Supplementary Material

After repositioning slices, we arrived at imaging data with gaps between the available imaging data layers. To solve this impainting problem, we used convolutional neural networks (CNN) to interpolate between the available layers. In the following details of data preparation, network architecture and training are summarized.

1 DATA PREPARATION

We trained and validated the machine learning model on the segmented (binarized) synthetic imaging data (resolution of 2.5 nm in all directions). To this end, we extracted all available zx- and zy-planes from the imaging data and split these planes into training data (ca. 80%) and validation data (ca. 20%). Masks were generated for all missing regions in the dataset. For training, we used data from our synthetic samples, for which complete images without missing regions could be prepared as ground truth (because, for these samples, the exact microstructure was known). We used an online data augmentation technique to increase the training data. That is, we applied random rotations and horizontal/vertical flips to the training data. We did not apply any transformations to the validation dataset.

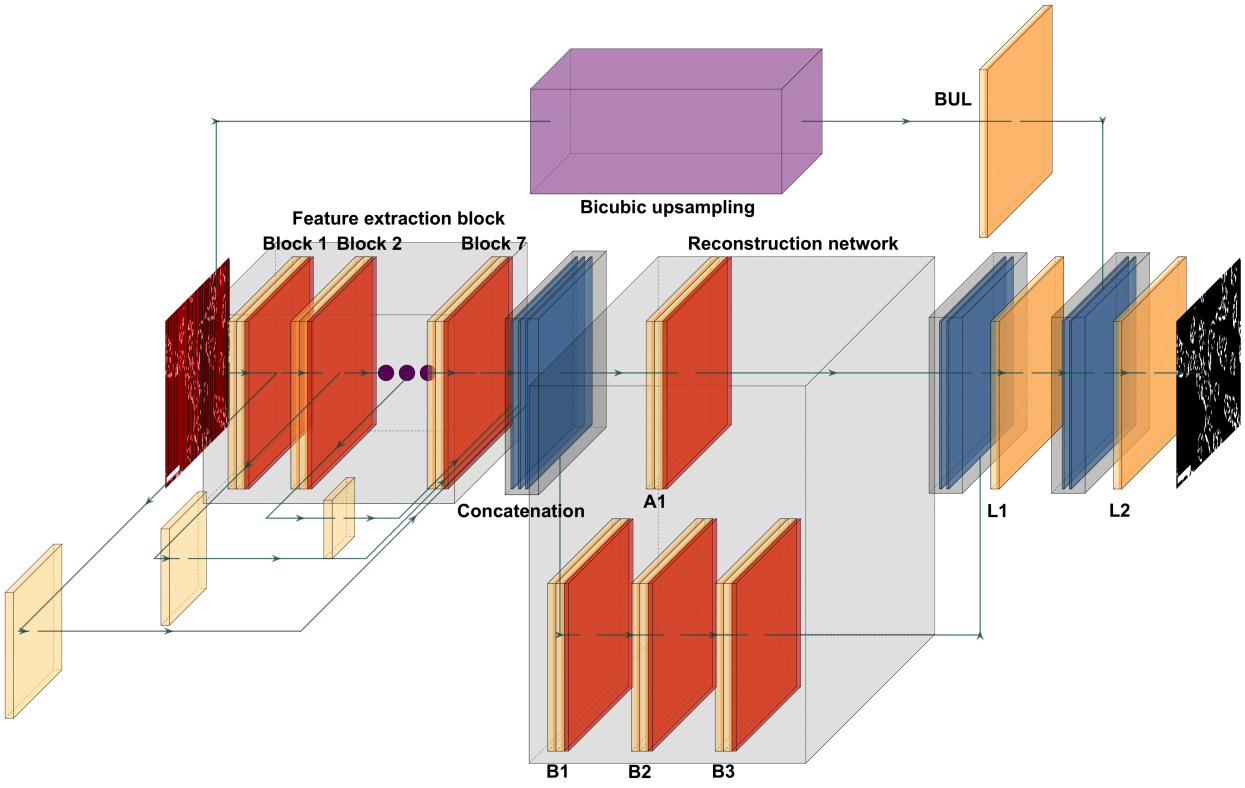

Figure S1. Machine learning architecture used in this paper to interpolate voxelated data points (i.e., to solve the inpainting problem of interest)

