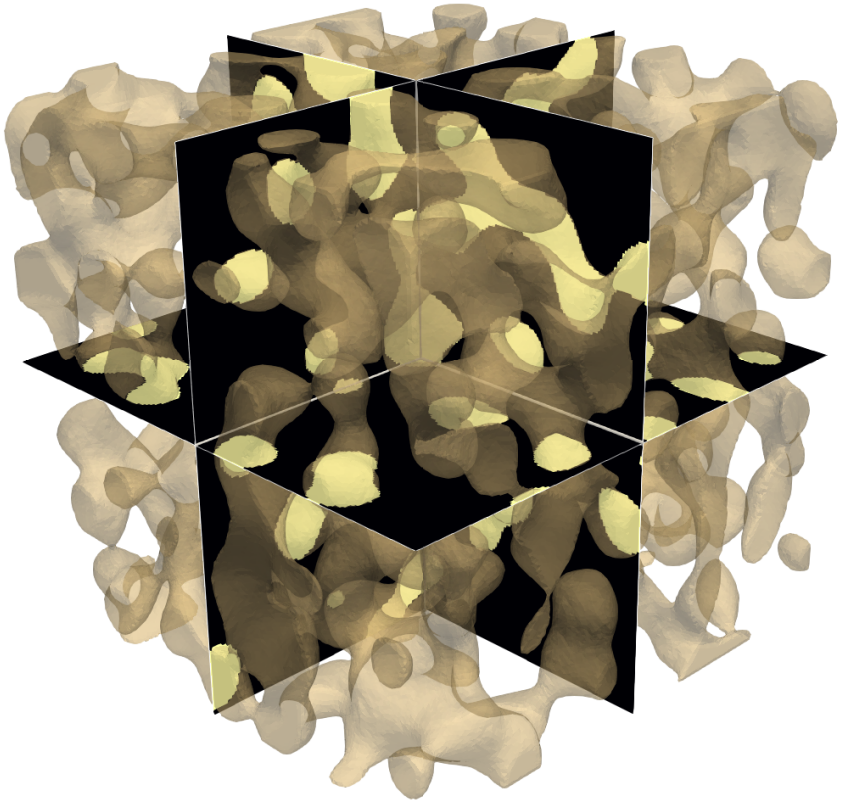


Prediction of mass transport properties in 3D microstructures using 2D CNNs

Master's thesis in Engineering Mathematics and Computational Science

Sævar Óli Valdimarsson

DEPARTMENT OF MATHEMATICAL SCIENCES

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2022
www.chalmers.se

Master's thesis 2022

Prediction of mass transport properties in 3D microstructures using 2D CNNs

Sævar Óli Valdimarsson

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematical Sciences
Chalmers University of Technology
Gothenburg, Sweden 2022

Prediction of mass transport properties in 3D microstructures using 2D CNNs
SÆVAR ÓLI VALDIMARSSON

Supervisor: Magnus Röding, Research Institutes of Sweden, RISE
Examiner: Aila Särkkä, Mathematical Sciences

Master's Thesis 2022
Department of Mathematical Sciences
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: A virtual 3D microstructure with three orthogonal cross-sections.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2022

Prediction of mass transport properties in 3D microstructures using 2D CNNs
SÆVAR VALDIMARSSON
Department of Mathematical Sciences
Chalmers University of Technology

Abstract

Porous materials and the relationship between their 3D microstructure and their mass transport properties is of interest in multiple fields. To analyse this relationship and build an understanding of it requires a great quantity of data, but obtaining experimental 3D data is difficult and expensive. An alternative is to generate virtual microstructures and simulate their mass transports, which can then be used to estimate the relationship. 2D experimental data is easier to obtain and work with than 3D experimental data, e.g. it requires less storage space and memory. It is of interest to investigate models that can estimate mass transport properties of 3D microstructures from 2D data. In this work, 2D data is extracted from a pre-existing 3D virtual microstructure dataset and the viability of using 2D convolutional neural networks (CNNs) to predict the mass transport properties is explored.

Keywords: microstructure, mass transport properties, convolutional neural network, 2D, 3D.

Acknowledgements

I would like to thank and acknowledge my supervisor Magnus Röding at RISE and examiner Aila Särkkä, as their help has been invaluable. Their guidance and feedback carried me through all stages of the thesis.
I would also like to thank my family, who have always been supportive and there for me.

Sævar Óli Valdimarsson, Gothenburg, 12 2022

Contents

List of Figures **xi**

List of Tables **xiii**

1 Introduction **1**

2 Background **3**
2.1 Mass Transport Properties 3
2.2 The Lattice-Boltzmann Method 4
2.3 Machine learning 5
2.3.1 Artificial Neural Networks 5
2.3.2 Convolutional Neural Networks 7

3 Data **9**
3.1 Generation Methods 9
3.2 Training Dataset 12
3.3 2D Data Extraction 13

4 Model Architecture **17**
4.1 Architecture Design 17
4.2 Examined Models 19
4.3 Data Augmentation 23

5 Discussion and Results **27**
5.1 Training Sessions 27
5.2 Results 28

6 Conclusion **33**

Bibliography **35**

List of Figures

2.1 Example of D2Q9 lattice. Velocity vectors are denoted by c, the stationary vector c_0 is not depicted. 4
2.2 An individual neuron. It has inputs $x_1, \ldots, x_N$, the corresponding input weights $w_1, \ldots, w_N$, bias b, value y and output that is regulated by the activation function f. 5
2.3 An example of an artificial neural network (ANN) with multiple layers of neurons without the biases. It has an input layer followed by L hidden layers which terminate in the output layer. The processing of the data takes place in the hidden layers. 6
2.4 An example of convolution where a convolution is being applied to a matrix with a kernel to extract features. The kernel (blue border) iterates over the input matrix and performs element wise multiplication and sums up the results. The sum is the extracted feature of that 3×3 area and is stored in the feature map. 7
2.5 A simple example of average pooling. 8

3.1 9 types of microstructures 10
3.2 3D sample of a microstructure generated using the smoothed hard ellipsoid method. This sample will be used to demonstrate the extraction methods used. 13
3.3 A demonstration of how a 2D binary slice is extracted from a 3D volumetric cube. 13
3.4 The 2D cross-section obtained from the 3D volumetric cube, showing pores (black) and solids (light yellow). 14
3.5 A demonstration how three orthogonal cross-sections are extracted from a 3D volumetric cube. 14
3.6 Three binary cross-sections extracted from a volumetric cube. The cross-sections are orthogonal to each other. 15
3.7 A volumetric cube rendered with orthographic projection (left) with the corresponding obtained density map (right), showing regions of high density (bright) and low density (dark). To show the transition, a transparent version of the orthographic projection has been superimposed over the density map (center). 15
3.8 Three extracted density maps, which are orthogonal to each other, showing regions of high density (bright) and low density (dark). . . . 16

4.1 Model architecture diagram. 19

4.2 Learning rate changes over epochs. 19
4.3 Examples of the two types of input data used in training 2D CNN models. 20
4.4 Model 1 is trained on a single 2D binary cross-section. The cross-section is perpendicular to direction of transport. 20
4.5 Model 2 is trained on a single 2D binary cross-section. Here, two cross-sections parallel to the direction of transport in the same orthogonal set are depicted. One of them is selected at random during training. 21
4.6 Model 3 is trained on three orthogonal cross-sections extracted from the same microstructure. The input is a $3 \times 192 \times 192$ matrix, which is obtained by concatenating the cross-sections together. 21
4.7 An example of how a pseudo-RGB image could be constructed from an orthogonal cross-section set. 22
4.8 Model 4 is trained on a single 2D density map. The map is perpendicular to direction of transport. 22
4.9 Model 5 is trained on three orthogonal density maps extracted from the same microstructure. The input is a $3 \times 192 \times 192$ matrix, which is obtained by concatenating the maps together. 23
4.10 Model 6 is a combination of model 3 and 4, and trains on three orthogonal cross-sections and a density map. The input is a $4 \times 192 \times 192$ matrix, which is obtained by concatenating the 2D data together. 23
4.11 Augmentations used during training. The example binary slice has been selected for its recognizable features. 25

5.1 An example of training and validation loss over 2 training sessions from model 5. The visible spikes are from the changing learning rate. 28
5.2 Scatter plots of diffusivity with true values M and predicted values $\hat{M}$. 30
5.3 Scatter plots of permeability with true values κ and predicted values $\hat{\kappa}$. 30

List of Tables

3.1 Sample memory sizes in binary file format. Individual 2D data is included for comparison with the final orthogonal sets. The downscaled cubes used by Prifling et al [9] is included for comparison. Binary values are represented by integers, integers are stored with 1 byte (8 bits) and floating numbers with 4 bytes (32 bits). 16

4.1 Model parameters. N is the number of channels or layers in the input data. 18
4.2 Learning rates. 19
4.3 Examined models and corresponding data arrangements 23

5.1 Diffusivity results. The corresponding 3D CNN model by Prifling et al [9] is included for comparison. 29
5.2 Permeability results. The corresponding 3D CNN model by Prifling et al [9] is included for comparison. 29

1
Introduction

Porous materials are everywhere, from simple coffee filters to aquifers that allow the flow of groundwater, and they are of interest in multiple fields: pharmaceuticals [1], catalysis, filtration and separation [2], and geological events [3].They have the property of allowing or restricting the flow of liquids and gases through themselves. This movement of fluids is called mass transport or mass transfer. However it is often not the porous material itself that is of interest but the matter being transported through: how the matter behaves and how it is affected by the material.

The aforementioned coffee filter provides a good example of this. Its purpose is to allow coffee to pass through it but not the coffee grounds, which enables the brewing process. Therefore, the functionality of the filter relies heavily on the mass transport properties of the filter.

Understanding the relationship between the structure of porous materials and their mass transport properties is of interest in various fields, whether it is describing mass transport in existing materials or designing new materials that control mass transport to achieve certain effects.

The most fundamental characteristic of porous materials is their porosity, i.e. the fraction of empty space in the material. The spatial arrangement of the empty space or the pores in the material also has a great effect on the mass transport. Due to the small size of the pores, often at microscopic scale, the structures are referred to as microstructures. The relationship between microstructural geometry and mass transport properties is highly complex. Any attempts to understand it requires high quality data to learn from and analyse. Geometric characteristics have been used alongside statistical methods to explore this relationship [1], but another approach has been through machine learning methods, especially with the advancements of convolutional neural networks in image recognition [4].

Large amounts of varied data is needed for such methods and 3D data is difficult and expensive to acquire experimentally. Instead, virtual microstructures that have similar characteristics as experimental structures are generated and then their mass transport is calculated using simulations [5–10].

Experimental 2D data can be easier to acquire than experimental 3D data [11]. To apply information learned from virtual data on experimental 2D data, 3D data must be simulated to compute the mass transport properties. Then extract the 2D data to mimic how experimental 2D data is acquired. 2D simulations [5–7] do not provide the required 3D context that 3D simulations do [8, 9].

Faster prediction methods that give mass transport properties with reasonable accuracy would allow for improvements in workflow. The faster methods can be used to get approximate values, before committing to a slower prediction method

that produces more accurate results [12].

Prifling at al [9] generated a large dataset of virtual microstructures and simulated their mass transport properties. Then they used different methods to predict the mass transport properties, including 3D CNNs. That paper forms the basis of the current work as the same dataset is used here.

In this work, prediction of mass transport properties from 2D data extracted from 3D data, through 2D CNNs will be investigated. 2D data is extracted from the 3D data in multiple ways. The dataset that will be used in this work was generated by Prifling et al [9]. It is large and varied which is highly useful for training machine learning models.

