Convolutional neural networks for segmentation of FIB-SEM nanotomography data from porous polymer films for controlled drug

Received: 2 February 2021 | Revised: 12 March 2021 | Accepted: 21 March 2021

DOI: 10.1111/jmi.13007

Journal of Microscopy RMS

ORIGINAL ARTICLE

Convolutional neural networks for segmentation of FIB-SEM nanotomography data from porous polymer films for controlled drug release

Fredrik Skärberg[1] | **Cecilia Fager**[2,3] | **Francisco Mendoza-Lara**[4] | **Mats Josefson**[4] | **Eva Olsson**[2] | **Niklas Lorén**[1,2] | **Magnus Röding**[1,5]

[1] Bioeconomy and Health, Agriculture and Food, RISE Research Institutes of Sweden, Göteborg, Sweden

[2] Department of Physics, Chalmers University of Technology, Göteborg, Sweden

[3] Department of Fibre and Polymer Technology, KTH Royal Institute of Technology, Stockholm, Sweden

[4] Oral Product Development, Pharmaceutical Technology & Development, Operations, AstraZeneca Gothenburg, Mölndal, Sweden

[5] Department of Mathematical Sciences, Chalmers University of Technology and University of Gothenburg, Göteborg, Sweden

Correspondence
Magnus Röding, RISE Research Institutes of Sweden, Frans Perssons väg 6, 41276 Göteborg, Sweden.
Email: magnus.roding@ri.se

Funding information
The Swedish Research Council, Grant number 2016-03809; The Swedish Research Council for Sustainable Development, Grant number 2019-01295

Abstract

Phase-separated polymer films are commonly used as coatings around pharmaceutical oral dosage forms (tablets or pellets) to facilitate controlled drug release. A typical choice is to use ethyl cellulose and hydroxypropyl cellulose (EC/HPC) polymer blends. When an EC/HPC film is in contact with water, the leaching out of the water-soluble HPC phase produces an EC film with a porous network through which the drug is transported. The drug release can be tailored by controlling the structure of this porous network. Imaging and characterization of such EC porous films facilitates understanding of how to control and tailor film formation and ultimately drug release. Combined focused ion beam and scanning electron microscope (FIB-SEM) tomography is a well-established technique for high-resolution imaging, and suitable for this application. However, for segmenting image data, in this case to correctly identify the porous network, FIB-SEM is a challenging technique to work with. In this work, we implement convolutional neural networks for segmentation of FIB-SEM image data. The data are acquired from three EC porous films where the HPC phases have been leached out. The three data sets have varying porosities in a range of interest for controlled drug release applications. We demonstrate very good agreement with manual segmentations. In particular, we demonstrate an improvement in comparison to previous work on the same data sets that utilized a random forest classifier trained on Gaussian scale-space features. Finally, we facilitate further development of FIB-SEM segmentation methods by making the data and software used open access.

KEYWORDS

controlled drug release, convolutional neural networks, deep learning, focused ion beam scanning electron microscopy, image analysis, machine learning, microstructure, polymer films, porous materials, semantic segmentation

1 | INTRODUCTION

A common means of controlling the release of active pharmaceutical ingredients is to coat oral dosage forms (tablets or pellets) with phase-separated polymer films. Frequently, the two cellulose derivatives ethyl cellulose (EC) and hydroxypropyl cellulose (HPC) polymer blends are used. When an oral dosage form is swallowed and the EC/HPC film comes in contact with water, the water-soluble HPC phase is leached out. The resulting material is a film consisting of a water-insoluble EC matrix with a porous network through which the drug is transported by diffusion. The drug release can be efficiently controlled by tailoring the structure of such a porous network, as studied both in experiments and simulations.[1–5]

A detailed understanding of how to control and tailor film formation and ultimately drug release requires high spatial resolution imaging and accurate characterization of such EC porous films. Whereas many pores are too small to be resolved to a satisfactory degree even with state-of-the-art X-ray computed tomography (X-ray CT) systems, combined focused ion beam and scanning electron microscope (FIB-SEM) nanotomography[6] has proven highly useful for imaging such materials. Using FIB-SEM, a 3D region of interest is imaged using a slice and image procedure. First, an initial planar cross-section is created using the FIB. Then, the planar cross-section surface is imaged using the SEM. The iterative slice and image procedure continues until the full volume of interest has been acquired. A FIB-SEM tomography data set typically consists of a few hundred 2D SEM images acquired in this fashion.

There are numerous challenges associated with using FIB-SEM on soft, poorly conducting and porous materials such as porous polymer films. Some examples are accumulation of charges at the surface, seen as very bright or very dark areas in the images,[7] so-called curtaining effects due to local thickness or hardness variations, seen as alternating bright and dark lines parallel to the ion beam,[8] and non-constant path lengths of electrons from the beam to the detector, seen as an artificial intensity gradient in the images.[9] For a comprehensive review of how to address the challenges mentioned above, we refer the reader to Fager et al.[10] for more details. Finally, the most pronounced challenge regarding FIB-SEM tomography data from porous materials is the occurrence of subsurface information: that a 2D SEM image of a porous cross-section will contain information about the structure not only of the current planar cross-section surface but also deeper inside the pores. Therefore, the supposedly 2D SEM images become 2.5D in a sense, which clearly distinguishes FIB-SEM from, for example X-ray CT. This is the main reason why image analysis of FIB-SEM data from porous materials is demanding.

Many segmentation approaches tailored for FIB-SEM have been proposed, such as adaptive thresholding,[11] level sets,[12] morphological image processing,[13] local threshold backpropagation,[14,15] watershed segmentation[16] and optical flow-based segmentation.[17] More to the point, in Fend et al.,[18] convolutional neural networks (CNNs) were used, specifically fully convolutional networks of the U-net type Ronneberger et al.[19] Briefly, for the U-net architecture, the input constitutes a patch of image data (of arbitrary size, in principle) and the output is a patch of classification scores of the same size as the input. Once trained, an entire image patch can thus be segmented at once which provides for fast inference. In contrast, other CNN segmentation methods rely on a sliding-window approach in which a 2D or 3D neighbourhood surrounding a single voxel is used to predict the class membership of that voxel. This approach demands explicit looping over all voxels of a data set and making as many independent predictions as there are voxels. This approach is much slower than the fully convolutional networks for inference. Since their conception in 2015, U-net type CNNs have been established as the best segmentation paradigm for a range of problems. Finally, of particular relevance for this work is Röding et al.[20] There, a random forest algorithm combined with Gaussian scale-space features was applied to the same data set used herein, providing good agreement with manual segmentations in terms of, for example accuracy and estimated porosity of the material.

In this work, we implement CNNs for segmentation of FIB-SEM image data. The data are acquired from three EC porous films where the HPC phases have been leached out. Before leaching, the films had different HPC fractions. Hence, the three data sets have varying porosities, in a range of interest for controlled drug release applications. In contrast to some other applications, we have obtained better performance with sliding-window CNNs than with U-net type CNNs,[21,22]); hence, we herein implement sliding-window CNNs with different neighbourhood dimensions and architectures. We demonstrate very good agreement with manual segmentations carried out by an expert and in particular, we demonstrate an improvement in comparison to previous work on the same data sets in Röding et al.[20] We make all the data and software used herein open access[23] to facilitate further development of FIB-SEM segmentation methods.