2 ARCHITECTURE

Our machine-learning model architecture was implemented using PyTorch and was inspired by Yamanaka et al. (2017). Their ML model was developed to solve problems of image super-resolution; by contrast, we used our modified ML model to solve an image inpainting problem. In this ML model, encoded features from the input image are extracted by seven convolutional blocks in the feature extraction network. These encoded features are then concatenated and passed through a reconstruction network, which reconstructs an output array of the same size as the input image. We use a different number of filters in the feature extraction and the reconstruction network blocks than Yamanaka et al. (2017). The number of filters in both network blocks is summarized in Table S1. In our modified ML model, we concatenated the output of the reconstruction network and a separate image dataset prepared from the input of the overall architecture using bicubic interpolation (Figure S1). This was in contrast to adding them together as mentioned by Yamanaka et al. (2017). Our approach provided good results even after a short training period of approximately 14 hours and 30 minutes. However, our machine learning-based method is computationally expensive compared to the classical method, cubic splines interpolation. Our method with a batch size of 128 takes approximately 44 seconds compared to 7 seconds for cubic splines interpolation for 128 images of size 512×512 pixels. It should be noted that the time taken to calculate results using our method depends on several parameters, including batch size and GPU memory capacity.

Table S1. Number of filters used in each block of our machine learning architecture

Feature extraction								Reconstruction				Concatenate		Bicubic layer
								A1	B1	B2	B3	L1	L2	BUL
96	81	70	60	50	41	32	16	96	64	48	32	1		1

3 TRAINING

We trained our machine learning architecture on Tesla K80 GPUs. We provided small image patches (64×64 pixels) with a stride of 32 pixels as input to train the ML model using a sliding window technique. We used a mean squared error (MSE) loss function and used the Adam optimization algorithm with an initial learning rate of 0.001. We reduced the learning rate by a factor of 10 if the loss did not decrease for ten consecutive training epochs. Additional parameters used for the training process are specified in Table S2. Figure S2 illustrates a typical training data and validation data MSE loss curve.

4 IMAGE POST-PROCESSING

We noticed clusters of missing pixels, *holes*, in some of the ligaments after performing cubic-splines interpolation (Figure S3). These *holes* were filled using image processing techniques to improve the quality of these interpolated structures. Specifically, we used the Scikit-image package [1] to detect *holes* which were entirely covered by the gold structures. We only considered *holes* having an area smaller than 20 pixels to ensure that any secondary hierarchy of the ligaments was not classified as artefacts. *Holes* detected in this manner were then filled with a gold intensity to enhance the overall structure of the ligaments.