The report is organized as follows. In chapter 2, the relevant background information needed to understand the thesis is explained. It covers a brief explanation of the relevant mass transport properties, a numerical method used to calculate the mass transport and finally some machine learning topics. Chapter 3 introduces the dataset used, the methods used to generate it and how the dataset is used in training, with details how 2D data is extracted from the 3D dataset. In chapter 4, the implementation of the models, as well as how the models are used with the extracted 2D data, are presented. Additionally, data augmentation methods used in training are presented. Chapter 5 details how the training was carried out, with the results presented and discussed. Finally, the conclusion is presented, with some ideas for further work in chapter 6.

2
Background

In this chapter, some background information on relevant mass transport properties and the numerical method used to compute them will be discussed. Additionally an introduction to machine learning and neural networks will be given.

2.1 Mass Transport Properties

The mass transport properties that will be investigated in this work are diffusivity and permeability.

Diffusivity is a measurement of the rate at which particles diffuse in a solution. In regards to porous materials diffusivity describes the diffusion of particles through the porous space. Diffusion is described by the diffusion equation,

$$\mathbf{J} = -D_0 \nabla c, \tag{2.1}$$

where $\mathbf{J} = (J_x, J_y, J_z)$ is the diffusive flux, D_0 the free diffusion coefficient and ∇c the concentration gradient. The transport in porous material is commonly characterized by the effective diffusion coefficient D_{eff}, that describes the rate of diffusion in the presence of obstacles. It is defined by

$$\langle J \rangle = -D_{\text{eff}} \tilde{\nabla} c, \tag{2.2}$$

which describes diffusion driven by a concentration difference $c_{\text{out}} - c_{\text{in}}$ at the inlet and outlet, on either side of a porous structure of length L. $\langle J \rangle$ is the diffusive flux in the first dimension, averaged out over the porous structure and $\tilde{\nabla} c = (c_{\text{out}} - c_{\text{in}})/L$. The diffusivity M is the ratio between the effective and free diffusion coefficients,

$$M = \frac{D_{\text{eff}}}{D_0}. \tag{2.3}$$

Permeability is the ability of a material to pass fluid through it. The permeability is obtained by calculating fluid flow using the Navier-Stokes equations and then using Darcy's law to extract the permeability k [13],

$$\bar{u} = \frac{k \Delta p}{\mu d}, \tag{2.4}$$

where $\bar{u}$ is the average velocity, Δp is the applied pressure difference, μ is the dynamic viscosity and d is the length of the microstructure in the flow direction.

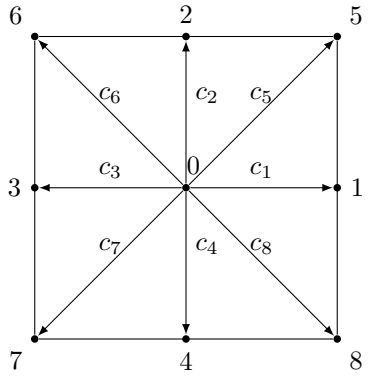


Figure 2.1: *Example of D2Q9 lattice. Velocity vectors are denoted by c, the stationary vector c_0 is not depicted.*

2.2 The Lattice-Boltzmann Method

The Lattice Boltzmann method (LBM) is a numerical method that can be used to simulate fluid flow (liquid or gas). From such simulations it is possible to extract the mass transport properties using the equations mentioned in the previous section. One way to explain how LBM works is to briefly describe the categories of fluid flow numerical solvers and how LBM fits within that scope.

There are multiple numerical methods that can be used to describe and solve fluid flow. One way to organize the methods is to group them into microscopic, mesoscopic and macroscopic scales [14].

At microscopic scale, individual molecules or particles are simulated, and positions, velocities and collisions are tracked. Particle-based solvers are computationally expensive and usually used for problems on a relatively small scale. At macroscopic scale, the fluid is treated as a continuum, where the individual molecules are not modelled but rather the fluid velocity and density. The Navier-Stokes equations are often used to describe that continuum. The mesoscopic scale is between the microscopic and macroscopic scales, and tracks distributions or representative collections of molecules. This is the scale that concerns the LBM.

LBM originates from the lattice gas method, a particle-based solver. Gas particles move between nodes in a lattice, represented by velocity vectors. This movement is governed by a streaming rule. When two or more particles meet in a node they are redistributed in a way that conserves mass and momentum within the node. This is governed by a collision rule which alongside the streaming rule describes the behavior of the particles.

Instead of tracking individual particles and their velocities, the LBM uses a distribution function to track the density of particles with specific velocities, at specific times. There is still streaming and collisions but they are handled differently within the context of particle densities. Lattices are denoted by $DmQn$ where m represents the dimension of the problem and n represents the number of velocities. For example $D2Q9$ represents a lattice with 2 dimensions and where each lattice node has 9 velocities. A lattice node of $D2Q9$ can be seen in Figure 2.1. 8 velocities can be seen but the velocity 0 which denotes stationary particles is not depicted.

If the reader wants to learn more about this topic, they are directed to the book "The Lattice Boltzmann Method: Principles and Practice" by Timm Krüger et al [14].

2.3 Machine learning

Machine learning is a group of methods where an algorithm builds a model that 'learns' to perform a specific task, instead of explicitly implementing it from the start. This enables defining tasks implicitly and then letting the model figure out how to carry it out. There are different ways to implement the learning process and each one has a impact on what the model can learn. The typical way is to organise machine learning methods into supervised, unsupervised and reinforcement learning. The one of interest here is supervised learning, where the model is given labelled data, i.e. the input data is paired with output data used to evaluate the performance and then to improve it. It is called supervised learning since there is a feedback loop where known and predicted outputs are compared. This approach has seen great benefits when there are large amounts of labelled data that can be used to direct the learning process.

Supervised learning is commonly split into classification and regression. Classification relates to categorising inputs into discrete classes. Regression relates to predicting numerical output values from based on input values. This latter case correpsonds to predicting mass transport properties from microstructures and will be used here.

2.3.1 Artificial Neural Networks

One of the most popular machine learning methods is artificial neural networks (ANNs). It was inspired by the behavior of biological neurons and how signals propagate through the network. Each artificial neuron has one or more inputs which may originate from other neurons or the input dataset, see Figure 2.2. These inputs are weighted to give them different importance. The weighted inputs are summed up together alongside the neuron's default value, which is called bias. Finally an activation function is used to transform the value before passing it on.

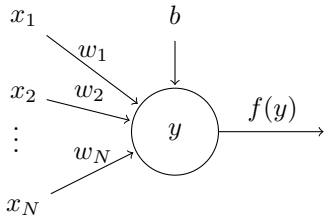


Figure 2.2: *An individual neuron. It has inputs $x_1, \ldots, x_N$, the corresponding input weights $w_1, \ldots, w_N$, bias b, value y and output that is regulated by the activation function f.*

This neuron output is calculated by Eq. 2.5, where x_i are the input values, w_i the input weights, b the bias and f the activation function.

$$f(y) = f\left(\sum_i w_i x_i + b\right) \tag{2.5}$$

These neurons are arranged in multiple layers, where the neurons in the previous layer provide the input for the next one, see Figure 2.3. When all neurons in a layer connect to all the neurons in the next layer, the layers are referred to as dense. The initial input propagates through the layers to the final output neuron which returns the predicted output.

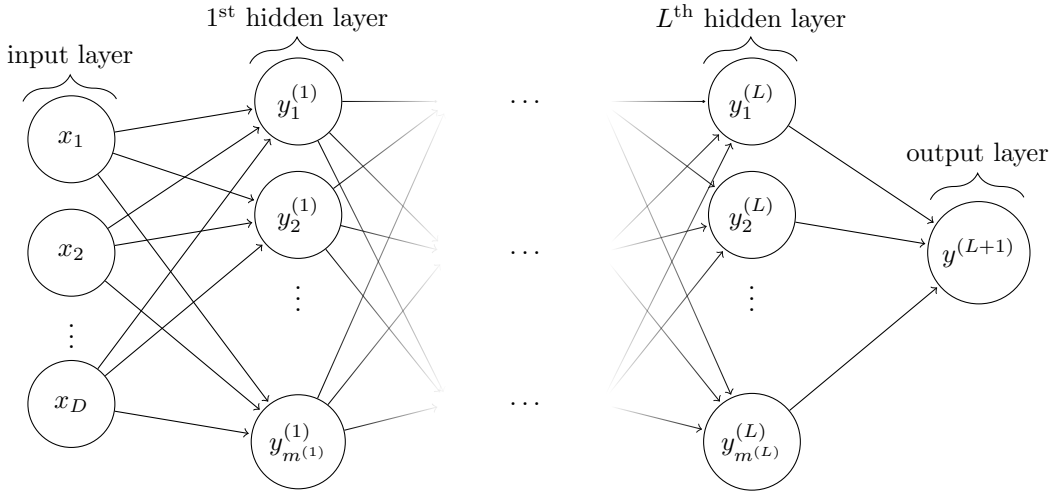


Figure 2.3: *An example of an artificial neural network (ANN) with multiple layers of neurons without the biases. It has an input layer followed by L hidden layers which terminate in the output layer. The processing of the data takes place in the hidden layers.*

The evaluation of the network is done by comparing the predicted values to the true values through a loss function, which calculates the error of the model. The model is then improved by attempting to minimise that error which is how the ANN learns. The required adjustments, i.e. the gradients are calculated through backpropagation. Different from regular propagation of values through the network, i.e. from the input to the output, backpropagation traverses the network backwards, calculating the gradient of each weight and bias. The gradients are adjusted by a *learning rate* and a *momentum* and then applied through a gradient descent method. Learning rate specifies the magnitude of the applied change. Higher values mean faster learning by the model but too large of a learning rate can also cause the model to diverge, i.e. perform badly and never recover. Momentum is inertia applied to the gradient to smooth out behavior characteristics of gradient descent.

As a loss function provides the value being optimised for, it has a big impact on the learning process. One loss function often used is the mean squared error:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(X_i - \widehat{X}_i\right)^2. \tag{2.6}$$

The function is quadratic which provides greater incentive to improve primarily the worst errors. With a linear loss function, improvements of the worst error and a

minor error would result in the same overall improvement, which may introduce outliers and worsen generalisation.

Each training iteration over the test data is called an epoch. Training time is often discussed in terms of the number of epochs. The number of epochs that is required for convergence varies. Some approaches fix the amount of epochs or time spent on training, others stop the training when the model stops improving.

When an ANN model is trained on a dataset it may learn specifically the training input, i.e. overfit to training data and perform badly on any previously unseen input. For this reason, the dataset is typically split into three sets: training, validation, and test sets. The training set is used for calculating the gradient and adjusting the weights. The validation set is used to assess the progress of the training but is not used directly in the training, which prevents over-fitting. To evaluate the performance of the model, neither the training nor validation set can be used. That is the purpose of the test set which has not been used during any steps of the training.

2.3.2 Convolutional Neural Networks

Convolutional neural networks (CNNs) have been used to great effect in image recognition [15] and have been applied successfully in recognizing pertinent features from the structural topology of images[4]. Convolutional neural networks are ANNs that have been extended by the addition of convolution and pooling layers.

A convolution layer applies convolution to the input with a kernel, which extracts features into a feature map. An example of convolution can be seen in Figure 2.4. It is difficult to say which features are important and useful for the neural network. For this reason, multiple trainable kernels are used to extract multiple feature maps.