2 | MATERIALS AND METHODS

Phase-separated films are produced as follows. Three solutions of EC (Ethocel Standard premium, viscosity

10 cP, Dow Wolff Cellulosics GmbH) and HPC (Klucel Pharm HPC, viscosity grade LF) with 95% ethanol as solvent[24–26] are prepared. All have 6% (w/w) polymer (total polymer concentration for EC and HPC combined), and the weight fractions of HPC are 22%, 30% and 45%. The solutions are sprayed onto a rotating drum. At low polymer concentration, EC and HPC form a homogeneous, one-phase polymer blend. However, as the ethanol evaporates, the mixture becomes incompatible, phase separation starts and continues until enough ethanol has evaporated such that the polymer structure becomes kinetically trapped. Then, the films are removed from the rotating drum and stored in a desiccator. By subjecting the dried films to stirred deionized water at ambient conditions for 24 h, the water-soluble HPC phases are leached out, producing porous EC films which are then air-dried. Because the molecular weights of these EC and HPC grades are close, the weight fractions reflect the volume fractions of HPC very closely. Therefore, the porosities of these films will be approximately 22%, 30%, and 45%, assuming that the HPC phases are fully leached out (the 22% sample will be close to the percolation threshold[27] and hence the HPC phase might not be entirely leached out). The samples are denoted HPC22, HPC30 and HPC45 below.

A TESCAN GAIA3 (TESCAN), equipped with a gas injection system for platinum and carbon, is used for FIB-SEM tomography. Prior to insertion into the FIB-SEM, the samples are coated with palladium to reduce charging effects. A platinum layer is deposited on the surface to reduce curtaining effects. Furthermore, a U-shaped trench is milled around the cross-section to reduce shadowing effects, followed by polishing of the cross-section. Imaging is carried out using the backscattered electron detector (BSE) and the electron beam scan speed is 2 μs/pixel. The pixel size is 10 nm. Serial sectioning is done with a slice thickness of 50 nm, acquiring 200 slices per sample. Imaging took approximately 24 h for each data set. We refer the reader to Fager et al.[10] for more experimental details.

3 | RESULTS AND DISCUSSION

First, we perform data pre-processing, followed by manual segmentation of a subset of the data set and extraction of data for training of the CNN models. Secondly, we describe the CNN network architecture, followed by data augmentation and the procedure for training and hyperparameter optimization. Finally, the best identified model is used to segment the full data sets, followed by post-processing to improve the results further.

3.1 | Pre-processing

First, the 2D cross-section images are aligned using ImageJ[28] with the StackReg plug-in and the Rigid Body method. In this manner, the images are aligned based on finding translations and rotations for optimal alignment. After alignment, equal sized volumes from all three samples are extracted, with resolution $2247 \times 3372 \times 200$ voxels, where each voxel is 10 nm $\times$ 10 nm $\times$ 50 nm. This comprises regions of interest of approximately 22.5 μm $\times$ 33.7 μm $\times$ 10.0 μm. Secondly, the 16-bit data are converted to $[0, 1]$ range. Thirdly, as in Röding et al.,[20] intensity gradients present in the x direction of the images are removed in the following fashion: we fit linear functions to the mean intensities of each data set as a function of the x-coordinate, using least squares. This function is subtracted, and its mean value is added again to retain the original intensity range.

3.2 | Manual segmentation

The manual segmentation used is the one produced in Röding et al.[20] Because of the size of the data sets, manual segmentation is carried out only on a small subset of the full data. In each data set, 100 regions (of size 256×256 voxels, and placed in the x-y plane, that is within a single planar cross-section) are randomly selected, corresponding to ~0.43% of the full data set. Neighbourhoods of size $384 \times 384 \times 7$ voxels centred around these regions are presented to the expert that performs manual segmentation, using MITK Workbench (Medical Imaging Interaction Toolkit, www.mitk.org) for careful manual tracing of the edges of the pores.

3.3 | Data extraction

The number of annotated samples available equals the number of voxels in the manually segmented regions combined, that is $256 \times 256 \times 100 \times 3 = 19,660,800$, although some of them are strongly correlated. To split the data into training, validation and test data, we use the same procedure as in Röding et al.[20] Although the regions are already randomly selected from the beginning, they are randomly shuffled again to minimize the effect of mistakes and fatigue during manual segmentation. Of the 100 regions in each data set, 60 are used for training, 20 for validation and 20 for testing. Thereby, in total, 180 regions are used for training, 60 for validation and 60 for testing. To facilitate comparison, we use the same regions (same random seed) as in the earlier work.

Once the regions have been divided into training, validation and test sets, individual samples are extracted. For training, stratified random sampling is used such that a 50/50 class balance between pore (leached HPC) and solid (EC) is obtained. For each of the training, validation and test data sets, 1% of the available data (voxels) are extracted, that is 117,964 samples for the training set and 39,320 samples each for the validation and test sets. Multiple sizes of the neighbourhoods used as input to the CNNs will be investigated as elaborated upon below.

3.4 | Network architecture

The aim for the investigated CNN architectures is to learn the task of predicting the class membership (pore or solid) of a single voxel using a neighbourhood centred around that voxel, that is an array of greyscale intensities, of size $n_{xy} \times n_{xy} \times n_z$ voxels, as input data.

An artificial neural network is essentially a composition of operations that together form an arbitrarily complex non-linear mapping from input to output. In a classical artificial neural network (here denoted ANN), the main building blocks are the so-called dense or fully connected layers, each consisting of a number of nodes (neurons) and a bias. For each node, its input is a vector x, and its scalar output y is a function of the sum of the elements of x weighted by w (i.e. a dot product) plus a bias b, $y = f(w \cdot x + b)$, where f is a (non-linear) so-called activation function. Conventionally, all elements of the input data provide input to all nodes of the first layer, the outputs of all nodes of the first layer provide input to all nodes of the second layer, and so on (thereby, fully connected). Finally, after a number of layers, a scalar or vector output is obtained. By optimizing the weights w and biases b (both weights and biases are jointly referred to as weights from here on) of all nodes with respect to a loss function that measures the deviation between the target output and the output obtained from the network, that is training the network, a highly non-linear function approximation is obtained. However, in an ANN, a high-dimensional input such as the image data used in this work would result in a vast number of parameters, and ANNs are generally not efficient for such input data.[29]

In contrast, in a CNN, the main building blocks are the convolutional layers. In such layers, the input is convolved with filters or convolution kernels. CNNs are generally more efficient than ANNs when working with image data due to the properties of the convolution kernels; the weights of a convolution layer are the elements of the relatively small convolution kernels, and the same convolution is applied over the entire input, radically reducing the number of weights.[30] In practice, a CNN architecture is typically formed by a set of convolutional layers, pooling layers and fully connected layers. In a convolutional layer, the input is convolved with one or several convolution kernels (typically of size 3×3, or occasionally 5×5), producing outputs known as feature maps. Although convolutions are linear, a non-linear activation function f is applied to the result to produce an output. In a pooling layer, typically placed after one or more convolutional layers, the feature maps are downsampled. A pooling layer does not have any tuneable weights, but rather performs a simple operation on small, non-overlapping patches (typically windows of size 2×2) of the feature maps from the preceding convolutional layer, such as computing the mean or the maximum. The effect is a reduction in resolution, and the succeeding convolutional layer(s) can be used to produce feature maps at another spatial scale. Fully connected layers are often used as the final layers of a CNN to produce an output. Whereas the combination of convolutional layers and pooling layers can be thought of as a feature extraction procedure, the fully connected layers act as a classifier of these features. In these final layers of the CNN, the input and output dimensions are far lower than in the beginning, making the use of fully connected layers more suitable.

