coordinates, i.e., 103, 119, 136, and 153 points, respectively. For each discretization, 30 simulations are performed using the same input parameters for each discretization. These are again included in the operating range of the training set and essentially correspond to the test set in Sec. III A. The parameters are depicted in Fig. 8. The red colored parameter combination is employed for a qualitative assessment of the reconstruction quality returned by the different discretizations at the end of the section, cf. Fig. 10. For comparison, we also perform these simulations for the three discretizations that were used during training. As for the previous simulations, a maximum time span of 7 s is considered for each test case.[8] The period under consideration begins with the first impact, followed by a load intensive initial phase, and ends when the stresses fall below a lower threshold. The total duration and the number of considered time steps depend on the approach conditions.

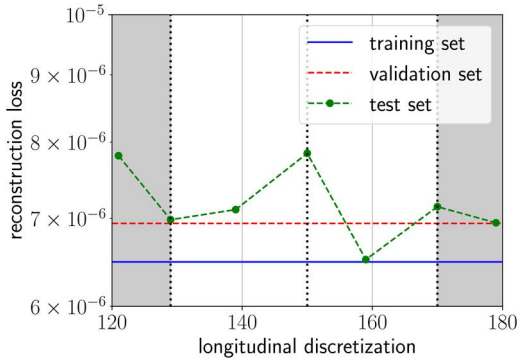


FIG. 9. Reconstruction loss obtained for the CNF with dimension $d = 32$ on the test set of dataset B. The errors on the training and validation sets refer to epoch 1500 in Fig. 7. The error on the test set is shown for each tested discretization. Dotted black lines indicate discretizations used during training. Shaded area indicates discretizations outside of range used during training.

Simulations of ditching typically do not yield significant differences in results for reasonable variations in the number of longitudinal cross sections used, although the different discretizations can lead to slightly different load histories, which can also influence the total duration and amount of time steps. Specifically, the 30 test cases result in a total of 807, 807, 810, 803, 792, 792, and 787 time steps for the longitudinal discretizations using 121, 129, 139, 150, 159, 170, and 179 points, respectively. This corresponds to a maximum difference of below 3%. Results displayed in Fig. 7, motivate the employment of the model from epoch 1500.

Figure 9 depicts the reconstruction errors on the test sets for each discretization. The errors on the training and validation sets are also displayed for comparison. In general, all errors are close together and within the range of what the studies shown above revealed. The largest error is obtained for the longitudinal discretization using 150 points, followed by that with 121 points. The latter is outside of the range used for training, while the former was used for training. Slightly lower errors are obtained for the discretizations with 170 and 139 points, followed by the ones using 129 and 180 points, respectively, which are similar to the validation error. The lowest error is returned for the discretization using 159 points and is close to the error on the training set. The discretization was not used during training but is inside the range of those used during training. However, there is no obvious correlation between the error level and the type of discretization, i.e., used or not used during training, or between error levels and the number of points in the longitudinal direction. Moving further outside of the training range, the errors grow up to $1.06 \cdot 10^{-5}$ for a discretization with 98 points and to $7.85 \cdot 10^{-6}$ [$8.23 \cdot 10^{-6}$] for a discretization with 197 [221] points.

Figure 10 shows the pressure loads for exemplary time steps using the four tested discretizations that were not used during training, as well as their reconstructions for the respective discretizations. The depicted time steps refer to the 1st, 10th, and 24th time steps and cover three typical phases of a ditching process for an exemplary simulation with an initial horizontal and vertical velocity of 70 and 1.9 m/s, respectively, cf. Fig. 8. After the initial time step, in which the fuselage first touches the water, the wetted impact area widens at the 10th time step. During the final depicted time step, loads are primarily acting at the rear end of the fuselage. Mind that the same trained CNF is used for all four cases. The results shown in the figure are of similar quality at the other test points and strongly support the hypothesis that the CNF can accurately reconstruct the loads across different discretizations. The obtained pressure fields are quite similar for the different discretizations in the present example. Nonetheless, the ability to reconstruct the different fields, which are associated with different input coordinates, using a single model is a highly valuable feature. Even with the 121-point discretization, which has the second largest error in Fig. 9, the quality of the reconstructions are satisfactory.