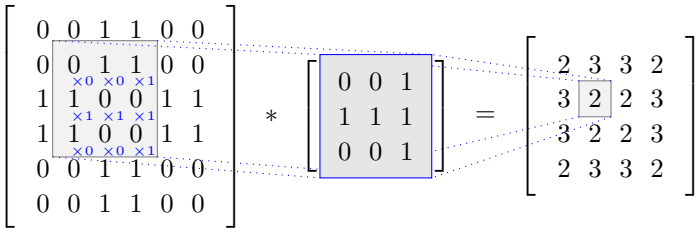


Figure 2.4: *An example of convolution where a convolution is being applied to a matrix with a kernel to extract features. The kernel (blue border) iterates over the input matrix and performs element wise multiplication and sums up the results. The sum is the extracted feature of that* 3×3 *area and is stored in the feature map.*

These feature maps can then be used as inputs for the artificial neural network, as the features add emphasis on what may be important to the network. A convolution by itself is a linear operation. Applying multiple convolutions to an input does increase potential expressiveness over a single convolution. However, applying a non-linear activation function after each convolution makes layering them more expressive and capable of extracting nonlinear features. Therefore it is beneficial to apply multiple convolutions to extract nonlinear features and refine the feature map. These features can then be consolidated with pooling layers. A pooling layer

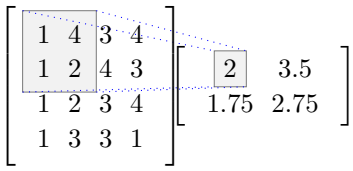


Figure 2.5: *A simple example of average pooling.*

compresses a feature map into a smaller feature map with the intention of conserving the most important features and decreasing the amount of data to a level that is feasible to be trained on by the neural network. The pooling is performed by combining multiple values into a single one, e.g. taking the highest value or taking the average. This convolution and pooling step is the main feature of CNNs and can be repeated multiple times to further refine and compress the feature maps.

3
Data

Supervised learning needs a large and varied dataset to produce a model that performs well. Obtaining and digitising experimental data is a relatively expensive and time consuming process, and the data produced may not be suited for machine learning purposes. For that reason, many have turned to generating virtual microstructures that can be easily produced and then used to simulate and model the mass transport properties.

The dataset used here was generated by Prifling et al [9]. Each virtual sample in the dataset is a 192^3 binary matrix, where 1 represents solid matter and 0 represents void (or a pore space). The dataset is generated using 9 different periodic and statistically isotropic methods. Isotropic refers to a structure that is uniform regardless of rotation, but since the samples are generated through stochastic methods, they can only be claimed to be the same on average in all directions. Being periodic refers to it being possible to repeat the structure on the boundaries while preserving the geometry.

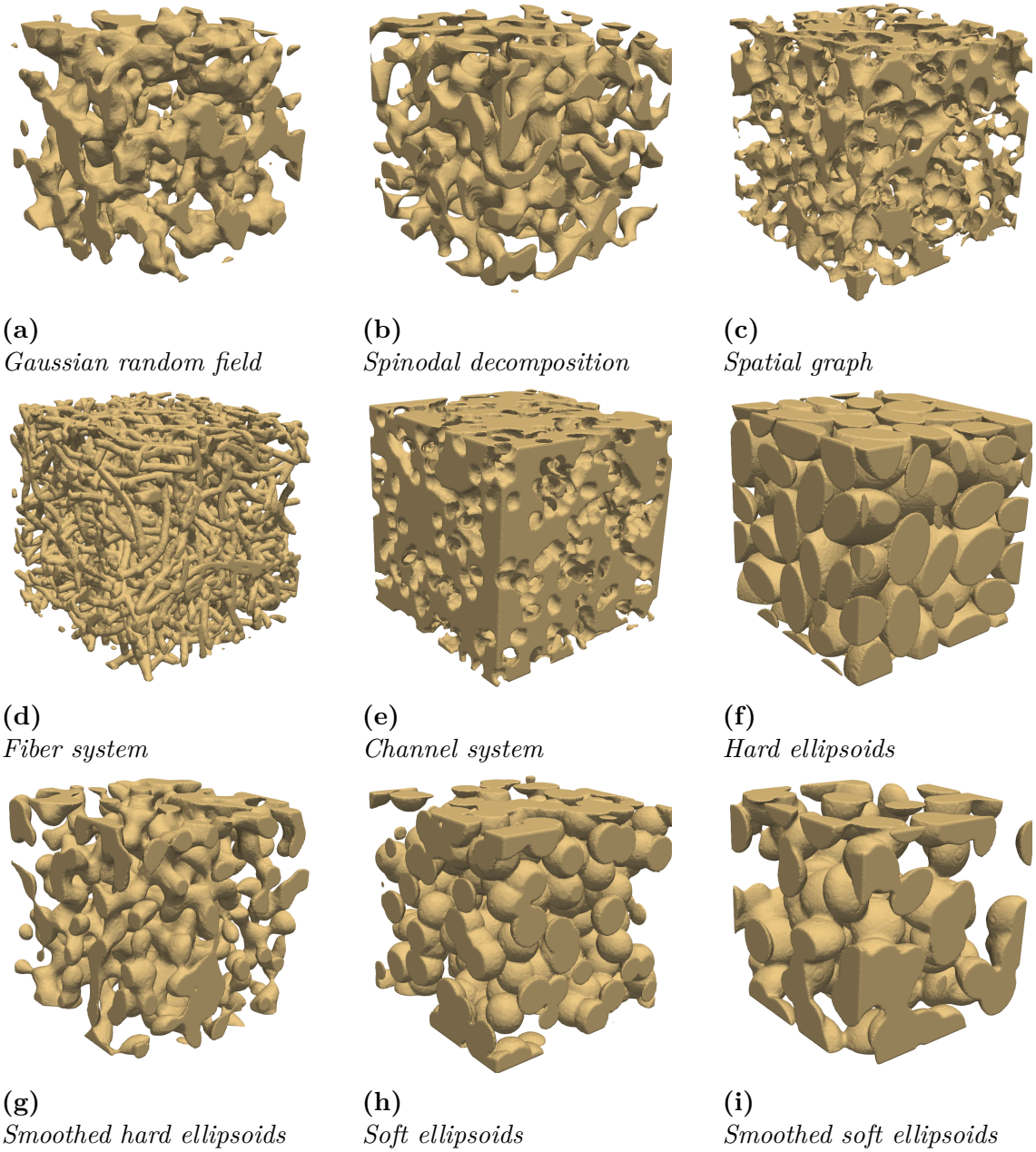
These different generation methods produce different types of geometries, yielding different mass transport properties. There are numerous geometric characteristics that influence the mass transport. Porosity, i.e. the proportion of pore space to the total space in the volume, the surface area, i.e. the interface between solid and pore parts, and tortuosity, i.e. how curved and restrictive the mass transport paths are, all have an effect on the flow.

In this chapter, the different methods used for generating the microstructures are explained to give some understanding of the microstructures in the dataset. Then it is explained how the dataset is split up for training purposes. Finally, it is explained how 2D data is extracted from the generated 3D microstructures.

3.1 Generation Methods

Here the different generation methods used by Prifling et al [9] are described briefly. As mentioned earlier, the different methods produce microstructures with different geometries: some methods produce binary structures, others produce continuous valued structures that need to be transformed to binary values. Continuous structures are transformed to binary matrices by defining a threshold value and setting the matrix entries as a solid or a pore. The threshold value allows for adjusting the porosity values as desired. Porosities of structures produced through binary methods can also be adjusted through different configurations. This is important for producing a dataset with sufficiently large variety of different structures.

These methods are: Gaussian random fields, spinodal decomposition, spatial stochastic graphs, fiber systems, channel systems and four different ellipsoid structures.



(a) *Gaussian random field*

(b) *Spinodal decomposition*

(c) *Spatial graph*

(d) *Fiber system*

(e) *Channel system*

(f) *Hard ellipsoids*

(g) *Smoothed hard ellipsoids*

(h) *Soft ellipsoids*

(i) *Smoothed soft ellipsoids*

Figure 3.1: *9 types of microstructures*

Gaussian random fields are random processes characterised by their mean and covariance function. They are well known within stochastic geometry and have been used in modelling 3D microstructures, for example, anodes in lithium-ion batteries and electrodes in solid oxide cells [16]. To generate a Gaussian random field with a desired covariance function, its spectral density can be multiplied with the Fast Fourier transform (FFT) of white noise and then inverse transformed. This produces a continuous field, on which a threshold needs to be applied to obtain a binary struc-

ture. The threshold can be varied to yield the desired porosity [8, 17].

Spinodal decomposition is when a uniform phase separates into two distinct phases. The phase separation mechanism is simulated by the Navier-Stokes and Cahn-Hillard equations and the computation was performed using Lattice Boltzmann method. The separation mechanism is determined by several factors such as surface tension, density and viscosity ratio between the two phases. These characteristics are varied to yield different behaviors and in turn different geometries. As the structure coarsens with time, the number of timesteps is specifically selected to ensure the structure has appropriate length scale. Then it is thresholded to ensure the desired porosity.

Spatial stochastic graphs are based on a model introduced by Gaiselmann et al [18]. Vertices are introduced randomly to a volume through a stochastic process. There are various parameters that control this process, such as vertices per volume density. Two vertices are connected by an edge if there does not exist a vertex in the critical region between them. The critical region is defined as the intersection between two spheres centered on the vertices and have radius equal to the distance between the vertices.

The fiber system used in the dataset is generated by the method described by Townsend et al [19] in their paper. Modifications were made by Prifling et al [9] to generate periodic and isotropic structures. Fibers are represented by a set of nodes and a Bézier curve that goes through them. The nodes are created by a random walk with limited angle of direction. The thickness of the fibers is randomly selected from a discrete distribution. Fibers are then added to an empty system until a desired porosity is reached.

The channel system is the inverse of the fiber system. Channels are generated in the same manner as the fibers, Bézier curves are generated through random walk with limited angle of direction. The difference is how they are used in the construction. The microstructure generated by the channel system starts out solid and then channels are 'subtracted' from it. This continues until a desired porosity is reached.

For *hard ellipsoids*, ellipsoid particles are randomly placed with random orientations. Then the configurations are relaxed by applying random translations and rotations until no ellipsoids overlap. Attempted translations and rotations are only performed in the cases where they lead to a less or equal amount of overlap of the particles in question. The specifics of how the desired porosity is achieved is out of the scope for this thesis and reader is pointed to the paper by Röding et al [8] where this structure is discussed in more detail.

Smoothed hard ellipsoid structures are generated in the same manner as hard ellipsoid structures, but with an additional step. The final discretized structure is smoothed by applying a Gaussian filter and then thresholded which results in a semi-continuous solid phase in contrast to the original discrete structure.

For *soft ellipsoids*, ellipsoid particles are randomly placed with random orientations. Particles can overlap and new particles are added until the desired porosity is achieved. Ellipsoid sizes are selected at random to obtain an appropriate range of length scales. The shape is also varied between oblate and prolate ellipsoids.

Smoothed soft ellipsoids are generated as smoothed soft ellipsoids, a Gaussian filter is then applied to the structure and then thresholded to smooth it out and obtain a semi-continuous solid phase.

3.2 Training Dataset

10000 samples of each of the 9 structure types were generated, which results in a dataset containing 90000 virtual microstructure samples. The distribution of porosities within each structure type was kept uniformly distributed to ensure a variety within each microstructure type.

The Lattice Boltzmann method was used to simulate mass transport through the sample structures and then permeability and diffusivity values were extracted from the simulations. This was done for each of the samples with the direction of flow being parallel to the first dimension. The resulting mass transport property values were then transformed to simplify training. Diffusivity M, which has values between 0 and 1 was logit-transformed,

$$y = \log\left(\frac{M}{1-M}\right), \tag{3.1}$$

while permeability k, which has positive values, was log-transformed,

$$y = \log(k). \tag{3.2}$$

These values were then used as labels, which enables the use of supervised learning to train a CNN to predict the mass transport properties of the virtual samples.

For the convolutional neural network the dataset is split into 3 sets: training, validation and test. It was split in a stratified manner 70% (63000), 15% (13500), 15% (13500) such that all sets have an equal amount of microstructure types, 7000 per type in training and 1500 per type in validation and testing. This is the same split as used by Prifling et al [9], which facilitates an easy comparison between their methods and the 2D CNN models trained here.