In this work, we explore 2D multi-channel CNNs. Assuming that the input is an $n_{xy} \times n_{xy} \times n_z$ array, the CNNs will consider the data as 2D images of size $n_{xy} \times n_{xy}$ with n_z channels. In other settings, channels might correspond to colours (e.g. red, green and blue) but here, they correspond to image slices in the x-y plane. The input sizes that will be considered are $n_{xy} \in \{33, 49, 65, 81, 97, 113\}$ (i.e. $16n + 1$ for $n \in \{2, 3, 4, 5, 6, 7\}$) and $n_z \in \{1, 3, 5, 7, 9, 11\}$. The fact that n_z is considerably smaller than n_{xy} is motivated by the fact that the voxel size in the image data is $10\,\text{nm} \times 10\,\text{nm} \times 50\,\text{nm}$ and hence non-isotropic. In Figure 1, the different neighbourhood sizes are illustrated.

The CNN architecture used is in principle the same for all combinations of n_{xy} and n_z. The convolutional part of the network consists of four blocks, each comprising two convolutional layers and one max-pooling layer. The convolutional layers use 3×3 kernels and the max-pooling layers use 2×2 windows. The number of filters, i.e. kernels applied in each convolutional layer is 16, 32, 64 and 80, respectively, in the four blocks. After each convolutional layer, exponential linear unit (Elu) activation,

$$f(x) = \begin{cases} x, & x > 0 \\ \gamma(e^x - 1), & x \leq 0 \end{cases} \tag{1}$$

is used,[31] with $\gamma = 1$. The output from the convolutional part is used as input to two fully connected layers with 128

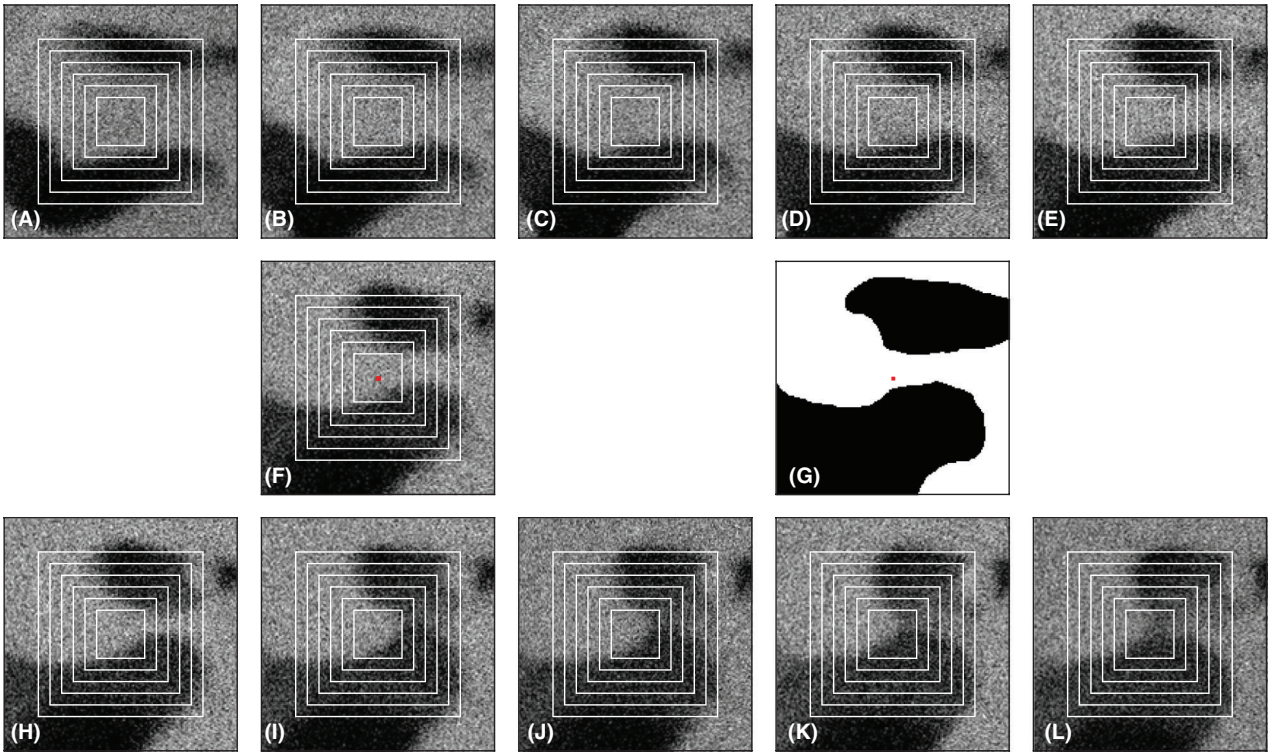

FIGURE 1 Illustration of the different neighbourhood sizes n_{xy} and n_z used. In (A–E), the five slices before the slice of interest are shown, and in (H–L), the five slices after the slice of interest are shown. In (F), the slice of interest is shown, complemented by the corresponding manual segmentation mask in (G). In each slice, squares indicating the values $n_{xy} = 33, 49, 65, 81, 97$ and 113 are shown. For $n_z = 1$, only (F) is used, for $n_z = 3$, (E), (F) and (H) are used and so on. In (F–G), the voxel of interest to be classified is indicated (red). The images constitute crops of the corresponding full 2D SEM images and the field of view is 1.6 μm × 1.6 μm (160 × 160 voxels)

and 64 nodes, both also using Elu activation. The final layer is an output layer with a single node. The output layer is intended to produce a score in $[0, 1]$ that can be thresholded to yield a classification. Therefore, sigmoid activation,

$$f(x) = \frac{1}{1 + e^{-x}}, \tag{2}$$

for which $f(x) \in (0, 1)$ for all $x \in (-\infty, \infty)$, is used. In Figure 2, the architecture is illustrated. The number of weights in the network depends on the neighbourhood size. Primarily, the number depends on n_{xy}, because its value greatly influences the number of connections from the convolutional part into the fully connected part, which constitutes a large fraction of the overall number of weights (we reiterate that the crucial, main advantage of the convolutional part is that it does *not* have so many weights). In contrast, n_z only influences the number of weights in the first convolutional layer. The number of weights as a function of neighbourhood size is shown in Table 1.

The choice of architecture is a result of a previous investigation,[22] and it was found that the selected number of blocks of convolutional and pooling layers, the kernel sizes and the number of filters, the number of fully connected layers and the number of nodes were appropriate for all the neighbourhood sizes investigated herein.

3.5 | Data augmentation

The purpose of data augmentation is to increase the amount of available data by adding modified versions of the original elements of the training data, which can help regularize the training of the CNNs and lead to better generalization of the performance to the validation and test sets. We investigate different data augmentation schemes (for the training data) such as adding noise and blur, random anisotropic scaling and rotations with arbitrary angles, but find only two augmentations that yield noticeable improvement: (i) random rescaling of the mean and standard deviation of the input intensity, and (ii) random flips and rotations by multiples of 90° in the x-y plane. Specifically, random rescaling of the intensities is carried out in the following fashion. For each sample to be augmented, let the mean and standard deviation of the intensities of the input neighbourhood of size $n_{xy} \times n_{xy} \times n_z$