IV. CONCLUSION

This paper presents a conditional neural field (CNF)-based surrogate model for ditching load prediction. Compared to a previously employed dimension reduction with a convolutional autoencoder (CAE) approach, the CNF approach is coordinate-based. It is therefore independent of the discretization and can be trained on datasets with varying discretizations.

On the first assessed dataset, all examples use the same discretization, and the CNF achieved a similar reconstruction error as a CAE.

28 October 2026 11:51:29

© Author(s) 2026

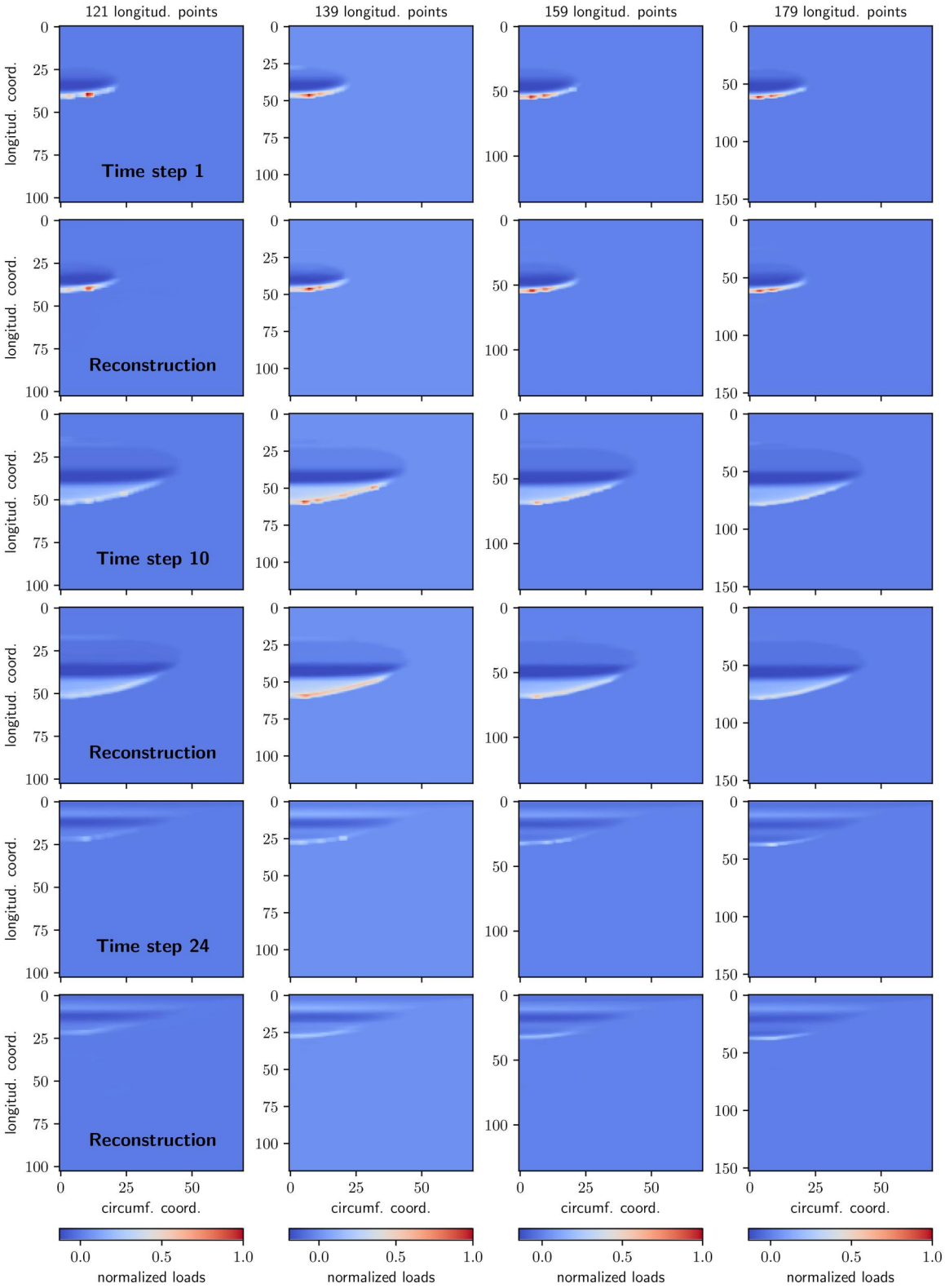


FIG. 10. Simulated loads with the different unseen discretizations at three time steps of the test case and the corresponding CNF reconstructions.