3.3 2D Data Extraction

The aim of this work is to predict mass transport properties in 3D microstructures using 2D data. Therefore a 2D representation needs to be extracted. The 192^3 volumetric cube in Figure 3.2 will be used as an example of the process.

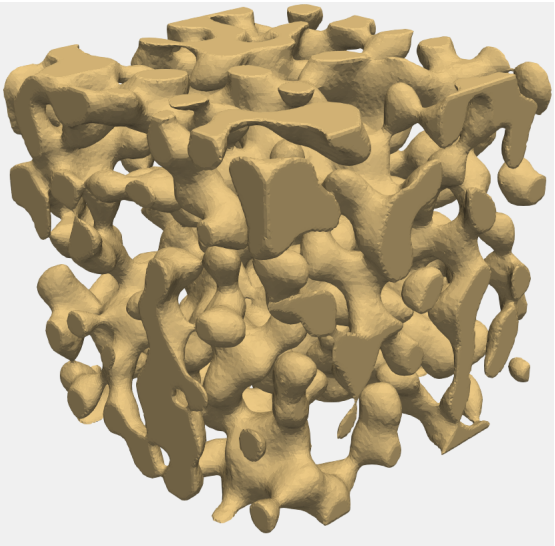


Figure 3.2: *3D sample of a microstructure generated using the smoothed hard ellipsoid method. This sample will be used to demonstrate the extraction methods used.*

One way the data can be represented is by selecting one 192^2 slice from the 192^3 cube. This produces a 192×192 binary cross-section where 1 represents solid and 0 represents pore space, the same as it does for the binary cube sample. An example of how a cross-section is selected and extracted is shown in Figure 3.3 and the resulting 2D binary sample is shown in Figure 3.4.

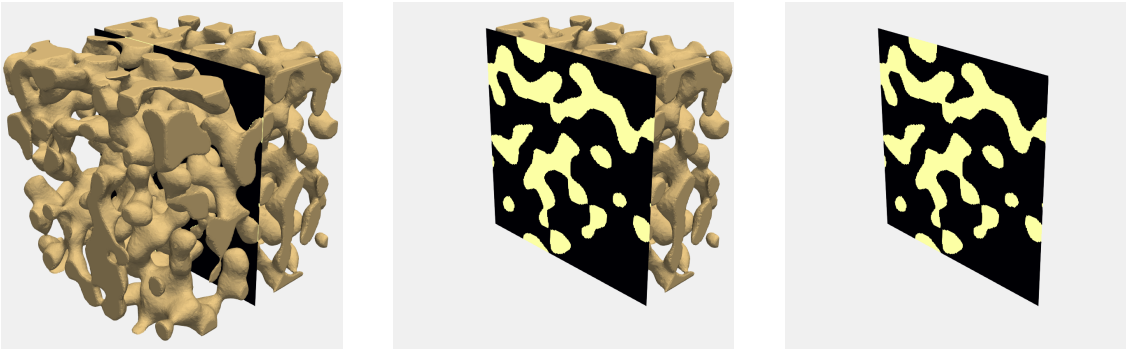


Figure 3.3: *A demonstration of how a 2D binary slice is extracted from a 3D volumetric cube.*

Figure 3.4: *The 2D cross-section obtained from the 3D volumetric cube, showing pores (black) and solids (light yellow).*

Extracting a cross-section can be performed on the fly for each sample during training. However, this can have a high computational cost. When feasible, it is better to perform expensive computations once and store the results. Additionally, storing all the volumetric cubes in the dataset takes up a large amount of space and is difficult to keep in memory during training. The entire volume is not needed for the training, only the extracted 2D data.

From each cube sample, it is possible to extract binary cross-sections from 3 different orientations. These orientations are orthogonal to each other and the resulting cross-sections would be orthogonal as well. An example of how three orthogonal cross-sections are extracted is shown in Figure 3.5 and the resulting 2D binary samples are shown in Figure 3.6. These binary samples are used to construct an *orthogonal set*, which takes less memory than the original cube. There are multiple options for constructing orthogonal sets. Each orientation offers 192 possible cross-sections. A single volumetric sample can therefore provide multiple orthogonal sets. In this work, eight sets of orthogonal cross-sections are extracted from each volumetric sample, with each cross-section being selected at random.

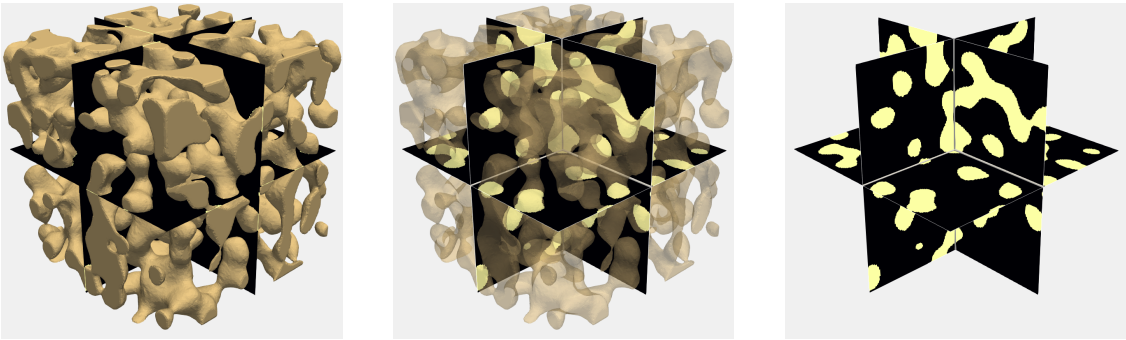


Figure 3.5: *A demonstration how three orthogonal cross-sections are extracted from a 3D volumetric cube.*

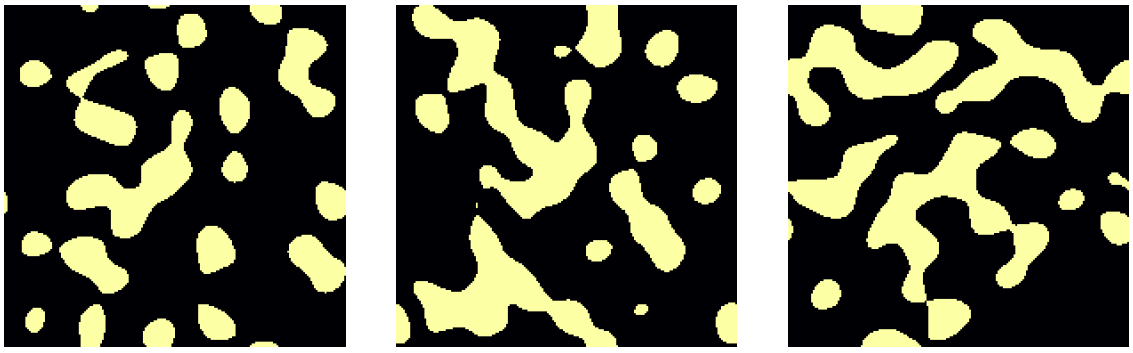


Figure 3.6: *Three binary cross-sections extracted from a volumetric cube. The cross-sections are orthogonal to each other.*

Another way to extract 2D data from a 3D volume is to calculate the mean value of the 192^3 binary matrix in a single direction. This produces a 192^2 density map with values ranging from 0 to 1, with higher values representing higher densities of solid matter. An example of how the density map in one orientation is obtained is shown in Figure 3.7.

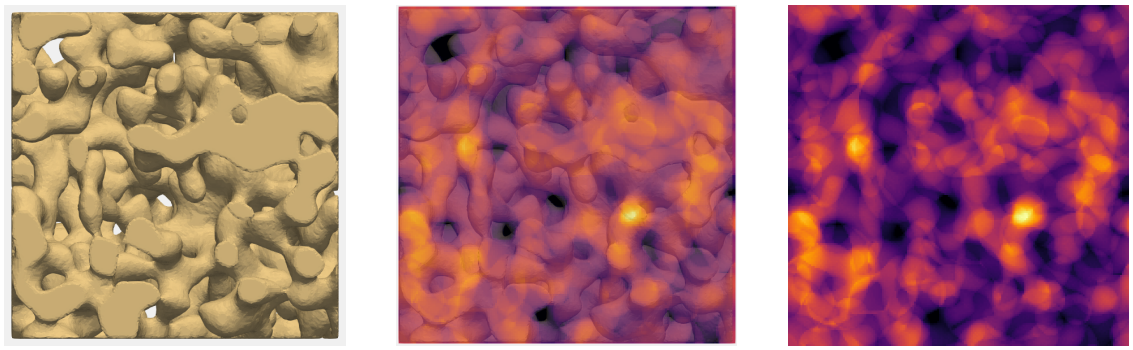


Figure 3.7: *A volumetric cube rendered with orthographic projection (left) with the corresponding obtained density map (right), showing regions of high density (bright) and low density (dark). To show the transition, a transparent version of the orthographic projection has been superimposed over the density map (center).*

Similarly to the binary cross-sections, it is possible to extract three density maps orthogonal to each other from each volumetric sample. An example of orthogonal density maps obtained from the cube in the beginning of the section (Figure 3.2) are shown in Figure 3.8. It is only possible to extract a single set of orthogonal density maps from each cube.

A comparison of the amount of memory used for the different collections of 2D data can be seen in Table 3.1. Note that the listed examples compare storing the samples in a binary file format which stores binary values as integers which takes 1 byte and floating numbers in density maps take 4 bytes.

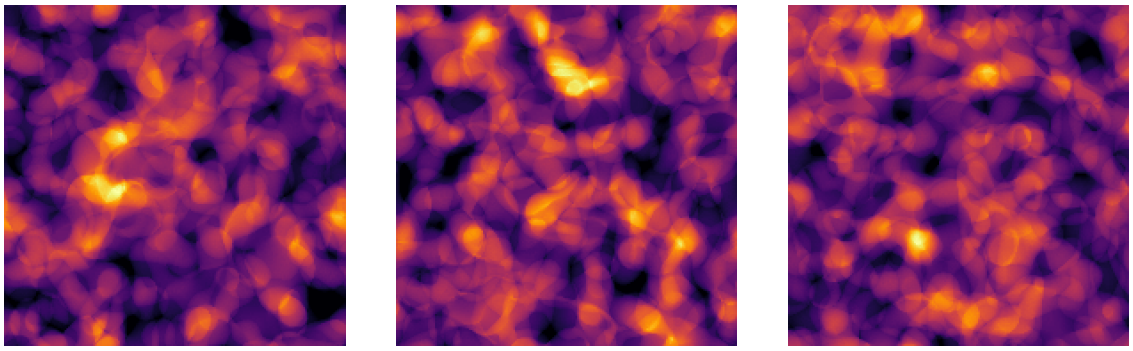


Figure 3.8: *Three extracted density maps, which are orthogonal to each other, showing regions of high density (bright) and low density (dark).*

Sample	Dimensions	Memory/Sample	Memory/Dataset
Volumetric cube	192^3	7.1 MB	637.0 GB
Binary cross-section	192^2	36.9 KB	3.3 GB
Density map	192^2	147.5 KB	13.3 GB
Orthogonal set of binary cross-sections	3×192^2	110.6 KB	9.9 GB
Orthogonal set of density maps	3×192^2	442.4 KB	39.8 GB
Eight sets of binary cross-sections	$8 \times 3 \times 192^2$	884.7 KB	79.6 GB
Down-scaled cube	96^3	884.7 KB	79.6 GB

Table 3.1: *Sample memory sizes in binary file format. Individual 2D data is included for comparison with the final orthogonal sets. The down-scaled cubes used by Prifling et al [9] is included for comparison. Binary values are represented by integers, integers are stored with 1 byte (8 bits) and floating numbers with 4 bytes (32 bits).*

4
Model Architecture

In this chapter, the architecture of the various models and how it was designed will be explained. The differences in the examined models will then be discussed in a separate section. Additionally, data augmentation and how it is used in training will be briefly explained.