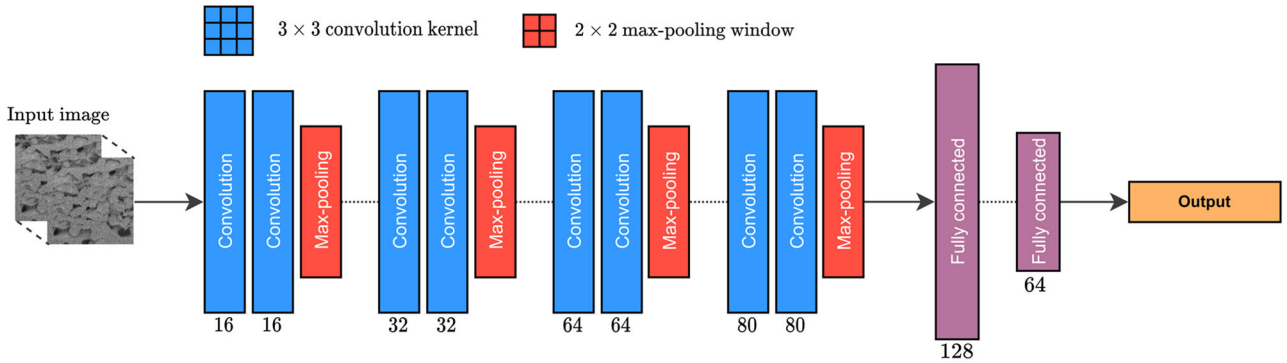

FIGURE 2 Illustration of the CNN architecture. The input, regarded as an $n_{xy} \times n_{xy}$ image with n_z channels, is fed into the convolutional part of the network. The convolutional part of the network comprises four blocks, each with two convolutional layers and one max-pooling layer and with increasing numbers of convolutional filters, 16, 32, 64 and 80. The feature maps produced from the convolutional part are fed into fully connected layers with 128 and 64 nodes, which are followed by the final output layer with a single node that produces the score

TABLE 1 The number of weights in the CNNs as a function of neighbourhood size

		n_z					
		1	**3**	**5**	**7**	**9**	**11**
n_{xy}	33	225,041	225,329	225,617	225,905	226,193	226,481
	49	276,241	276,529	276,817	277,105	277,393	277,681
	65	347,921	348,209	348,497	348,785	349,073	349,361
	81	440,081	440,369	440,657	440,945	441,233	441,521
	97	552,721	553,009	553,297	553,585	553,873	554,161
	113	685,841	686,129	686,417	686,705	686,993	687,281

voxels be μ and σ. Sample a new mean μ_{aug} and new standard deviation σ_{aug} such that

$$\mu_{\text{aug}}/\mu \sim \mathcal{N}(1, a) \tag{3}$$

and

$$\sigma_{\text{aug}}/\sigma \sim \mathcal{N}(1, b), \tag{4}$$

where a and b are arbitrary standard deviation parameters that control the amount of augmentation. Each sample is augmented with respect to intensity with probability 0.5. The flips and rotations by multiples of 90° in the x-y plane are randomly selected such that each of the eight possible transformations (including the original) occur with equal probability.

3.6 | Performance metrics

For assessing classification performance, we use two metrics. First, we use accuracy, which is simply equal to the ratio of the number of correct classifications and the number of total classifications. Second, we use the (volume of the) intersection over (the volume of the) union, abbreviated IoU and also known as the Jaccard index. Generally, it is defined by

$$\text{IoU} = \frac{|\, L \cap P \,|}{|\, L \cup P \,|}, \tag{5}$$

where L is the set of labels (manual segmentation) and P is the set of predictions (automatic segmentation). In this work, P is obtained by thresholding the raw outputs (scores) of a CNN. Ideally, $P = L$ and IoU $= 1$, and it always holds that IoU $\in [0, 1]$. The IoU can be generalized to a class-symmetric metric by averaging the IoUs computed with respect to both classes. The mean intersection over union becomes

$$\text{mIoU} = \frac{1}{2}\left(\frac{|\, L_0 \cap P_0 \,|}{|\, L_0 \cup P_0 \,|} + \frac{|\, L_1 \cap P_1 \,|}{|\, L_1 \cup P_1 \,|}\right). \tag{6}$$

Here, L_0 is the set of samples manually labelled 0 (pores) and L_1 is the set of samples manually labelled 1 (solid)

and analogously for P_0 and P_1 Rahman and Wang.[32] Given that the full data sets do not have a 50/50 class balance, it can be argued that the class-symmetric mIoU is a more fair metric than IoU and takes pores and solid equally well into account.

3.7 | Training and hyperparameter optimization

We train CNNs for neighbourhood sizes $n_{xy} \times n_{xy} \times n_z$ voxels, considering $n_{xy} \in \{33, 49, 65, 81, 97, 113\}$ and $n_z \in \{1, 3, 5, 7, 9, 11\}$ as discussed above. The networks are implemented in Tensorflow 2.4.0.[33] The conventional stochastic gradient descent (SGD) with momentum[34,35] is used to optimize the weights of the networks with respect to the binary cross-entropy loss, defined for N training samples as

$$\mathcal{L}(y_i, f(x_i)) = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(f(x_i)) + (1 - y_i)\log(1 - f(x_i)), \quad (7)$$

where $y_i \in \{0, 1\}$ is the label and $f(x_i) \in [0, 1]$ is the prediction score from the model for training sample i. The prediction score can be interpreted as a probability of the label belonging to class 1. Because the loss decreases the more confident the predictions are (i.e. the closer to 0 or 1 they are) and because it is differentiable with respect to the weights of the CNN, it is a more appropriate optimization target than, for example accuracy or mIoU. Binary cross-entropy loss is the most common loss function for binary classification problems.[36]

Here, we explore multiple hyperparameters. The momentum of the SGD optimizer is found to have little impact on the result, and its value is set to 0.99. Likewise, the batch size is set to 128, although 64 and 256 provides similar performance. The remaining hyperparameters, the ones for which optimizing their values appears to yield the largest performance benefits, are studied in more detail in a random search optimization,[37] executed independently for all architectures: The data augmentation parameters, the learning rate schedule and regularization parameters. The ranges for the hyperparameters in the random search were selected after an initial investigation.

For the data augmentation, the parameters a and b are sampled uniformly in $[0.025, 0.10]$. The learning rate schedule implemented is a step-wise decay learning rate schedule defined as follows: Let $\log_{10}(\mathrm{LR})$ initially be randomly selected in $\{-2.5, -2.25, -2\}$, and let it decrease by 0.25 every kth iteration, for a k randomly selected in $\{4, 6, 8, 10, 12, 14, 16, 18, 20\}$.

Furthermore, we consider several varieties of regularization. First, we implement weight decay, that is l_2 regularization, where a regularization term $c\|\|w\|\|^2$ is added to the loss function to be minimized. In effect, the weights w are forced to decay towards zero, which has a regularizing effect because small weights that can be considered noise contributions are reduced, leading to a higher priority of large-amplitude weights.[38,39] The weight decay parameter c is sampled log-uniformly in $[10^{-7}, 10^{-5}]$ (i.e. $\log_{10} c$ is sampled uniformly in $[-7, -5]$). Second, we consider dropout regularization, both regular dropout and spatial dropout. Regular dropout is a commonly used means of regularization applied after a fully connected layer. During training, outputs are randomly dropped by a probability p_d, which reduces overfitting by encouraging individual nodes to rely less on other nodes.[40] In contrast, spatial dropout is applied after a convolutional (or pooling) layer. During training, entire feature maps are randomly dropped by a probability p_sd, which encourages individual kernels to rely less on other kernels.[41] Here, regular dropout is applied twice, after each of the two fully connected layers, and spatial dropout is applied once, after the final block of convolutional and pooling layers. We randomly sample p_d and p_sd in $[0, 0.6]$. (It is worth mentioning briefly here that another common regularization method, batch normalization,[42] is not investigated because it has been found to not be useful in combination with the Elu activation function.[31]