28 October 2026 11:51:29

© Author(s) 2026

Slightly lower errors were obtained for the larger CNF layer dimensions 64 and 128. When paired with an LSTM network in the latent space, the CNF model, however, achieved slightly lower accuracy than the CAE on spatiotemporal ditching load predictions. The smallest tested layer dimension of 32 leads to the lowest error for the CNF in this case, while having around 88% fewer parameters than the CAE. The second dataset used consists of ditching loads that are trained for different discretizations and evaluated on yet other discretizations. The results demonstrated that the CNF can generalize to discretizations that were not seen during training, which is a very useful feature for the engineering application of a surrogate model.

General downsides of the CNF approach are the longer training and inference times compared to those of a CAE, especially for encoding test data. When data are available in the same discretization, CAE-based models thus have several advantages over the CNF approach. For more complicated datasets with heterogeneous discretizations, the CNF approach can be used without further data processing and therefore is the preferable choice.

ACKNOWLEDGMENTS

H.S., P.P.L., and T.R. acknowledge support by the German Federal Ministry for Economic Affairs and Energy under aegis of the "Luftfahrtforschungsprogramm LuFo VI" project HYMNE (Grant No. 20E2218A). This paper is a contribution to the research training group RTG 2583 on "Modeling, Simulation and Optimization of Fluid Dynamic Applications" funded by the Deutsche Forschungsgemeinschaft (DFG). The authors acknowledge the support of TU Hamburg's Machine Learning in Engineering (MLE) initiative.

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

Henning Schwarz: Conceptualization (equal); Data curation (equal); Formal analysis (lead); Investigation (lead); Methodology (equal); Software (lead); Validation (lead); Visualization (lead); Writing – original draft (equal); Writing – review & editing (equal). **Pyei Phyo Lin:** Data curation (equal); Methodology (equal); Writing – review & editing (equal). **Jens-Peter M. Zemke:** Conceptualization (equal); Methodology (equal); Writing – review & editing (equal). **Thomas Rung:** Conceptualization (equal); Funding acquisition (lead); Methodology (equal); Project administration (lead); Resources (lead); Supervision (lead); Writing – original draft (equal); Writing – review & editing (equal).

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author upon reasonable request.