4.1 Architecture Design

To determine the viability of the different 2D data types for predicting mass transport properties, multiple arrangements of the data will be examined to be compared and contrasted. A different model will be trained for each data arrangement. To make the comparison more fair, the same CNN model architecture will be used. This does not include the input layer of the CNNs which needs to be adapted to the input data in each case.

The starting point for the architecture design is the 3D CNN model used by Prifling et al [9]. One purpose of this work is to compare the performance of 2D CNNs and 3D CNNs using data that originates from the same dataset. As their 3D CNN model has been established to perform well on the dataset it is reasonable to assume that it would perform decently when adapted for 2D inputs. This adaptation involved replacing the 3D convolutional layers with 2D convolutional layers. Keeping the architecture similar has the additional benefit of making comparisons between the 3D and 2D CNNs easier, provided the changes to the architecture of the latter are not too extreme. However, it should not be taken for granted that it is the best possible 2D CNN design and variations should be explored.

The main hyper-parameters explored were the number of dense layers after the convolution layers, the number of neurons per dense layer and the number of feature maps in the convolution layers. The exploration was performed using random search due to ease of implementation [20]. The hyper-parameters were selected uniformly randomly from a selection of values deemed reasonable and then the networks were trained for a predefined amount of time. After training, the results were compared to the original architecture used as a starting point. However, no clear improvements were discovered so another attempt was made by adding two convolution layers and a pooling layer and repeating the random search. This yielded no significant improvement either so the number of dense layers and neurons per dense layer were kept the same. The number of feature maps was increased to compensate for the loss of 3D context. However, there were no significant improvements. The final CNN model architecture can be seen in Table 4.1, where N is the number of channels in

the input data. A visual representation of the architecture is shown in Figure 4.1. The model was implemented in Tensorflow 2.9.0 [21].

Layer	Activation	Specifics	Output shape
Input			192x192xN
Convolution	ELU	3x3 Kernel	190x190x32
Convolution	ELU	3x3 Kernel	188x188x32
Pooling	AVG		94x94x32
Convolution	ELU	3x3 Kernel	92x92x64
Convolution	ELU	3x3 Kernel	90x90x64
Pooling	AVG		45x45x64
Convolution	ELU	3x3 Kernel	43x43x128
Convolution	ELU	3x3 Kernel	41x41x128
Pooling	AVG		20x20x128
Flatten			51200
Dense	ELU	Dense	64
Dense	ELU	Dense	64
Dense	ELU	Dense	64
Dense	ELU	Dense	64
Output	Linear	Dense	1

Table 4.1: *Model parameters. N is the number of channels or layers in the input data.*

The activation function used in the CNN model is called ELU which stands for Exponential Linear Unit and is defined as,

$$f(x) = \begin{cases} x, & \text{if } x > 0 \\ \alpha(\exp(x) - 1), & \text{if } x \leq 0 \end{cases}, \tag{4.1}$$

with parameter $\alpha = 1$. It was chosen because it has been shown to perform well with CNNs [22]. Average pooling (AVG) was used to consolidate the feature maps.

The change from 3D convolution to 2D convolution freed some computational resources, which could then be used to speed up the training. Increasing the batch from 16 to 512 enables more computations to be performed in parallel, decreasing the time spent on each epoch. However increasing the batch size necessitates increasing the learning rate as well. The learning rates in the original 3D CNN would change during the training. In the beginning it would be small to prevent the model from diverging, then increase to speed up the convergence as the training goes on, before decreasing again to allow for finer adjustments. The new learning rates follow the same behavior and can be seen in Figure 4.2, the values can be seen in Table 4.2.

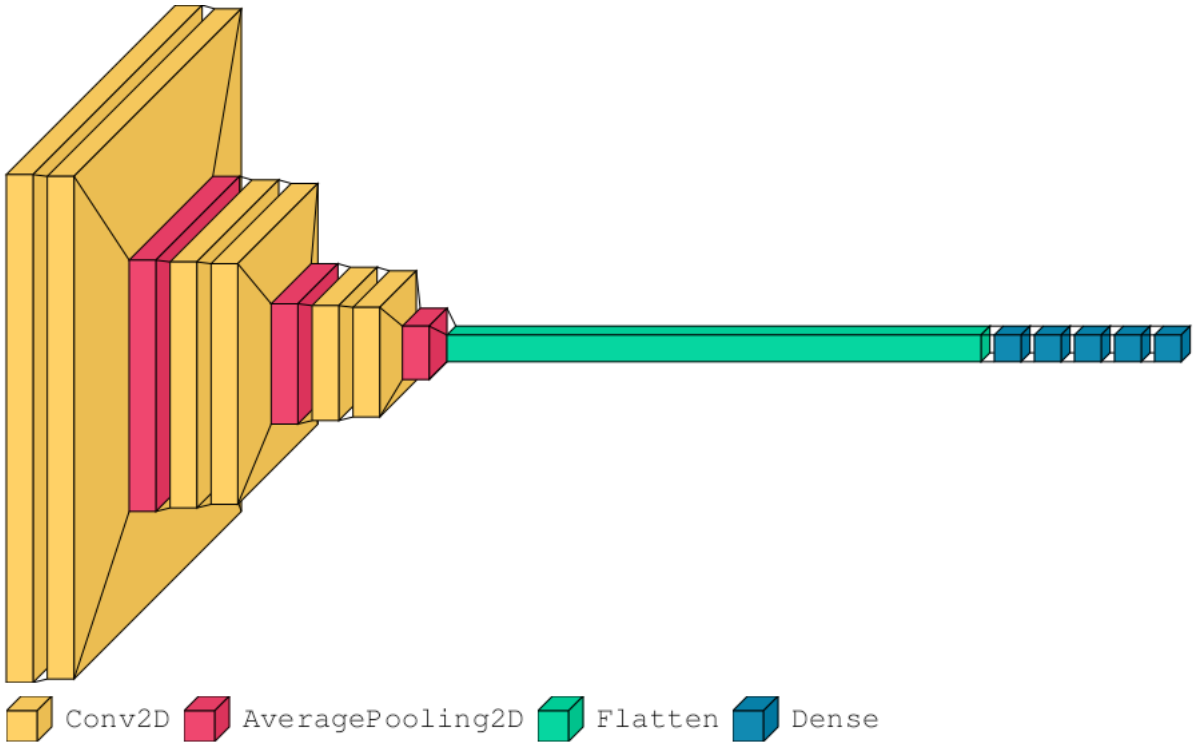


Figure 4.1: *Model architecture diagram.*

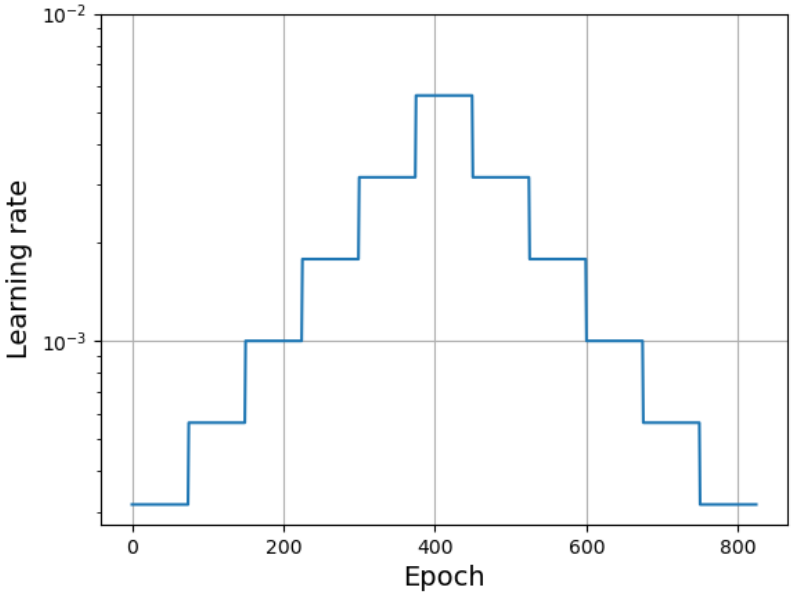


Figure 4.2: *Learning rate changes over epochs.*

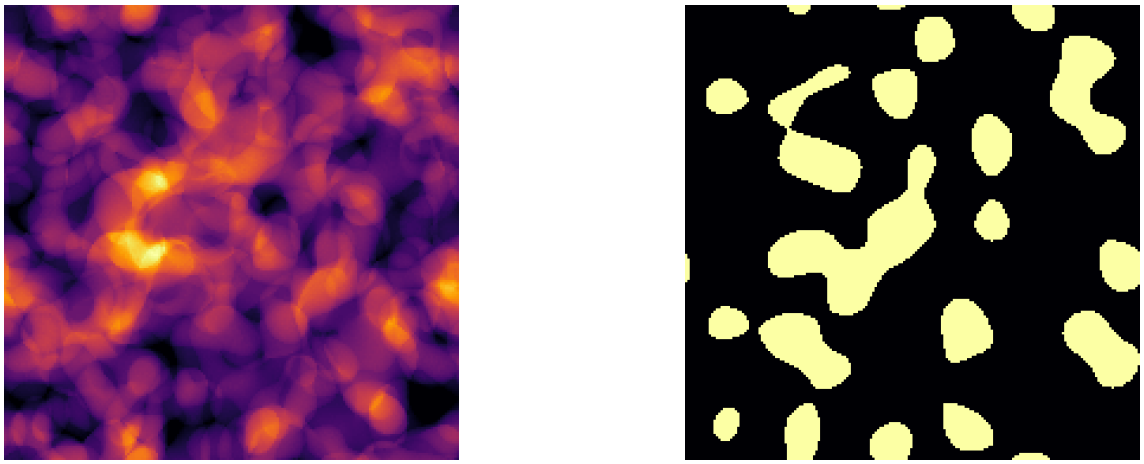
$\log_{10}$(LR)	LR
-3.50	0.000316
-3.25	0.000562
-3.00	0.001000
-2.75	0.001778
-2.50	0.003162
-2.25	0.005623
-2.50	0.003162
-2.75	0.001778
-3.00	0.001000
-3.25	0.000562
-3.50	0.000316

Table 4.2: *Learning rates.*

4.2 Examined Models

The extracted 2D data comes in two types: 2D binary cross-sections and density maps. Both are stored in orthogonal sets. To investigate how viable and useful the 2D data is in comparison to the original 3D data, multiple models with different

input criteria are created. For 2D CNNs it is possible to layer multiple 2 dimensional matrices together to use as an input. Using this method, 6 different arrangements of 2D input data will be created, resulting in the training of 6 CNN models..



(a) *Density map* **(b)** *Cross-section*

Figure 4.3: *Examples of the two types of input data used in training 2D CNN models.*

The *first model* will train on 2D cross-sections perpendicular to the direction of transport. This gives a view of the transport cross section, however 3D contextual information will be missing, such as whether the transport paths are tortuous. Fortunately due to the statistically isotropic properties of the cubic samples, some topological properties may be estimated. An example of input data can be seen in Figure 4.4.