To reduce the effect of the random seed, which controls not only the parameter values in the random search optimization but also random weight initialization and shuffling of data, 100 separate training instances with unique random seeds are run for all neighbourhood sizes. For all the weights, Glorot/Xavier uniform initialization is used, meaning that the initial values are sampled from a zero-centred uniform distribution, the bounds of which are determined by the input and output dimensions of each layer.[43] Each training instance is run for 40 epochs (iterations over the whole data set). For each neighbourhood size, the best model (over all runs and epochs) with respect to the binary cross-entropy loss of the validation data is selected. In Figure 3, the lowest obtained validation loss obtained for each neighbourhood size is shown. As can be seen, the decrease in loss for increasing n_{xy} is the most monotonic for small values of n_z. This is not surprising, as it reflects the increasing difficulty of attaining good convergence for larger input dimensions. The best-performing model is found for $n_{xy} = 113$ and $n_z = 3$. This result does not imply that more information along the z-direction is not in principle beneficial; indeed, with a more comprehensive hyperparameter search, the best-performing model might very well be found for $n_z > 3$. In Figure 4, the

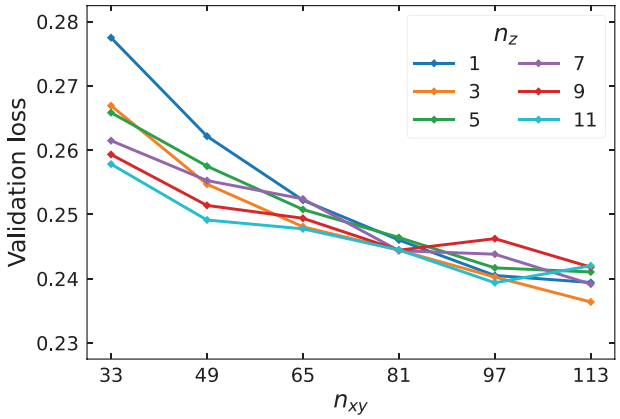

FIGURE 3 Binary cross-entropy loss of the validation data as a function of n_{xy} for different values of n_z

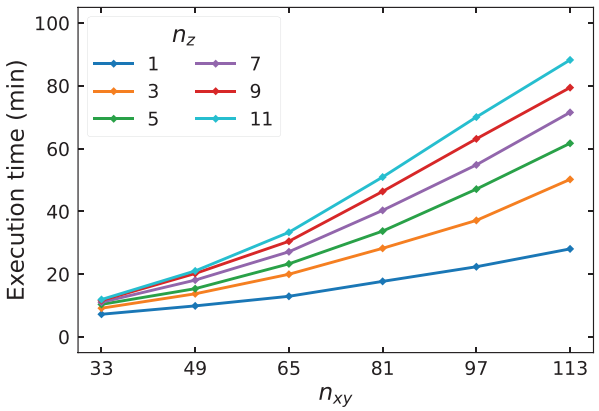

FIGURE 5 Mean execution time for training as a function of neighbourhood size

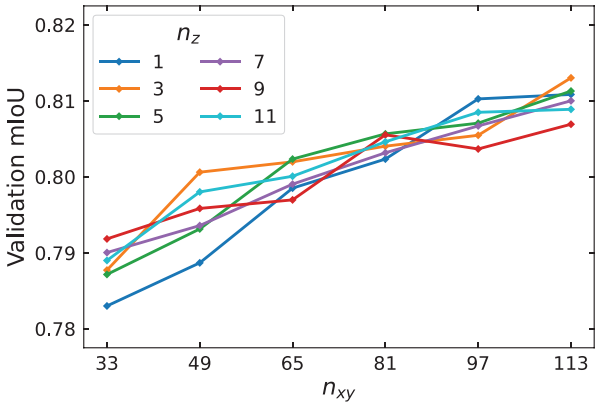

FIGURE 4 mIoU of the validation data as a function of n_{xy} for different values of n_z

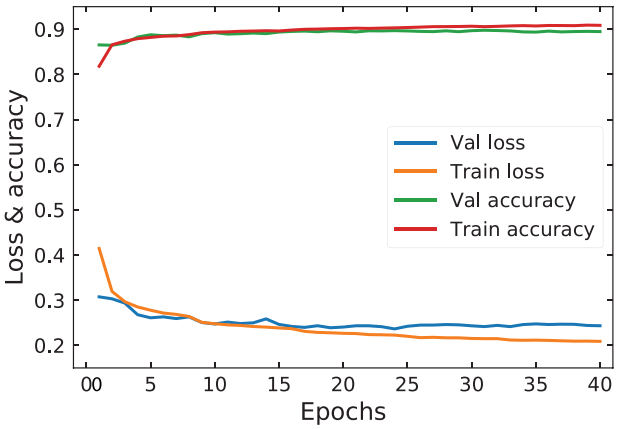

FIGURE 6 Training and validation loss curves and training and validation accuracy for the best-performing model

mIoU of the same models (i.e. chosen with respect to minimizing the loss, not with respect to maximizing the mIoU) is shown. Of these models, the best-performing model in terms of mIoU is found for the same neighbourhood size. For that particular model, $a \approx 0.048$ and $b \approx 0.056$. The initial learning rate is $\log_{10}(\mathrm{LR}_0) = -2$, decreasing every 8th epoch. Furthermore, $c \approx 4.16 \times 10^{-6}$, $p_\mathrm{d} \approx 0.401$ and $p_\mathrm{sd} \approx 0.195$. Training is executed in a cluster environment using a single NVIDIA V100 GPU for each run. In Figure 5, the mean execution time for training as a function of neighbourhood size is shown. The execution time is greatly influenced by the number of weights as provided in Table 1, and not surprisingly, it is monotonically increasing with both n_{xy} and n_z. In Figure 6, the training and validation loss curves and training and validation accuracy for the best-performing model are shown. As can be seen, the validation loss and accuracy have reached a plateau. Furthermore, the training loss and accuracy do not deviate strongly from the validation curves, indicating that overfitting is not severe.

3.8 | Segmentation of the full data sets

The best identified model is used for segmentation of the full data sets. Because of the substantial amounts of data required to be processed (the combined size of all input neighbourhoods for all three data sets is >600 TB in 32-bit precision), neighbourhoods are extracted in batches of 2^{16} (such a batch of neighbourhoods for $n_{xy} = 113$ and $n_z = 3$ constitutes approximately 9 GB in 32-bit precision). Near the edges, the data sets are mirrored. The actual prediction is done in batches of 2^{11}. To facilitate parallel execution in a cluster environment, the code is configured to run the prediction on a single random slice in a random data set; hence, the work is divided into 600 independent runs. The neighbourhoods are extracted while looping sequentially over the slice in a sliding-window approach, illustrated in Figure 7. Prediction of a single slice took on average 30 min on a single NVIDIA V100 GPU, yielding a combined execution time of 100 h per data set. The prediction yields a score array which is then post-processed to produce the