REFERENCES

[1]J. L. Lumley, "The structure of inhomogeneous turbulent flows," in *Atmospheric Turbulence and Radio Wave Propagation* (Nauka, 1967), pp. 166–178.
[2]L. Agostini, "Exploration and prediction of fluid dynamical systems using auto-encoder technology," Phys. Fluids **32**, 067103 (2020).
[3]H. Eivazi, H. Veisi, M. H. Naderi, and V. Esfahanian, "Deep neural networks for nonlinear model order reduction of unsteady flows," Phys. Fluids **32**, 105104 (2020).
[4]R. Pache and T. Rung, "Data-driven surrogate modeling of aerodynamic forces on the superstructure of container vessels," Eng. Appl. Comput. Fluid Mech. **16**, 746–763 (2022).
[5]H. Eivazi, S. L. Clainche, S. Hoyas, and R. Vinuesa, "Towards extraction of orthogonal and parsimonious non-linear modes from turbulent flows," Expert Syst. Appl. **202**, 117038 (2022).
[6]Y.-E. Kang, S. Yang, and K. Yee, "Physics-aware reduced-order modeling of transonic flow via β-variational autoencoder," Phys. Fluids **34**, 076103 (2022).
[7]A. Solera-Rico, C. Sanmiguel Vila, M. Gómez-López, Y. Wang, A. Almashjary, S. Dawson, and R. Vinuesa, "β-variational autoencoders and transformers for reduced-order modelling of fluid flows," Nat. Commun. **15**, 1361 (2024).
[8]H. Schwarz, M. Überrück, J.-P. M. Zemke, and T. Rung, "Machine learning based prediction of ditching loads," AIAA J. **63**, 1835–1854 (2025).
[9]M. Loft, H. Schwarz, and T. Rung, "Data-driven pressure field prediction for ships in regular sea states," Eng. Appl. Comput. Fluid Mech. **19**, 2553337 (2025).
[10]S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Comput. **9**, 1735–1780 (1997).
[11]A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems* (Curran Associates., Inc., 2017).
[12]P. Wu, S. Gong, K. Pan, F. Qiu, W. Feng, and C. Pain, "Reduced order model using convolutional auto-encoder with self-attention," Phys. Fluids **33**, 077107 (2021).
[13]A. Hemmasian and A. Barati Farimani, "Reduced-order modeling of fluid flows with transformers," Phys. Fluids **35**, 057126 (2023).
[14]M. A. Maram, G. Bletsos, T. T. Nguyen, A. Hassan, M. Palm, and T. Rung, "Adjoint-based shape optimization of a ship hull using a Conditional Variational Autoencoder (CVAE) assisted propulsion surrogate model," Comput. Fluids **321**, 107284 (2026).
[15]T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, and P. Battaglia, "Learning mesh-based simulation with graph networks," in International Conference on Learning Representations, 2021.
[16]J. Brandstetter, D. E. Worrall, and M. Welling, "Message passing neural PDE solvers," in International Conference on Learning Representations, 2022.
[17]S. Barwey, V. Shankar, V. Viswanathan, and R. Maulik, "Multiscale graph neural network autoencoders for interpretable scientific machine learning," J. Comput. Phys. **495**, 112537 (2023).
[18]N. R. Franco, S. Fresca, F. Tombari, and A. Manzoni, "Deep learning-based surrogate models for parametrized pdes: Handling geometric variability through graph neural networks," Chaos **33**, 123121 (2023).
[19]L. Serrano, L. L. Boudec, A. Kassaï Koupaï, T. X. Wang, Y. Yin, J.-N. Vittaut, and P. Gallinari, "Operator learning with neural fields: Tackling PDEs on general geometries," in *Advances in Neural Information Processing Systems*, edited by A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (Curran Associates, Inc., 2023), Vol. 36, pp. 70581–70611.
[20]Y. Xie, T. Takikawa, S. Saito, O. Litany, S. Yan, N. Khan, F. Tombari, J. Tompkin, V. Sitzmann, and S. Sridhar, "Neural fields in visual computing and beyond," Comput. Graph. Forum **41**, 641–676 (2022).
[21]J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "DeepSDF: Learning continuous signed distance functions for shape representation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, 2019).
[22]N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio, and A. Courville, "On the spectral bias of neural networks," in *Proceedings of the 36th International Conference on Machine Learning, Proceedings of Machine Learning Research*, edited by K. Chaudhuri and R. Salakhutdinov (PMLR, 2019), Vol. 97, pp. 5301–5310.
[23]V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein, "Implicit neural representations with periodic activation functions," in *Advances in Neural Information Processing Systems*, edited by H. Larochelle, M. Ranzato,