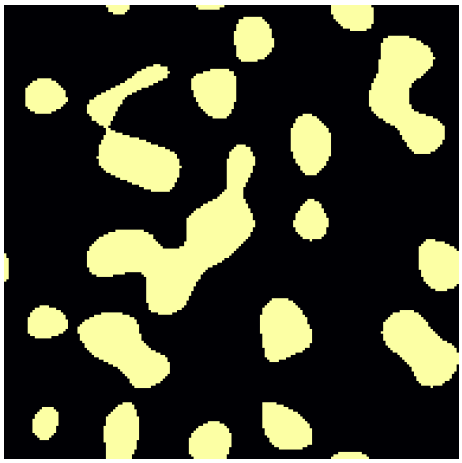


Figure 4.4: *Model 1 is trained on a single 2D binary cross-section. The cross-section is perpendicular to direction of transport.*

The *second model* will train on 2D cross-sections parallel to the direction of transport. This gives a cross-section view of the transport paths, making it easier to view obstruction and tortuosity though the complete 3D data is not available. This model should have similar properties as the first one due to the statistical isotropy of

the samples, but is still worth exploring. Cross-sections are stored in orthogonal sets which makes it possible to select between them during training, increasing variety. An example of input data can be seen in Figure 4.5.

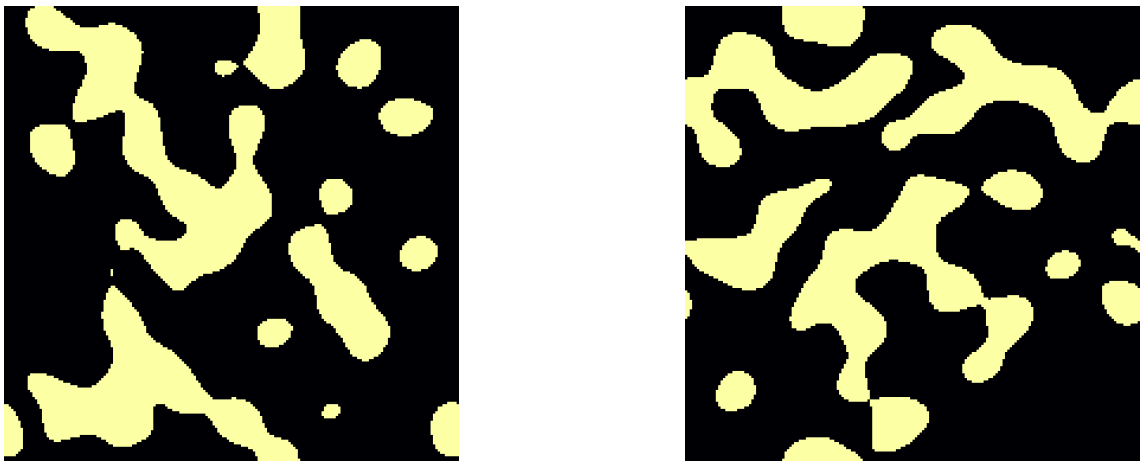


Figure 4.5: *Model 2 is trained on a single 2D binary cross-section. Here, two cross-sections parallel to the direction of transport in the same orthogonal set are depicted. One of them is selected at random during training.*

The *third model* will train on three 2D binary cross-sections which are orthogonal to each other and use all the cross-sections in the orthogonal set. This gives a cross-section of the transport from the three main orientations and preserves more of the spatial context. An example of input data can be seen in Figure 4.6.

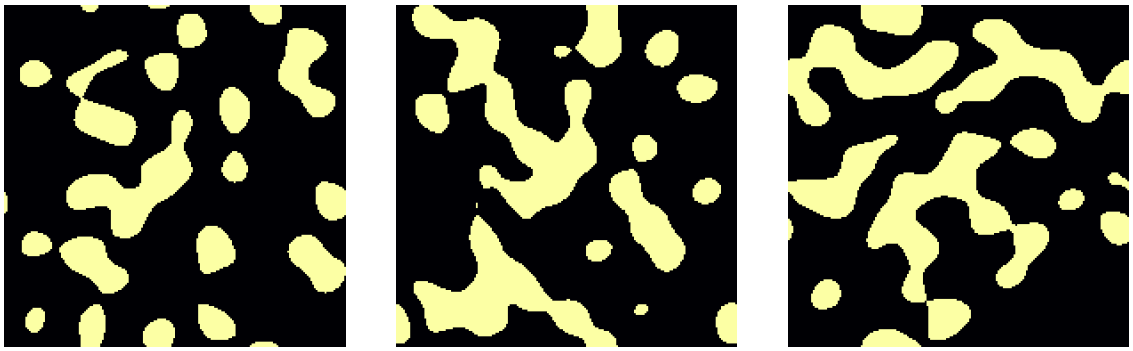


Figure 4.6: *Model 3 is trained on three orthogonal cross-sections extracted from the same microstructure. The input is a* $3 \times 192 \times 192$ *matrix, which is obtained by concatenating the cross-sections together.*

This is not the first time a CNN model has been trained on multiple cross sections to estimate properties of a volumetric structure. Rong et al [23] did so with thermal conductivity of composite materials and obtained good results. However, multiple parallel cross-sections were used instead of orthogonal ones. The performance impact of the number of parallel cross-sections in the input data was one of the factors investigated.

Materials science is not the only field where applying 2D CNNs to process 3D data has been attempted. The medical field works a great deal with volumetric data,

e.g MRI, and there have been similar attempts of utilising 2D CNNs instead of 3D CNNs. Yu et al [24] compared 2D and 3D CNNs in reducing the number of false positives in lung cancer screenings. The 2D CNNs were trained on a pseudo-RBG image dataset where the color channels represented an orthogonal cross-sections from 3D voxel samples. For an example of how pseudo-RGB image could be constructed from cross-sections of a microstructure, see Figure 4.7. Orthogonal cross-sectional data has been given other names, Prasoon et al [25] called this method triplanar and Roth et al [26] called it 2.5D.

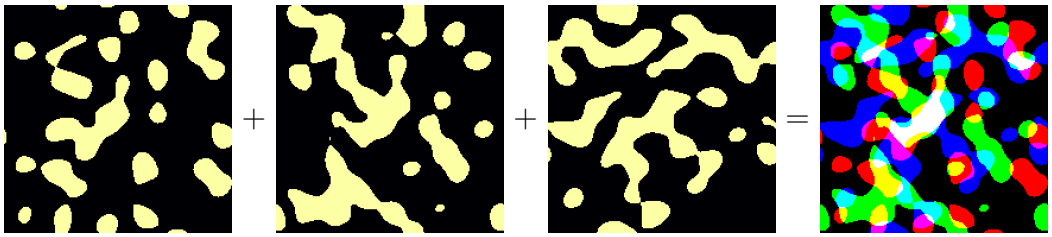


Figure 4.7: *An example of how a pseudo-RGB image could be constructed from an orthogonal cross-section set.*

The *fourth model* will train on a density map perpendicular to the direction of transport. This gives more complete information about the samples such as porosity and some idea about the distribution of solid and pores. An example of input data can be seen in Figure 4.8.

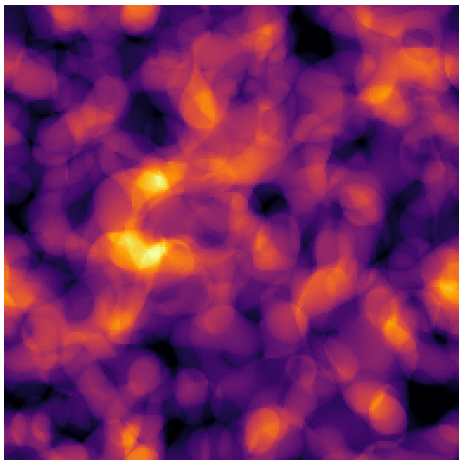


Figure 4.8: *Model 4 is trained on a single 2D density map. The map is perpendicular to direction of transport.*

The *fifth model* will train on three density maps orthogonal to each other, which will use all the cross-sections in the orthogonal sample set. The porosity can be obtained from the density maps. The distribution of solid and pores can be more accurately estimated due to the three orientations. An example of input data can be seen in Figure 4.9.

The *sixth model* will be a combination of the third and fourth models. This is to examine and contrast the different types of extracted 2D data. An example of input data can be seen in Figure 4.10.

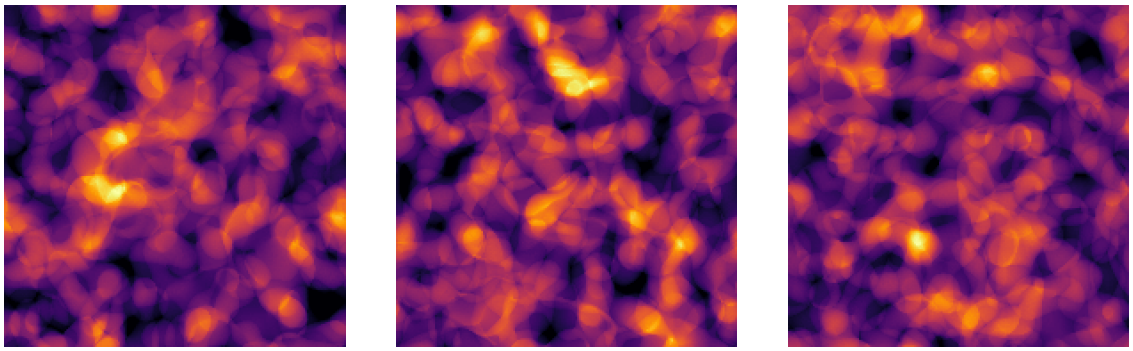


Figure 4.9: *Model 5 is trained on three orthogonal density maps extracted from the same microstructure. The input is a* $3 \times 192 \times 192$ *matrix, which is obtained by concatenating the maps together.*

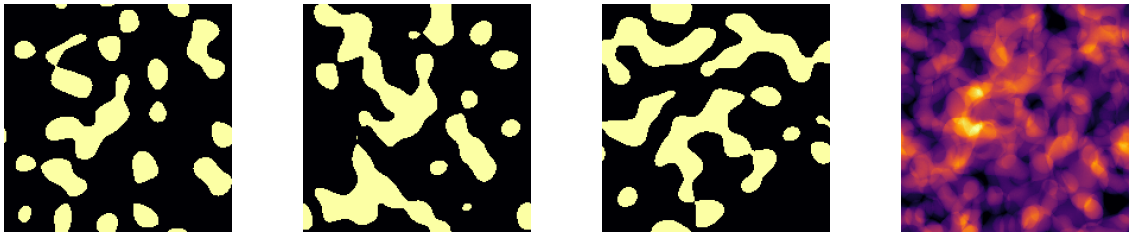


Figure 4.10: *Model 6 is a combination of model 3 and 4, and trains on three orthogonal cross-sections and a density map. The input is a* $4 \times 192 \times 192$ *matrix, which is obtained by concatenating the 2D data together.*

Method	Type	Orientation
Model 1	Binary	Perpendicular
Model 2	Binary	Parallel
Model 3	Binary	3 Orthogonal
Model 4	Density map	Perpendicular
Model 5	Density map	3 Orthogonal
Model 6	Mixed	Mixed

Table 4.3: *Examined models and corresponding data arrangements*

4.3 Data Augmentation

As discussed earlier, over-fitting is a problem in machine learning. One way to prevent overfitting is to have a large and varied training dataset. The size of the training set can be artificially increased by creating new samples from the pre-existing ones. This often involves modifying existing samples in a way that does not require recalculation of the label. Such augmentations come with a computational cost, but if the operations are relatively cheap then it offers great benefits if storage space is limited.

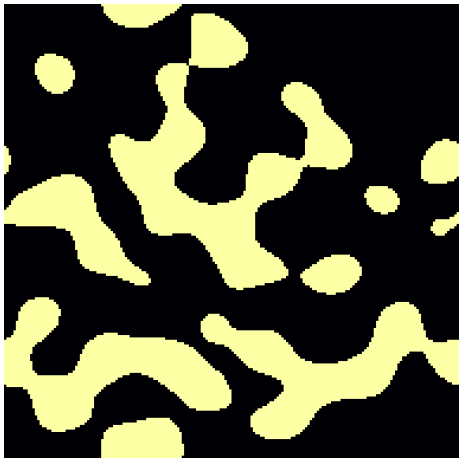
When augmenting the microstructures, care is needed not to affect the morphological features. The microstructures are statistically homogeneous, isotropic and

have periodic boundary conditions. The microstructures can be rotated, mirrored and transposed without affecting their properties, as long as the direction in which the mass transport is simulated, the x axis, is not altered. Therefore augmentations that affect the direction of transport will not be chosen. Alterations on the y and z axes are fine as the structures have periodic boundaries, they can be swapped, flipped and shifted.