[1] https://scikit-image.org/docs/stable/api/skimage.morphology.html

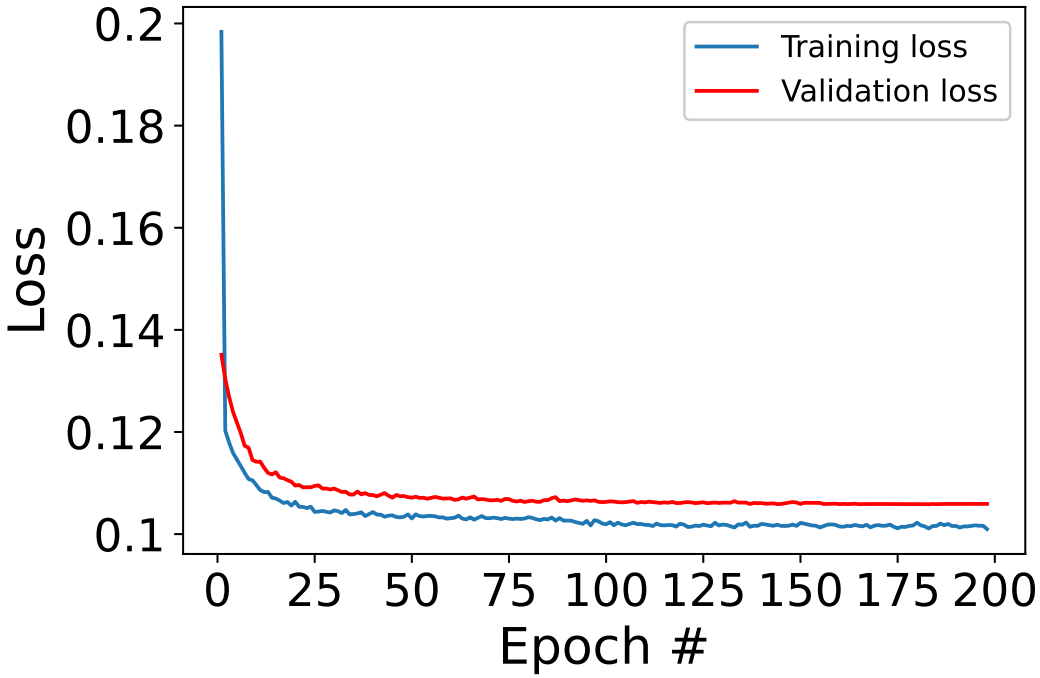

Figure S2. Training data and validation data MSE loss function over 200 epochs

Table S2. Parameters used for training machine learning model (interpolation)

Parameter	Value
Patch size	64
Stride	0.5
Batch size	64
Epochs	500 with early stopping with patience=25
Loss	MSE loss
Optimizer	Adam
Learning rate	0.001 with reducing it by factor of 0.10 with patience of 10

(A) (B)

Figure S3. Interpolated ligament of hierarchical nanoporous gold using cubic-splines interpolation **(A)** without post-processing and **(B)** after post-processing

5 ADDITIONAL RESULTS USING ANISOTROPY-BASED ERROR MEASURES FOR SYNTHETIC DATA

Table S3. Performance comparison of different interpolation techniques on 250 different synthetic datasets.

Dataset name	$e_{L_2}^{TPCF} \downarrow$	$e_{L_2}^{LPF} \downarrow$	$e_{L_2}^{D} \downarrow$
Original - [Target values]	0.1038 ± 0.0283	0.0188 ± 0.0131	0.0138 ± 0.0096
RNN-*s*	0.1149 ± 0.0247	0.0479 ± 0.0222	0.0306 ± 0.0185
RCS-*s*	0.1091 ± 0.0278	0.0482 ± 0.0188	0.0299 ± 0.0173
RML-*s*	0.1086 ± 0.0274	0.0341 ± 0.0131	0.0211 ± 0.0114

Table S4. Performance comparison of different interpolation techniques on 250 different synthetic datasets.

Dataset name	$e_{L_2}^{TPCF} \downarrow$	$e_{L_2}^{LPF} \downarrow$	$e_{L_2}^{D} \downarrow$
Original - [Target values]	0.1038 ± 0.0283	0.0188 ± 0.0131	0.0138 ± 0.0096
DNN-*s*	0.2104 ± 0.0467	0.0633 ± 0.0276	0.0386 ± 0.0191
SRCS-*s*	0.1153 ± 0.0267	0.0716 ± 0.0222	0.0455 ± 0.0198
RML-*s*	0.1086 ± 0.0274	0.0341 ± 0.0131	0.0211 ± 0.0114

REFERENCES

Yamanaka, J., Kuwashima, S., and Kurita, T. (2017). Fast and accurate image super resolution by deep cnn with skip connection and network in network. In *International Conference on Neural Information Processing* (Springer), 217–225