28 October 2026 11:51:29

© Author(s) 2026

R. Hadsell, M. Balcan, and H. Lin (Curran Associates, Inc., 2020), Vol. 33, pp. 7462–7473.
[24]A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Advances in Neural Information Processing Systems*, edited by J. Platt, D. Koller, Y. Singer, and S. Roweis (Curran Associates, Inc., 2007), Vol. 20.
[25]M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng, "Fourier features let networks learn high frequency functions in low dimensional domains," in *Proceedings of the 34th International Conference on Neural Information Processing Systems, NeurIPS'20* (Curran Associates Inc., Red Hook, NY, 2020).
[26]Y. Yin, M. Kirchmeyer, J.-Y. Franceschi, A. Rakotomamonjy, and P. Gallinari, "Continuous PDE dynamics forecasting with implicit neural representations," in The Eleventh International Conference on Learning Representations, 2023.
[27]P. Du, M. H. Parikh, X. Fan, X.-Y. Liu, and J.-X. Wang, "Conditional neural field latent diffusion model for generating spatiotemporal turbulence," Nat. Commun. **15**, 10416 (2024).
[28]J. Guo, P. Du, X. Fan, Y. Li, and J.-X. Wang, "Conditional neural field for spatial dimension reduction of turbulence data: A comparison study," Phys. Fluids **38**, 025153 (2026).
[29]G. Catalani, J. Fesquet, X. Bertrand, F. Tost, M. Bauerheim, and J. Morlier, "Towards scalable surrogate models based on neural fields for large scale aerodynamic simulations," Comput. Fluids **306**, 106929 (2026).
[30]Z. Li, N. B. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, "Fourier neural operator for parametric partial differential equations," in International Conference on Learning Representations, 2021.
[31]S. Wang, J. H. Seidman, S. Sankaran, H. Wang, G. J. Pappas, and P. Perdikaris, "CViT: Continuous vision transformer for operator learning," arXiv:2405.13998 [cs.LG] (2025).
[32]L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis, "Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators," Nat. Mach. Intell. **3**, 218–229 (2021).
[33]S. Pan, S. L. Brunton, and J. N. Kutz, "Neural implicit flow: A mesh-agnostic dimensionality reduction paradigm of spatio-temporal data," J. Mach. Learn. Res. **24**, 1–60 (2023).
[34]M. Raissi, P. Perdikaris, and G. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," J. Comput. Phys. **378**, 686–707 (2019).
[35]H. Streckwall, O. Lindenau, and L. Bensch, "Aircraft ditching: A free surface/free motion problem," Arch. Civil Mech. Eng. **7**, 177–190 (2007).
[36]H. Schwarz, P. P. Lin, J.-P. M. Zemke, and T. Rung, "Disentangled latent spaces for reduced order models using deterministic autoencoders," Comput. Fluids **302**, 106837 (2025).
[37]D. Ha, A. M. Dai, and Q. V. Le, "Hypernetworks," in International Conference on Learning Representations, 2017.
[38]V. Sitzmann, M. Zollhoefer, and G. Wetzstein, "Scene representation networks: Continuous 3D-structure-aware neural scene representations," in *Advances in Neural Information Processing Systems*, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Curran Associates, Inc., 2019), Vol. 32.
[39]E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. Courville, "FiLM: Visual reasoning with a general conditioning layer," Proc. AAAI **32**, 3942–3951 (2018).
[40]V. Dumoulin, E. Perez, N. Schucher, F. Strub, H. de Vries, A. Courville, and Y. Bengio, "Feature-wise transformations," Distill (published online 2018).
[41]A. Jacot, F. Gabriel, and C. Hongler, "Neural tangent kernel: Convergence and generalization in neural networks," in *Advances in Neural Information Processing Systems*, edited by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Curran Associates, Inc., 2018), Vol. 31.
[42]T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," in *Advances in Neural Information Processing Systems*, edited by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (Curran Associates, Inc., 2016), Vol. 29.
[43]D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in Proceedings of the 3rd International Conference on Learning Representations (ICLR), 2015.
[44]L. Zintgraf, K. Shiarli, V. Kurin, K. Hofmann, and S. Whiteson, "Fast context adaptation via meta-learning," in *Proceedings of the 36th International Conference on Machine Learning*, Proceedings of Machine Learning Research, edited by K. Chaudhuri and R. Salakhutdinov (PMLR, 2019), Vol. 97, pp. 7693–7702.
[45]A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala *et al.*, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems, 32*, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Curran Associates, Inc., 2019), pp. 8024–8035.
[46]L. Bensch, V. Shigunov, and H. Söding, "Computational method to simulate planned ditching of a transport airplane," in *Computational Fluid and Solid Mechanics 2003*, edited by K. Bathe (Elsevier Science Ltd, Oxford, 2003), pp. 1251–1254.

28 October 2026 11:51:29

© Author(s) 2026