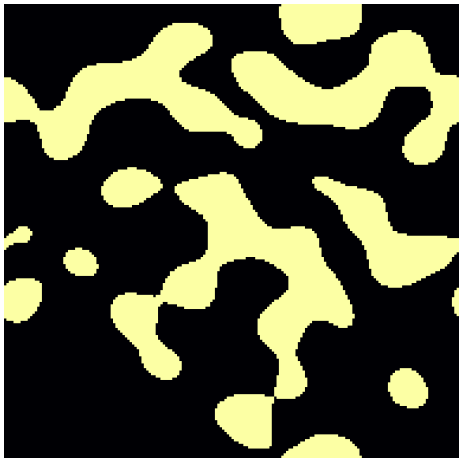
The augmentations used in this work can be seen in Figure 4.11. They are applied at random to each sample during training.

(a) *Original*

(b) *Swap Y and Z axes.*



(c) *Mirror along Y-axis. Same as mirroring across Z-axis.*



(d) *Mirror along Z-axis. Same as mirroring across Y-axis.*

(e) *Circular shift on Y-axis.*

(f) *Circular shift on Z-axis.*

Figure 4.11: *Augmentations used during training. The example binary slice has been selected for its recognizable features.*

5

Discussion and Results

In this chapter, organisation of the training and related topics will be covered. Then the results will be presented and discussed.

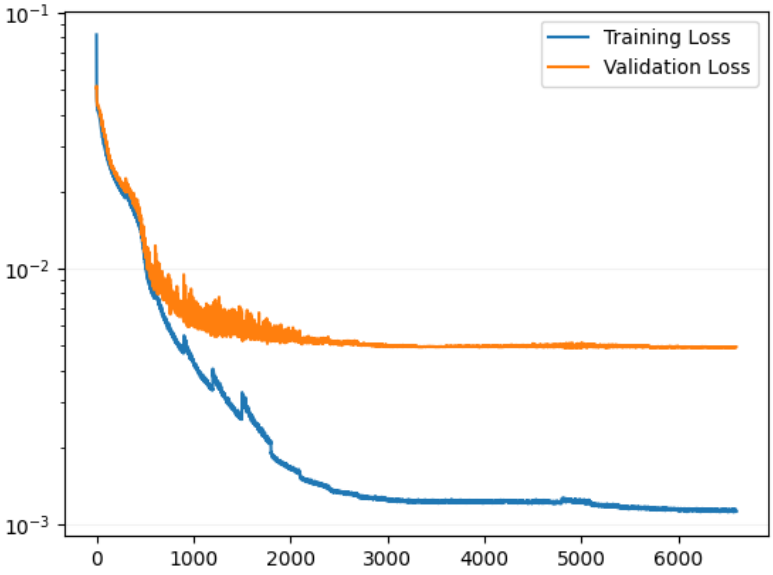
5.1 Training Sessions

The training was performed on the Alvis GPU computer cluster at Chalmers Centre for Computational Science and Engineering (C3SE). The training of each model utilised a single NVIDIA Tesla A40 GPU with 48GB RAM, with Intel(R) Xeon(R) Gold 6338 CPU @ 2GHz, with 16 CPU cores allocated for the training.

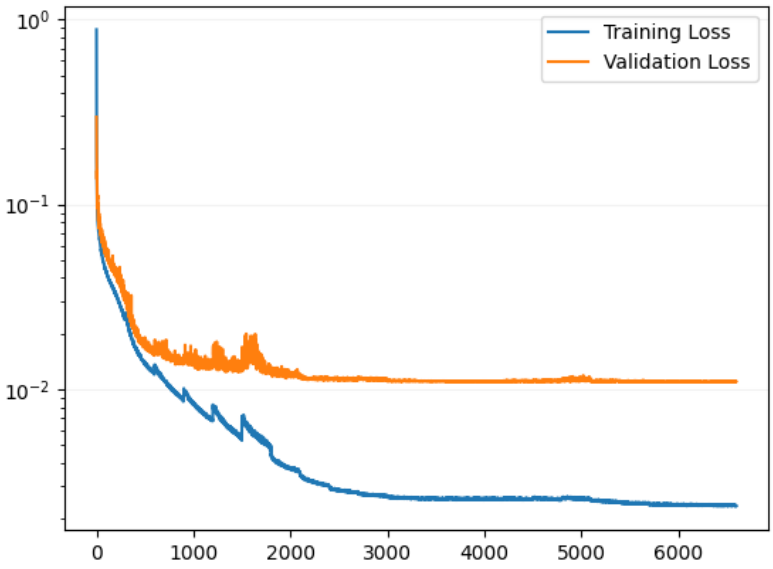
The models were trained over three training sessions, 5 days each, due to the wall time limit on the cluster. In each session, the models were trained for 825 epochs for binary inputs and 3300 epochs for density maps and mixed input. The training on the density maps and mixed input took less time due to the smaller training sets, which resulted in each epoch taking less time. The number of training epochs was increased to 3300 to take advantage of all the allocated training time. This was done both for permeability models and diffusivity models.

For the first session each model was initialised four times, each with random weights and biases. In second session, the weights from the best performing models of the first session were used to set the weights of the new models. This allowed for continuation of training despite the wall time limit. Additionally, each model was trained with new random weights during the second session. The intention was to see if the models from the previous training session had reached a point of diminishing returns and training more instances of the model could produce improvements. However, in only a few instances did the models initialised with random weights outperform the best models from previous session. Even in those cases, the improvement had been minor. Therefore, the third and final session was only initialised with the best performing models. However, this training session only resulted in minor improvements for some of the models. The other models did not improve.

Examples of the training progress as a function of loss over epochs from two training sessions can be seen in Figure 5.1. As mentioned earlier, little to no improvements occurred during the last training session. For that reason it is not included in the loss progression figure.



(a) *Diffusivity.*



(b) *Permeability.*

Figure 5.1: *An example of training and validation loss over 2 training sessions from model 5. The visible spikes are from the changing learning rate.*

5.2 Results

The results from the trained models can be seen in Tables 5.1 and 5.2, for diffusivity and permeability respectively. The error of the models is quantified in terms of

Mean Squared Error,

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(X_i - \widehat{X}_i\right)^2, \tag{5.1}$$

the same error used for training loss, and in terms of Mean Absolute Percentage Error

$$\text{MAPE} = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{X_i - \widehat{X}_i}{X_i}\right|. \tag{5.2}$$

The results from the 3D CNNs trained by Prifling et al [9] are included for comparison.

MSE shows the loss value of the models for each of the training subsets: training, validation and testing. Comparing the MSE errors over the subsets shows how well the models generalize. However it is difficult to understand intuitively what the values represent in regards to the accuracy of the models. MAPE is easier to put into context, for example a MAPE of 10 means that on average the predicted value is 10% off from the true value.

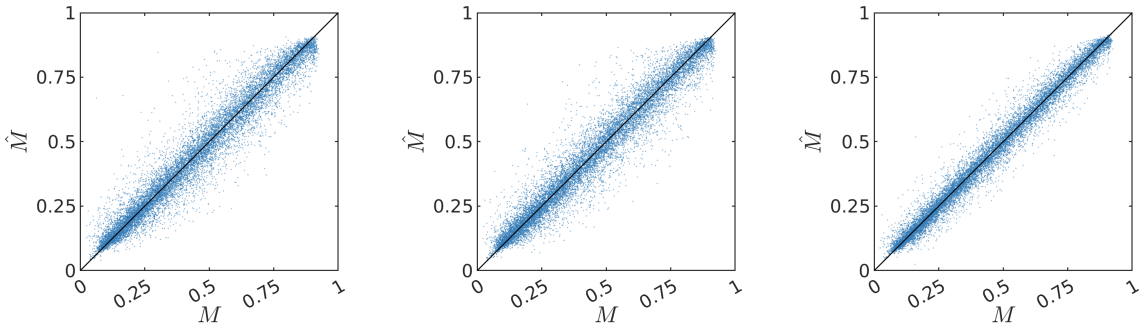
Method	MSE			MAPE (%)
	Train	Validate	Test	Test
Model 1	0.0815	0.0842	0.0842	11.66
Model 2	0.0810	0.0876	0.0876	11.91
Model 3	0.0363	0.0389	0.0384	7.88
Model 4	0.0047	0.0057	0.0061	2.97
Model 5	0.0012	0.0049	0.0047	2.51
Model 6	0.0038	0.0068	0.0063	3.09
3D CNN	0.0016	0.0023	0.0020	1.65

Table 5.1: *Diffusivity results. The corresponding 3D CNN model by Prifling et al [9] is included for comparison.*

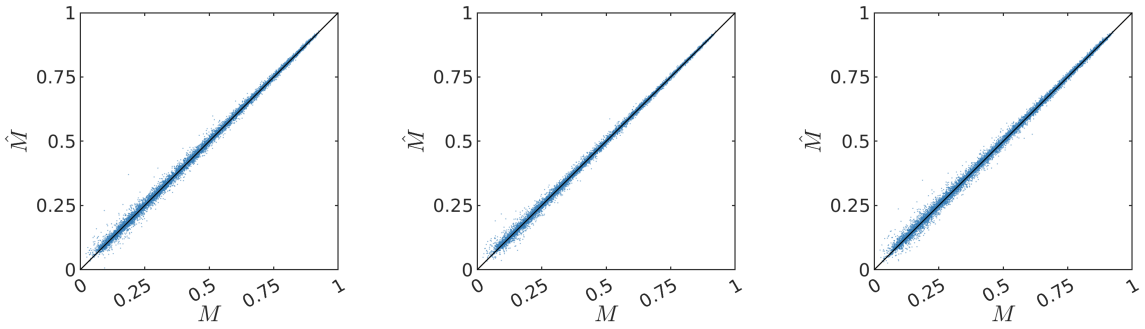
Method	MSE			MAPE (%)
	Train	Validate	Test	Test
Model 1	0.1491	0.1618	0.1581	31.91
Model 2	0.1463	0.1678	0.1638	31.34
Model 3	0.0651	0.0753	0.0724	19.79
Model 4	0.0119	0.0187	0.0177	8.58
Model 5	0.0024	0.0109	0.0103	6.24
Model 6	0.0100	0.0170	0.0159	8.39
3D CNN	0.0033	0.0059	0.0050	4.33

Table 5.2: *Permeability results. The corresponding 3D CNN model by Prifling et al [9] is included for comparison.*

Additionally, scatter plots of the predicted and true values from the test set have been generated, see Figure 5.2 for diffusivity and Figure 5.2 for permeability.