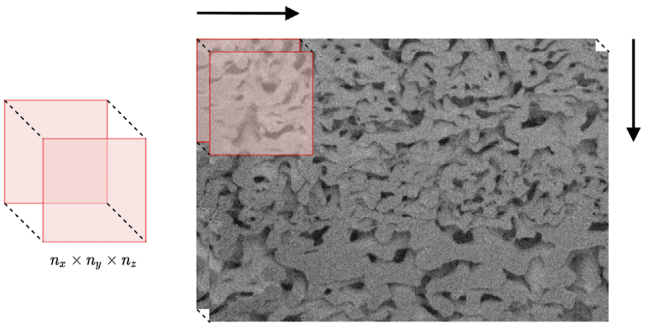

FIGURE 7 Illustration of the sliding window technique utilized when extracting neighbourhoods for voxel-wise classification, depicting full 2D SEM image slices with field of view approximately 22.5 μm × 33.7 μm

final segmentation. If the class balance in the full data sets were 50/50, it would make sense to produce a final segmentation from the raw score array by thresholding at $T = 0.5$. However, we make two observations that lead us to introducing some post-processing steps. First, although adjacent neighbourhoods are overlapping considerably, leading to spatial correlation in the score array, the prediction of each voxel is in principle independent of all other voxels. To make better use of the inherent spatial correlation, we regularize the score array by Gaussian smoothing in the x-y plane with a standard deviation σ. Second, because of the class imbalance, we consider using a threshold $T \neq 0.5$, and it is intuitively clear that $T < 0.5$ is sensible because of the average porosity being considerably lower than 50%. More precisely, we do a first Gaussian smoothing, followed by a first thresholding, followed by a second Gaussian smoothing (of the produced binary array), followed by a second and final thresholding. The parameters σ and T are optimized with respect to mIoU of the validation set (the full, manually segmented regions, as opposed to the subsets previously used for training of the CNNs). (It is worth mentioning here that we also investigate 3D Gaussian smoothing as well as using independent values of σ and T for the two iterations, but that no benefit is found from doing so.) The obtained values are $\sigma \approx 2.63$ pixels and $T \approx 0.402$. Finally, small connected objects (<100 voxels) are removed from both the pores and the solid parts. The results before and after post-processing in terms of mIoU, accuracy and porosity are shown in Tables 2 and 3, respectively. It is worth noting that the porosities in the training, validation and test sets differ more for HPC45 than for the other two data sets. The reason is that the structure gets increasingly heterogeneous with increasing porosity, leading to a larger variance in the porosity of the manually segmented regions. It is evident

TABLE 2 Results before post-processing for the individual data sets as well as combined

	Train	Val	Test
HPC22			
mIoU	0.7601	0.7523	0.7530
Accuracy	0.8951	0.8911	0.8962
Porosity (%), manual	22.07	22.00	20.42
Porosity (%), automatic	27.43	27.57	25.65
HPC30			
mIoU	0.8425	0.8303	0.8085
Accuracy	0.9253	0.9197	0.9075
Porosity (%), manual	29.76	29.14	29.27
Porosity (%), automatic	33.32	32.80	33.65
HPC45			
mIoU	0.7707	0.7597	0.7491
Accuracy	0.8706	0.8638	0.8573
Porosity (%), manual	44.17	49.62	42.20
Porosity (%), automatic	53.22	55.31	49.84
Combined			
mIoU	0.7983	0.7911	0.7774
Accuracy	0.8970	0.8915	0.8870

Note: The results are based on the full, manually segmented regions, as opposed to the training of the CNNs that is carried out on a subset of the manually segmented data

TABLE 3 Results after post-processing for the individual data sets as well as combined

	Train	Val	Test
HPC22			
mIoU	0.7820	0.7723	0.7757
Accuracy	0.9129	0.9084	0.9141
Porosity (%), manual	22.07	22.00	20.42
Porosity (%), automatic	22.01	22.06	20.62
HPC30			
mIoU	0.8537	0.8422	0.8204
Accuracy	0.9335	0.9285	0.9170
Porosity (%), manual	29.76	29.14	29.27
Porosity (%), automatic	29.23	28.76	29.32
HPC45			
mIoU	0.8057	0.7750	0.7761
Accuracy	0.8930	0.8732	0.8760
Porosity (%), manual	44.17	49.62	42.20
Porosity (%), automatic	47.81	49.71	44.34
Combined			
mIoU	0.8217	0.8063	0.7980
Accuracy	0.9131	0.9034	0.9024

Note: The results are based on the full, manually segmented regions, as opposed to the training of the CNNs that is carried out on a subset of the manually segmented data

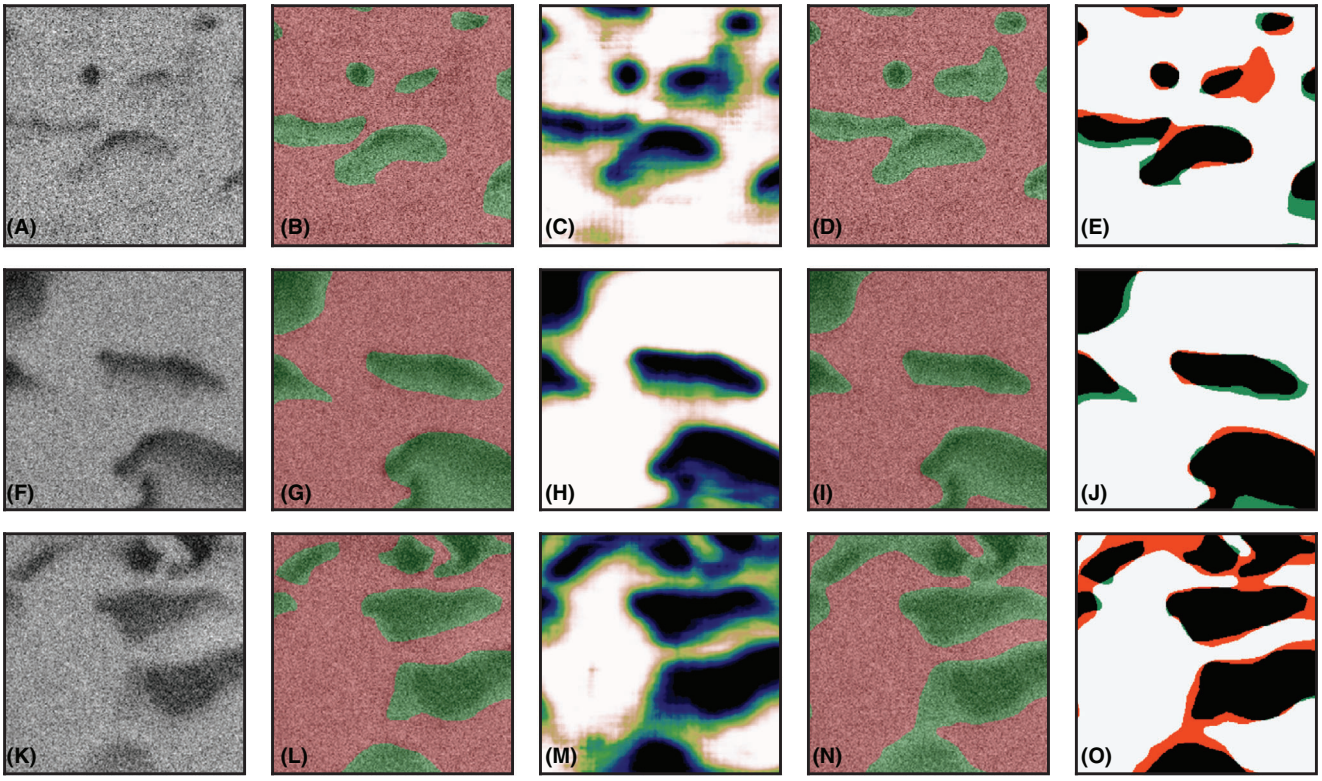

FIGURE 8 Comparison of the manual and automatic segmentations for one region each of the test data from the HPC22 (first row), HPC30 (second row) and HPC45 (third row) data sets. For HPC22, in (A), the image data is shown. In (B), the manual segmentation is superimposed on top of the data, showing pores (green) and solid (red). In (C), the raw score output from the CNN is shown. The raw score can be interpreted as the probability of a voxel of belonging to the solid phase (black ≈ 0, green ≈ 0.5, white ≈ 1). In (D), the automatic segmentation is superimposed on top of the data, showing pores (green) and solid (red). In (E), an overlay of the manual and automatic segmentations is shown, with correctly classified pores (black), correctly classified solid (white), pores incorrectly classified as solid (green) and solid incorrectly classified as pores (orange). The same is shown for HPC30 in (F–J) and for HPC45 in (K–O). The field of view is 2.56 μm × 2.56 μm

that the post-processing substantially improves both mIoU and accuracy for all data sets, and that the post-processing brings the automatic segmentations closer to their manual counterparts in terms of porosity. Because the data split is identical, the results of Table 3 can be directly compared to those of tab. 3 in Röding et al.[20] Compared to the previously obtained results, all mIoU and accuracy values for the validation and test sets are improved, whereas for the training data set, they are improved for the HPC22 data but not for the others. Most notably, for the HPC45 data set the test mIoU is increased from 0.7089 to 0.7761 and the test accuracy is increased from 0.8329 to 0.8760. However, in contrast to the HPC22 and HPC30 data sets where the test porosity is improved compared to Röding et al.,[20] for the HPC45 it actually gets worse with an increase from 42.80% to 44.34%. This is to some extent likely to be an effect of the limited data set size, but also illustrates that the correlation between on the one hand mIoU and accuracy and on the other hand the fraction of samples assigned to the different classes is not always direct. By computing individual porosities from the 100 square regions, merging training, validation and test data, and assuming that the values are independent, we can investigate differences between the porosities by means of two-sample t-tests. The results indicate that there are no significant differences between the porosities from the manual and automatic segmentations ($p \approx 0.98$, 0.77 and 0.14). Nevertheless, given that the manually segmented regions cover ∼0.43 % of the full data sets, we cannot know that the same would hold for the full data sets.

In Figure 8, one example region from each of the test sets are shown, including a comparison between the manual and automatic segmentations. Finally, in Figure 9, a single full slice from each of the three data sets is shown, comparing the data and the automatic segmentation. Whereas this figure does not facilitate quantitative assessment of the results, it demonstrates that the segmentations appear reasonable also outside the manually segmented regions. The full data sets and segmentations are available online.[23]

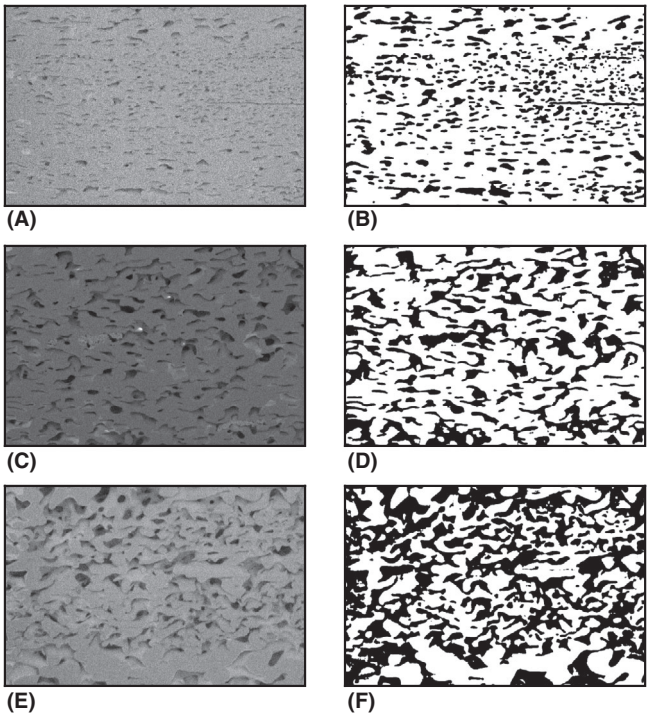

FIGURE 9 The data and the corresponding segmentation for one slice from each of the three data sets, showing (A–B) HPC22, (C–D) HPC30 and (E–F) HPC45. The field of view is approximately 22.5 μm × 33.7 μm

4 | CONCLUSION

We have developed a segmentation method based on CNNs for FIB-SEM data. The CNNs are trained on data acquired from three samples of porous films produced from EC/HPC polymer blends used for controlled release applications. The CNNs are conventional multi-channel 2D CNNs using a sliding window-type approach, where each voxel is classified as solid or pore using a neighbourhood centred around the voxel of interest as input. A large number of neighbourhood sizes were explored together with other parameters such as size and degree of regularization of the model, and the best-performing model was selected for final segmentation of the entire data sets. The results are in good agreement with manual segmentations. In particular, in terms of the two performance metrics used, accuracy and mIoU, for the validation and test data, the results are superior to earlier results produced using a random forest classifier on the same data.[20] Furthermore, the porosities of the identified porous structures are in good agreement with the expected porosities. Segmentation using the proposed method is rather time-consuming, but this should be acceptable given that the experiments are also very time-consuming. But indeed, there is a trade-off between mainly the neighbourhood size, but also other parameters such as the model size, and computational speed. Nevertheless, we judge that improved performance compared to, for example random forest classifiers should still be preferred as the precise characterization and understanding of these materials is very important for the application. It is worth noting that the network architectures, hyperparameters and weights (and also the post-processing) could in principle be optimized with respect to each individual data set HPC22, HPC30 and HPC45 (and hence to the individual porosities). This would likely improve the results somewhat for each of the data sets, considering that the geometry of the pores and the amount of subsurface information changes with changing porosity. However, this would have to be done at the expense of obtaining less robust CNNs, trained on less data and not having learned to cope with changes in porosity.

There are several promising avenues for further work. First, segmentation by multiple experts could be used to quantify the degree of uncertainty in the manual segmentation. Second, semi-supervised approaches could be used to make better use of the very large part of the data set that is not manually segmented. Third, more neighbourhood sizes could be considered, as the investigation indicates that larger sizes could be beneficial. Fourth, other losses, such as mIoU-based losses and losses that take the estimated porosity into account, could further improve the result. Fifth, using sample weighting to emphasize the border between pores and solid could put additional focus on regions that are difficult to segment. Sixth, more sophisticated data augmentation could be utilized to obtain more variation in the data set. Seventh, ensembles of CNNs might be able to produce more accurate segmentations than a single CNN. Eighth, more complex geometrical features than porosity could be used for assessment of segmentation quality, such as pore size distributions, although this would not be possible without more extensive manual segmentation of larger, three-dimensional regions.

Finally, we provide the data and software open access[23] to facilitate development of new FIB-SEM segmentation methods.

ACKNOWLEDGEMENTS

We acknowledge Anna Olsson and Christian von Corswant at AstraZeneca Gothenburg for discussions and for providing the samples and Chalmers Material Analysis Laboratory for their support of microscopes. We acknowledge the financial support of the Swedish Research Council (Grant number 2016-03809), the Swedish Research Council for Sustainable Development (Grant number 2019-01295), the Swedish Foundation for Strategic Research (the project 'Material structures seen through microscopes and statistics'), and Chalmers Area of Advance Materials Science. A GPU used for part of

this research was donated by the NVIDIA Corporation. The computations were in part performed on resources at Chalmers Centre for Computational Science and Engineering (C3SE) provided by the Swedish National Infrastructure for Computing (SNIC).

ORCID

Magnus Röding https://orcid.org/0000-0002-5956-9934

REFERENCES

1. Barman, S., Rootzén, H., & Bolin, D. (2019). Prediction of diffusive transport through polymer films from characteristics of the pore geometry. *AIChE Journal*, *65*, 446–457.
2. Gebäck, T., Marucci, M., Boissier, C., Arnehed, J., & Heintz, A. (2015). Investigation of the effect of the tortuous pore structure on water diffusion through a polymer film using lattice Boltzmann simulations. *Journal of Physical Chemistry B*, *119*, 5220–5227.
3. Marucci, M., Ragnarsson, G., von Corswant, C., Welinder, A., Jarke, A., Iselau, F., & Axelsson, A. (2011). Polymer leaching from film coating: Effects on the coating transport properties. *International Journal of Pharmaceutics*, *411*, 43–48.
4. Tiwari, G., Tiwari, R., Sriwastawa, B., Bhati, L., Pandey, S., Pandey, P., & Bannerjee, S. K. (2012). Drug delivery systems: An updated review. *International Journal of Pharmaceutical Investigation*, *2*, 2–11.
5. Wassén, S., Bordes, R., Gebäck, T., Bernin, D., Schuster, E., Lorén, N., & Hermansson, A.-M. (2014). Probe diffusion in phase-separated bicontinuous biopolymer gels. *Soft Matter*, *10*, 8276–8287.
6. Inkson, B. J., Mulvihill, M., & Möbus, G. (2001). 3D determination of grain shape in a FeAl-based nanocomposite by 3D FIB tomography. *Scripta Materialia*, *45*, 753–758.
7. Holzer, L., Indutnyi, F., Gasser, P. H., Münch, B., & Wegmann, M. (2004). Three-dimensional analysis of porous $BaTiO_3$ ceramics using FIB nanotomography. *Journal of Microscopy*, *216*, 84–95.
8. Giannuzzi, L. A., & Stevie, F. A. (2005). *Introduction to focused ion beams: Instrumentation, theory, techniques and practice*. Springer.
9. Joos, J., Carraro, T., Weber, A., & Ivers-Tiffée, E. (2011). Reconstruction of porous electrodes by FIB/SEM for detailed microstructure modeling. *Journal of Power Sources*, *196*, 7302–7307.
10. Fager, C., Röding, M., Olsson, A., Lorén, N., von Corswant, C., Särkkä, A., & Olsson, E. (2020). Optimization of FIB-SEM tomography and reconstruction for soft, porous, and poorly conducting materials. *Microscopy and Microanalysis*, *26*, 837–845.
11. Blayvas, I., Bruckstein, A., & Kimmel, R. (2006). Efficient computation of adaptive threshold surfaces for image binarization. *Pattern Recognition*, *39*, 89–101.
12. Jørgensen, P. S., Hansen, K. V., Larsen, R., & Bowen, J. R. (2010). A framework for automatic segmentation in three dimensions of microstructural tomography data. *Ultramicroscopy*, *110*, 216–228.
13. Prill, T., Schladitz, K., Jeulin, D., Faessel, M., & Wieser, C. (2013). Morphological segmentation of FIB-SEM data of highly porous media. *Journal of Microscopy*, *250*, 77–87.
14. Salzer, M., Spettl, A., Stenzel, O., Smått, J.-H., Lindén, M., Manke, I., & Schmidt, V. (2012). A two-stage approach to the segmentation of FIB-SEM images of highly porous materials. *Materials Characterization*, *69*, 115–126.
15. Salzer, M., Thiele, S., Zengerle, R., & Schmidt, V. (2014). On the importance of FIB-SEM specific segmentation algorithms for porous media. *Materials Characterization*, *95*, 36–43.
16. Taillon, J. A., Pellegrinelli, C., Huang, Y.-L., Wachsman, E. D., & Salamanca-Riba, L. G. (2018). Improving microstructural quantification in FIB/SEM nanotomography. *Ultramicroscopy*, *184*, 24–38.
17. Moroni, R., & Thiele, S. (2020). FIB/SEM tomography segmentation by optical flow estimation. *Ultramicroscopy*, *219*, 113090.
18. Fend, C., Moghiseh, A., Redenbach, C., & Schladitz, K. (2021). Reconstruction of highly porous structures from FIB-SEM using a deep neural network trained on synthetic images. *Journal of Microscopy*, *281*, 16–27.
19. Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention* (pp. 234–241).
20. Röding, M., Fager, C., Olsson, A., von Corswant, C., Olsson, E., & Lorén, N. (2021). Three-dimensional reconstruction of porous polymer films from FIB-SEM nanotomography data using random forests. *Journal of Microscopy*, *281*, 76–86.
21. Lennefors, M., & Visuri, W. J. (2020). *Deep learning for semantic segmentation of FIB-SEM volumetric image data* (Master's thesis). Chalmers University of Technology.
22. Skärberg, F. (2020). *Convolutional neural networks for semantic segmentation of FIB-SEM volumetric image data* (Master's thesis). University of Gothenburg.
23. Skärberg, F., Fager, C., Olsson, A., von Corswant, C., Olsson, E., Lorén, N., & Röding, M. (2020). Zenodo. https://doi.org/10.5281/zenodo.4317170.
24. Fager, C., & Olsson, E. (2018). Soft materials and coatings for controlled drug release. In V. Uskokovic & D. P. Uskokovic (Eds.), *Nanotechnologies in preventive and regenerative medicine* (pp. 244–259). Elsevier.
25. Jansson, A., Boissier, C., Marucci, M., Nicholas, M., Gustafsson, S., Hermansson, A.-M., & Olsson, E. (2014). Novel method for visualizing water transport through phase-separated polymer films. *Microscopy and Microanalysis*, *20*, 394–406.
26. Siepmann, F., Hoffmann, A., Leclercq, B., Carlin, B., & Siepmann, J. (2007). How to adjust desired drug release patterns from ethylcellulose-coated dosage forms. *Journal of Controlled Release*, *119*, 182–189.
27. Marucci, M., Hjärtstam, J., Ragnarsson, G., Iselau, F., & Axelsson, A. (2009). Coated formulations: New insights into the release mechanism and changes in the film properties with a novel release cell. *Journal of Controlled Release*, *136*, 206–212.
28. Schneider, C. A., Rasband, W. S., & Eliceiri, K. W. (2012). NIH Image to ImageJ: 25 years of image analysis. *Nature Methods*, *9*, 671–675.
29. Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, *61*, 85–117.
30. Rawat, W., & Wang, Z. (2017). Deep convolutional neural networks for image classification: A comprehensive review. *Neural Computation*, *29*, 2352–2449.

31. Clevert, D.-A., Unterthiner, T., & Hochreiter, S. (2016). Fast and accurate deep network learning by exponential linear units (ELUs). arXiv preprint arXiv:1511.07289.
32. Rahman, M. A., & Wang, Y. (2016). Optimizing intersection-over-union in deep neural networks for image segmentation. In *International symposium on visual computing* (pp. 234–244).
33. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., & Zheng, X. (2015). *TensorFlow: Large-scale machine learning on heterogeneous systems*. Software Retrieved from https://www.tensorflow.org/
34. Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT '2010* (pp. 177–186). Springer.
35. Qian, N. (1999). On the momentum term in gradient descent learning algorithms. *Neural Networks*, *12*, 145–151.
36. Yamashita, R., Nishio, M., Do, R. K. G., & Togashi, K. (2018). Convolutional neural networks: An overview and application in radiology. *Insights into Imaging*, *9*, 611–629.
37. Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, *13*, 281–305.
38. Krogh, A., & Hertz, J. A. (1992). A simple weight decay can improve generalization. In *Advances in neural information processing systems* (pp. 950–957).
39. van Laarhoven, T. (2017). L2 regularization versus batch and weight normalization. *CoRR*, abs/1706.05350.
40. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*, 1929–1958.
41. Tompson, J., Goroshin, R., Jain, A., LeCun, Y., & Bregler, C. (2015). Efficient object localization using convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 648–656).
42. Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167.
43. Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (pp. 249–256).

How to cite this article: Skärberg F, Fager C, Mendoza-Lara F, et al. Convolutional neural networks for segmentation of FIB-SEM nanotomography data from porous polymer films for controlled drug release. *Journal of Microscopy*. 2021;1-13. https://doi.org/10.1111/jmi.13007