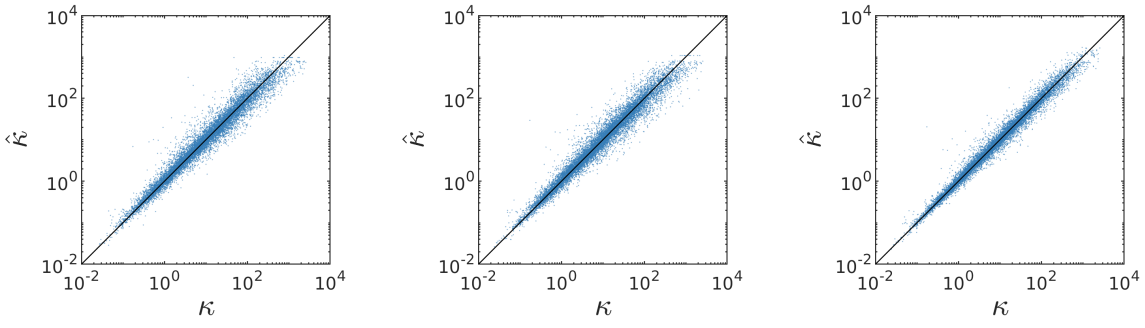


(a) *Model 1* **(b)** *Model 2* **(c)** *Model 3*

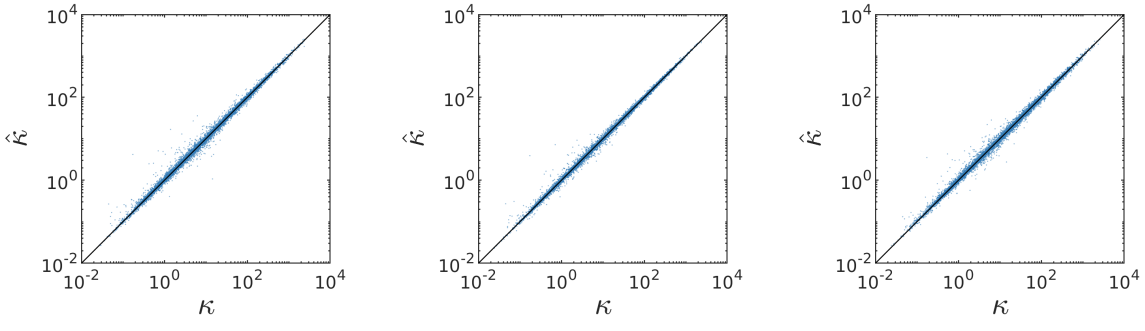


(d) *Model 4* **(e)** *Model 5* **(f)** *Model 6*

Figure 5.2: *Scatter plots of diffusivity with true values M and predicted values $\hat{M}$.*



(a) *Model 1* **(b)** *Model 2* **(c)** *Model 3*



(d) *Model 4* **(e)** *Model 5* **(f)** *Model 6*

Figure 5.3: *Scatter plots of permeability with true values κ and predicted values $\hat{\kappa}$.*

It is clear that the models using density maps (4-6) outperform the ones using binary cross-sections (1-3). This is expected because a single density map contains all the information needed to calculate porosity, which is an important characteristic for the mass transport properties. An estimation of the solid and pore distribution can also be observed from the density maps which is an additional piece of information that can be used in training.

Some of the information contained in a density map can be estimated from a cross-section, but information is lost when it is separated from the 3D context. As can be seen from model 3, introducing more cross-sections includes more of the 3D context and the error is decreased, in comparison to model 1 which uses only a single cross-section as input.

Models 1 and 2 were trained to test if there was any difference between cross-sections perpendicular or parallel to the direction of transport. As the 3D samples are statistically isotropic, the orientation should not matter to the CNN models. This is confirmed by the results, as there is no substantial difference between the errors.

Model 5 performs better than model 4, as it has the orthogonal density maps which provide additional information to the CNN. This is similar to how model 3 with three orthogonal cross sections performs better than models 1 and 2, which only have a single cross section. However the improvements are not as great as they were for the binary cross-section models. This indicates that the relative information increase is smaller. As mentioned earlier a density map provides information about the porosity and the distribution of pores and solids. Using this as an example, additional density maps do not provide any new information about porosity. However, since they are from different orientations the spatial distribution is also from different orientations which provides more 3D context.

Model 6 provides a great comparison of density maps and binary cross-sections. Its performance can be compared to its constituent models, 3 and 4, which utilise different 2D data. It is an improvement over model 3 which used only orthogonal cross-sections but not over model 4. In fact the performances are comparable, with no clear difference in error values that can not be attributed to the stochastic nature of the CNNs. It can be reasoned that the addition of the orthogonal cross-sections provides little to no additional information that the CNN could take advantage of during training.

Comparing the 2D CNN models explored here and the 3D CNN models by Prifling et al [9], it is clear that the 3D CNN models have the best performance. However what the 2D CNN models offer is an approach to handle limited data when a full volumetric sample is not available. Additionally the 2D CNN models take a fraction of a second to predict a mass transport property, which is substantially faster than it takes to simulate and extract the properties [8].

6
Conclusion

In this thesis, prediction of mass transport properties using 2D CNNs was investigated. The 2D data was extracted from virtually generated 3D microstructures that had been previously used to train various prediction models, including 3D CNNs. Two different methods were used to extract 2D data, density maps and binary slices. The data was then arranged in six different arrangements for six different CNN models.

The produced models provide insights to how useful different 2D data and different arrangements can be, provided that assumptions can be made about the isotropy of the data. Though models using a singular cross-section performed worse than the other methods, they can still be used to produce a fast approximation and narrow down options before committing to a more accurate but more complex and costly process. Orientation of a cross-section to direction of mass transport had little to no effect. This was expected due to the isotropy of the samples.

Density maps performed the best but they require more information about the microstructure than cross-sections. Acquiring this data might not be any cheaper than acquiring full experimental 3D data. However, the models can still predict mass transport properties with reasonable accuracy, faster than full mass transport simulations of the 3D structures. Also 2D CNNs are easier to work with than 3D CNNs.

Some possible future work on the topics of this work would be to explore multiple parallel cross-sections and how the number of them would affect the prediction of mass transport properties, similar to what has been done for thermal conductivity. As mentioned earlier, the orientation of the cross-sections was explored in this work, culminating with the combination of all orientations with the set of orthogonal cross-sections. That showed that having multiple cross-sections increased the accuracy and decreased the error. Additionally, it would be interesting to examine whether different spacing between cross-sections would have some additional effects. Alternatively, limited density maps could be explored, as in taking a density map on a fraction of the data or as a mean of multiple cross-sections. There may be microstructures that are too homogeneous and the resulting density maps would be too uniform and missing information about spatial distribution. The performance of prediction models using density maps in those cases would need to be examined

Bibliography

[1] Barman, Sandra et al. "Prediction of diffusive transport through polymer films from characteristics of the pore geometry". In: *AIChE Journal* 65.1 (2019), pp. 446–457. DOI: 10.1002/aic.16391.

[2] Slater, Anna G. and Cooper, Andrew I. "Function-led design of new porous materials". In: *Science* 348.6238 (2015). DOI: 10.1126/science.aaa8075.

[3] Vasseur, Jérémie et al. "Permeability of polydisperse magma foam". In: *Geology* 48 (Mar. 2020). DOI: 10.1130/G47094.1.

[4] Kondo, Ruho et al. "Microstructure recognition using convolutional neural networks for prediction of ionic conductivity in ceramics". In: *Acta Materialia* 141 (2017), pp. 29–38. DOI: 10.1016/j.actamat.2017.09.004.

[5] Wu, Jinlong et al. "Seeing permeability from images: fast prediction with convolutional neural networks". In: *Science Bulletin* 63.18 (2018), pp. 1215–1222. ISSN: 2095-9273. DOI: 10.1016/j.scib.2018.08.006.

[6] Wu, Haiyi et al. "Predicting effective diffusivity of porous media from images by deep learning". In: *Scientific reports* 9.1 (2019), pp. 1–12. DOI: 10.1038/s41598-019-56309-x.

[7] Graczyk, Krzysztof M and Matyka, Maciej. "Predicting porosity, permeability, and tortuosity of porous media from images by deep learning". In: *Scientific reports* 10 (2020), pp. 1–11. DOI: 10.1038/s41598-020-78415-x.

[8] Röding, Magnus et al. "Predicting permeability via statistical learning on higher-order microstructural information". In: *Scientific reports* 10 (2020), pp. 1–17. DOI: 10.1038/s41598-020-72085-5.

[9] Prifling, Benedikt et al. "Large-Scale Statistical Learning for Mass Transport Prediction in Porous Materials Using 90,000 Artificially Generated Microstructures". In: *Frontiers in Materials* 8 (2021). ISSN: 2296-8016. DOI: 10.3389/fmats.2021.786502.

[10] Kamrava, Serveh et al. "Linking morphology of porous media to their macroscopic permeability by deep learning". In: *Transport in Porous Media* 131 (2020), pp. 427–448. DOI: 10.1007/s11242-019-01352-5.

[11] Srisutthiyakorn, Nattavadee. "Deep-learning methods for predicting permeability from 2D/3D binary-segmented images". In: *SEG technical program expanded abstracts 2016*. Society of Exploration Geophysicists, 2016, pp. 3042–3046. DOI: 10.1190/segam2016-13972613.1.

[12] Araya-Polo, Mauricio et al. "Deep learning–driven permeability estimation from 2D images". In: *Computational Geosciences* 24.2 (2020), pp. 571–580. DOI: 10.1007/s10596-019-09886-9.

[13] Torquato, Salvatore and Haslach Jr, HW. “Random heterogeneous materials: microstructure and macroscopic properties”. In: *Appl. Mech. Rev.* 55.4 (2002), B62–B63.

[14] Krüger, Timm et al. *The Lattice Boltzmann Method. Principles and Practice*. 1st ed. Springer International Publishing, 2017.

[15] O’Shea, Keiron and Nash, Ryan. “An Introduction to Convolutional Neural Networks”. In: *CoRR* abs/1511.08458 (2015). arXiv: 1511.08458. URL: http://arxiv.org/abs/1511.08458.

[16] Kremer, Lea Sophie et al. “Influence of the electrolyte salt concentration on the rate capability of ultra-thick NCM 622 electrodes”. In: *Batteries & Supercaps* 3.11 (2020), pp. 1172–1182. DOI: 10.1002/batt.202000098.

[17] Lang, Annika and Potthoff, Jürgen. In: 17.3 (2011), pp. 195–214. DOI: 10.1515/mcma.2011.009.

[18] Gaiselmann, Gerd et al. “Quantitative relationships between microstructure and effective transport properties based on virtual materials testing”. In: *AIChE Journal* 60 (2014), pp. 1983–1999. DOI: 10.1002/aic.14416.

[19] Townsend, Philip et al. “Stochastic modelling of 3D fiber structures imaged with X-ray microtomography”. In: *Computational Materials Science* 194 (2021), p. 110433. ISSN: 0927-0256. DOI: 10.1016/j.commatsci.2021.110433.

[20] Bergstra, James and Bengio, Yoshua. “Random search for hyper-parameter optimization.” In: *Journal of machine learning research* 13.2 (2012).

[21] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[22] Clevert, Djork-Arné et al. “Fast and accurate deep network learning by exponential linear units (elus)”. In: *arXiv preprint arXiv:1511.07289* (2015).

[23] Rong, Qingyuan et al. “Predicting the effective thermal conductivity of composites from cross sections images using deep learning methods”. In: *Composites Science and Technology* 184 (2019), p. 107861. ISSN: 0266-3538. DOI: https://doi.org/10.1016/j.compscitech.2019.107861.

[24] Yu, Juezhao et al. “2D CNN versus 3D CNN for false-positive reduction in lung cancer screening”. In: *Journal of Medical Imaging* 7.5 (2020), p. 051202. DOI: 10.1117/1.JMI.7.5.051202.

[25] Prasoon, Adhish et al. “Deep Feature Learning for Knee Cartilage Segmentation Using a Triplanar Convolutional Neural Network”. In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2013*. Ed. by Kensaku Mori et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 246–253. ISBN: 978-3-642-40763-5.

[26] Roth, Holger R. et al. “A New 2.5D Representation for Lymph Node Detection using Random Sets of Deep Convolutional Neural Network Observations”. In: *CoRR* abs/1406.2639 (2014). arXiv: 1406.2639. URL: http://arxiv.org/abs/1406.2639.

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY