Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Geoscience and Petroleum

Tormod Kårstad

Predicting Inter-Pore Hydraulic Conductivity Using Convolutional Neural Network

Master's thesis in Reservoir Technology & Petrophysics
Supervisor: Carl Fredrik Berg
Co-supervisor:

June 2022

NTNU
Norwegian University of
Science and Technology

Tormod Kårstad

Predicting Inter-Pore Hydraulic Conductivity Using Convolutional Neural Network

Master's thesis in Reservoir Technology & Petrophysics
Supervisor: Carl Fredrik Berg
Co-supervisor:
June 2022

Norwegian University of Science and Technology
Faculty of Engineering
Department of Geoscience and Petroleum

NTNU

Kunnskap for en bedre verden

Abstract

The overall goal of this project is to develop a method for generating precise estimates of flow properties. This thesis describes an attempt to use a convolutional neural network (CNN) to predict the inter-pore hydraulic conductivity of a rock sample subvolume. The images used as input data were generated from a grid. The grid is a subvolume from a 3D CT-scan of a rock. The images were generated in a way that conceivably would be comprehensible for a CNN model.
Different CNN models were built using different image resolutions, amount of training data, training sample distribution, and image dimensions (2D and 3D). Eventually, a CNN model was used to provide a Pore Network Model with predictions for the hydraulic conductivities. The permeability of the network model was estimated and compared to the numerical estimation.

Sammendrag

Det overordnede målet for denne oppgaven er å utvikle en metode for å generere presise estimat av strømningsegenskaper. Oppgaven beskriver et forsøk på å bruke et konvolusjonelt nevralt nettverk (CNN) til å predikere den hydrauliske konduktiviteten for sammenkoblede pore-par i et undervolum av en steinprøve. Bildene som brukes som input data ble generert fra et grid. Griddet er et delvolum hentet fra en 3D CT-scan av en steinprøve. Bildene ble generert på en måte som kan tenkes å være forståelig for en CNN-modell.
Ulike CNN-modeller ble bygget ved bruk av ulik bildeoppløsning, mengde treningsdata, treningsdatafordeling, og bildedimensjoner (2D og 3D). Til slutt ble det brukt en CNN-modell til å tildele predikerte hydrauliske konduktiviterer til en Pore-Nettverks Modell. Permeabiliteten for modellen ble estimert og sammenlignet med det nummeriske estimatet.

Acknowledgements

This thesis is the final product of a 5-year study program and countless hours of listening to Elvis Presley. It is the master's thesis for a MSc program in Petroleum Geosciences and Engineering, specializing in Reservoir Engineering and Petrophysics. It was written at the Norwegian University of Science and Technology at the Department of Geoscience and Petroleum.
Thank you, Marie-Laure Olivier, for providing me a desk at PoreLab, and thank you, PoreLab, for fresh fruit each Monday. A special thank you to Michael Reuter for lending me his stationary computer at PoreLab. Without it, I would have remarkably fewer results.
I want to thank my supervisor, Carl Fredrik Berg, who provided me with excellent guidance and music recommendations. The Matthaus Passion was depressing.
Finally, a special thank you to Even Olsen, and Øya Helsehus for letting me play Beat Saber during breaks.

Contents

Figures

Tables

Chapter 1

Introduction

Porous media is everywhere. A blackboard sponge is a porous medium used for removing chalk. When a sponge is filled with water, it holds the water in its pore space until it is forced to let it go. This could be due to compaction of the pore space when the sponge is pushed against the blackboard or due to the water vaporizing in the summer heat. Another example of porous media is outdoor clothing. Shoes that are waterproof and "breathe" at the same time are designed by people that know porous media. The shoes are supposed to let air in and out and, at the same time, prevent water from passing through. Describing the properties of porous media is a topic of interest in many areas, especially in the oil and gas industry, where the properties of porous media, that is, the reservoir, are of great value.

Pore scale modeling is a tool used give a better understanding about porous media's physical and chemical processes. In 1956, Fatt introduced the network of tubes. This was a step toward achieving a better understanding of flow and transport in porous media. Since Fatt introduced his first groundbreaking contribution to the field, pore scale modeling has progressed rapidly along with the development of new and more powerful computer technology, especially in the last 20 years (M. J. Blunt, Bijeljic *et al.* 2013). The network of tubes originally intended to try and understand displacement processes (Fatt 1956). Now, nearly 70 years later, pore scale modeling has become a tool used for both scientific and commercial purposes, e.g., in the oil and gas industry. There are different ways to obtain images of the pore space of porous media. CT-scans are one of the standard 3D-imaging tools. Such imaging of rock samples provides information about the pore space and mineralogy that would be more or less impossible to obtain using traditional experimental methods (M. J. Blunt, Bijeljic *et al.* 2013). Some companies have their expertise in digital core analysis and offer services for the oil and gas industry. For an oil and gas company, one of the most valuable properties is the permeability of the reservoir. It describes how easily oil or gas may flow in

the reservoir. For, e.g., field optimization, a precise estimation of the permeability yield valuable information.

While the global properties may be the most interesting for commercial oil companies, local properties could be just as crucial regarding porous media science. A global property may refer to the hydraulic conductance of a reservoir, while a local property could be the hydraulic conductance between two interconnected pores in that reservoir. New information about local properties may alter how the industry measures the basic rock properties. Such properties are used in reservoir characterization and performance prediction.

In recent years, the application of artificial intelligence (AI) has increased rapidly also in the oil and gas industry. One example is the use of AI to classify lithological facies. This was done using a subcategory of AI called a convolutional neural network (CNN) (Imamverdiyev and Sukhostat 2018). Other applications of AI are estimating total recoverable reserve volumes, optimizing well placement, analyzing reservoir data, and much more.

This thesis will describe an attempt to train CNN models to predict the hydraulic conductance between a pair of interconnected pores. Images of the subvolume between two interconnected pores will be used as training data, and the effective hydraulic conductance of the pore body-throat-body system will be used as labels. The images will be generated from 3D CT-scan images of a sandstone sample. It is crucial that the images capture the essential features needed for the prediction of hydraulic conductance.

In this project, the following will be attempted:

- To develop methods for generating images from a subvolume of the 3D image of the rock sample to be used for training of a CNN-model, and to structure the images in such a way that the model is able to extract useful information from them.

- To find a proper way to scale the labels based on the spatial extent of the subvolume from which it is calculated. This is done to eliminate the differences in hydraulic conductance caused by the size differences of subvolumes.

- To train a model that shows clear signs that it can extract information from the images and, by that, give sensible predictions.

- To train multiple models using different training set sizes, different image resolution, different image structuring or image post-processing, and compare the results.

- To use a model for predicting the full data set of images and use those pre-

dictions for pore network property calculation and then compare the results with results of the conventional pore network model.

The structure of the thesis will now be described. Chapter 2, Background, first describes digital rock modeling and imaging. Then follows a description of some parts of the scientific work that has contributed to the work on pore scale modeling, from Fatt (1956) to modern pore network modeling. Since this thesis concerns CNN, some background regarding AI and machine learning (ML) will be introduced at the end of the Background chapter. A more detailed description of CNN architecture, convolutional layers, and other functions will be given in Chapter 3, the Theory chapter. This chapter also describes the method for calculating the effective hydraulic conductance and a section about the Stokes equation, which will be used when scaling the labels. The Materials chapter, Chapter 4 describes the grid and `link.dat`- and `node.dat`-files used in this project. The `.dat`-files store information about the pore throats and bodies, such as length, radii, location, shape factor, and more. It also describes the codes for building a CNN-model that handles 2D or 3D input. The inputs are images generated from the grid. How these images are generated and structured is described in Chapter 5, Methodology. In Chapter 6, Results Discussion, the results of training the different models will be presented and discussed. Finally, the Conclusion is found in Chapter 7, followed by a suggestion for Future Work, Chapter 8.

Since this thesis is a continuation of the project report from last semester, there are similarities. Parts of the chapters Background, Theory, and Methodology are similar to the project report regarding content and structuring.

Attached is a link to a GitHub page where the relevant python codes can be found. The codes were used to generate input data and for building CNN models. GitHub: https://github.com/TormodKa/Thesis.git

Chapter 2

Background

2.1 Digital Rock Physics

Digital rock physics (DRP) is commonly used to denote methods to simulate rock properties from digital representations of the pore space of a porous medium. Over the last decade, the use of DRP has increased due to the advancing technology and availability of necessary equipment (Wildenschild and Sheppard 2013). DRP involves both imaging and analysis and combines different methods for the two. Proper use of DRP can provide estimates of properties that are seen as valuable. Such valuable properties can be hydraulic and electrical conductivity. This is done by constructing a digital 3D model of the porous medium and then simulating physical processes (Andrä *et al.* 2013). In addition, experimental methods are costly and time-consuming. SCAL (special core analysis) properties are valued in the oil and gas industry. Numerical methods that are able to estimate SCAL properties are therefore seen as a valuable tool for saving time and money.

Simulations on digital representations of porous media are mainly performed by one of two different methods. Direct simulation on a grid representation is the most computational demanding method. An advantage with this method is that the simulations are conducted on binarized images of the pore space. This means that the original geometry of the porous medium is maintained. There are several different simulation techniques in existence. The most popular is the lattice-Boltzmann method (Martin J. Blunt 2017). It is a particle-based method that approximates the Navier-Stokes equation, and the method is fairly simple to implement (C. F. Berg *et al.* 2017). The second common method for simulations on digital representations of porous media is called *Pore network modelling* (PNM). In this method, physical processes are simulated on a network representation of the void space.

2.2 Imaging

Imaging is important in the context of DRP, and it is done for mainly two purposes: 1) To obtain descriptive information about the rock sample. Imaging of the rock sample could yield information about grain size- and pore size-distribution, information about mineralogy and clay content, to mention some. 2) Imaging is also important for obtaining a 3D structure of the rock sample to be used for computation of e.g., the pore space, needed for transport simulations.

2.2.1 2D imaging

Scanning electron microscopes (SEM) was the most common tool for producing high-resolution images of a rock sample. Where optical microscopes uses light rays, SEMs use electrons to generate images of the sample. This could be viewed as a destructive method, meaning that it destroys the rock sample. The imaging it self is not destructive, but the sample preparation is. For the purpose of imaging it is common to use end caps from plugs, as the use of such end caps for other purposes is limited (C. F. Berg *et al.* 2017). The next concept that will be explained is the concept of back-scattered electrons (BSE).

BSE, or back-scattered electrons, is a technique applied to acquire images of sub-micron scaled features, commonly used in rock characterization (C. F. Berg *et al.* 2017). Initially, the rock sample is covered by epoxy and placed in a vacuum chamber until the sample is filled with epoxy. Then a thin section of the sample is cut off, fixed to a glass plate and polished until the surface is flat with a thickness of 20 μm. When the thin section is fixed and ready, it is bombarded with electrons from an electron beam. Some of the electrons will be pulled towards the opposite charged nucleons, while some will be back-scattered out of the sample.

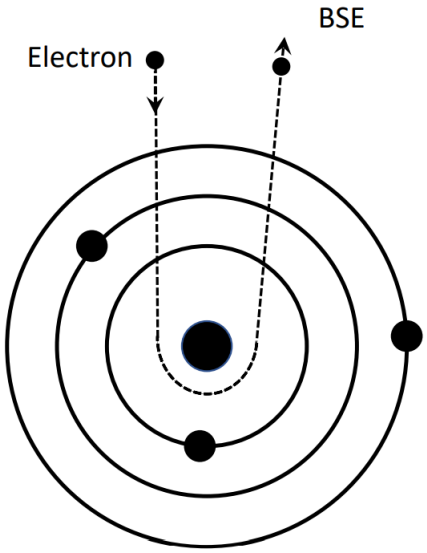


Figure 2.1: Illustration of BSE. A cartoon-ish atom back-scatters an electron.

Above the sample are detectors consisting of a material with the traits of a semiconductor. A conductor of this kind is only affected by high-energy electrons, which makes them a good detector of back-scattered electrons. When an electron collide with the detector, the electron and the semiconductor reacts and creates an electron-hole pair. A recom-

bination process occurs to set up an equilibrium, which means that the electron-hole pair disappears (Joseph M Polese 2016). If the free electrons and pairs are separated before being recombinated, an electrical current will be generated. The current is measured by an electronic circuit, and eventually converted into an image. A heavy element will deflect electrons more strongly than lighter elements. A heavier element will therefore appear brighter in a SEM image (Sok *et al.* 2009). The depth of penetration depends on the acceleration voltage beam. A high acceleration voltage beam increase the penetration death, while lower acceleration voltage beams makes it easier to examine the surface layers. For a grey scale images, the epoxy-filled pores will appear darker than the denser grains (Nadeau and Hurst 1991).

2.2.2 3D imaging

X-ray computed tomography (CT) is a fully non-destructive technique for creating 3D digital images of a sample. CT-imaging reconstructs a 3D image of a rock sample by capturing X-ray images of the sample from different angles (Wildenschild and Sheppard 2013). This kind of technique has been applied for decades in the oil industry for many different purposes (Cromwell *et al.* 1984). The scientists Gilliland and Coles (1990), Honarpour *et al.* (1985), Hove *et al.* (1987) and Wellington and Vinegar (1987) all studied the possible areas to apply this imaging technique within the oil and gas industry. Core inspections, where to drill plug samples, core plug inspections, rock characterization and imaging of flooding experiments are some of the typical applications. For these kinds of application it is common to use a so called medical CT-scanner, similar to the ones used in the medical industry, where the sample is fixed while the source and detectors rotate around it (C. F. Berg *et al.* 2017).

There are different micro-tomography systems. Three main systems are medical CT, micro-CT and synchotron micro-tomography. They all vary in resolution, and are therefore used for different purposes. Rock property calculations require high enough resolution to resolve all important parts of the structure. A standard medical CT-scanner has a resolution between 200-500 μm. This does not provide high-enough image resolution for the purpose of rock sample imaging of the pore space. The beam line emitted from electron acceleration at a synchrotron facility was used as the X-ray source for the first high-resolution rock sample images (Flannery *et al.* 1987). This is usually no longer used for regular sample imaging, but more often used to capture transient processes on the pore scale (S. Berg *et al.* 2013). Micro-CT scanners are more commonly used for DRP. With a resolution down to approximately 1 μm, they capture the pore space structure more precisely than the medical CT scanner (C. F. Berg *et al.* 2017).

2.2.3 2D-to-3D reconstruction

The 2D images from Subsection 2.2.1 solely contain 2D information about the rock sample. DRP calculations require the 2D images to be reconstructed into 3D images (C. F. Berg *et al.* 2017). The structure of a rock is not random, it is the result of a geological process. Trying to replicate the rock forming process gives valuable information of the rock sample, and at the same time it links together the structure of the pores to the geological process that formed the rock. This supports two of the main objectives when reconstruction reconstructing a rock sample: 1) improved network extraction, 2) relating pore space and the process that fromed it. A 3D reconstruction with incorporated prior information could improve the accuracy of the network extraction, while relating the pore structure to the process that formed the rock is valuable for rock characterization (C. F. Berg *et al.* 2017).

As mentioned in Section 2.3.1, Bakke and P.-E. Øren (1997) introduced a process based 3D reconstruction. They applied three steps of sedimentation, compaction and diagenesis to mimic the rock forming process. It yields accurate predictions for permeability and conductivity, compared to typical statistical reconstruction techniques (Hilfer and Manwart 2001).

2.3 Pore Network Modelling

Several properties on a macroscopic scale are determined by the physical characteristics of the fluid in the pore space and the solid around it. In addition, the microstructure of the porous medium plays a role. Transport properties, such as permeability, relative permeability, formation factor and capillary pressure are all properties of interest for the oil and gas industry, as well as for other fields. To estimate these transport properties, the equations describing the physical processes that takes place on the pore scale needs to be appropriately averaged. A tool that is commonly applied in this kind of investigation is the network model (Øren *et al.* 1998).

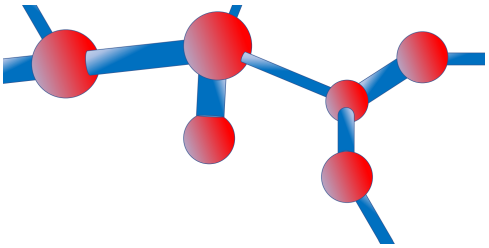


Figure 2.2: Illustration of how the pore network elements are represented in a pore network model. Red spheres represents pore bodies, while the blue tubes represents pore throats.

The pore space observed in porous media is often complex and chaotic. Fig. 2.3 illustrates such a pore space. In a network model, the complex and chaotic pore space of a porous medium is simplified. The void space of a rock

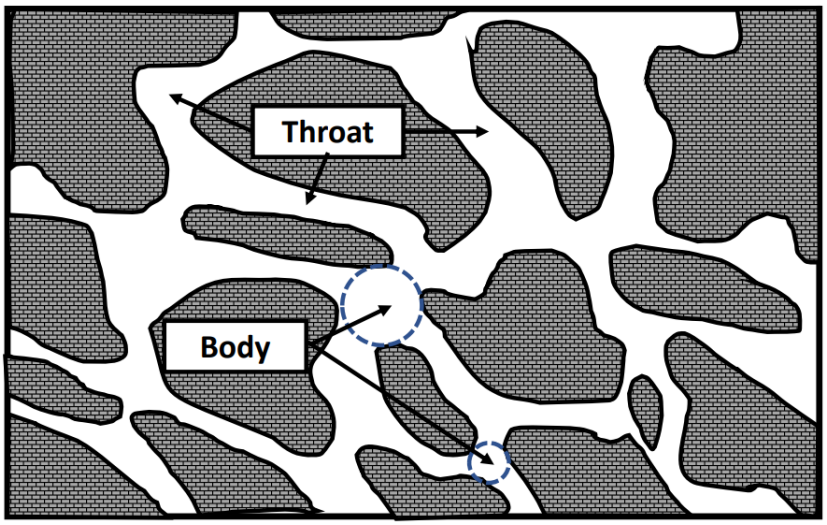


Figure 2.3: Illustration of pore body and pore throats in a cartoonish porous medium. The two dotted circles indicates two of the many pore bodies in the 2D pore space.

matrix is represented by a network of throats and pores. Throats are narrow passages where fluid may flow through, and where two or more throats meet is a larger void space called a pore body. This is illustrated in Fig. 2.4, which shows a PNM approach to Fig. 2.3. In many cases, the pore and throat surfaces can be highly irregular. It may even be made up of different materials.In a pore network model, this is simplified, and due to the simplifications, details in the representation of the void space is not very accurate. The level of detail accuracy needed depends on for what it is going to be applied (M. Sahimi 2011). A common representation of the pore network model, is the stick-and-balls approach, as illustrated in Fig. 2.2. It illustrates the simplifications done when building a PNM from a rock sample. The pore network is used as input when simulating e.g. drainage or water injection.

2.3.1 History

Fatt (1956)

Fatt introduced the first pore network model in 1956. At that time, the science concerning porous media was simplified and largely based on empirical data. Fatt introduced his work by stating that much of the science regarding flow through porous media is far too simplified to give any quality predictions. In addition, Fatt claimed that the absence of any well founded theoretical description of fluid flow in porous media had lead to too many empirical descriptions. Some equations

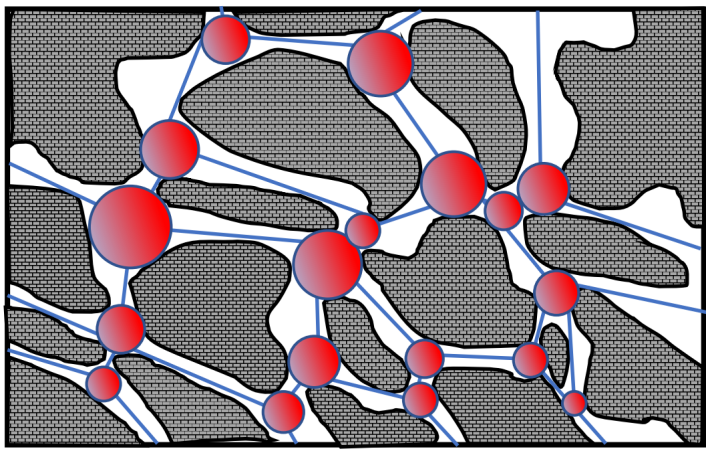


Figure 2.4: Fig. 2.3 including pore network approximated elements.

were derived from far too simplified geometric models, but they were still used for scientific purposes. Two models that was commonly used at the time was the sphere pack and the bundle of tubes. Fatt claimed they were too simple. A consequence of the simplicity of the model was that the equations derived from them lead to inadequate predictions.

Since the bundle of tubes model is oversimplified, its results were of limited applicability when applied to real systems. An example of the bundle of tubes model is illustrated in Fig. 2.6. Valvatne and M. J. Blunt (2004) pointed out that the bundle of tubes model had difficulties with predicting multiphase data, such as the capillary pressure and relative permeabilities, that are influenced by the spatial distribution and the connectivity of the pores and throats. Real porous media are more or less isotropic with respect to fluid flow. The major weakness of the bundle of tubes model is that it is perfectly anisotropic. It fails to resemble real porous media by the absence of connection between the tubes.Cross-connection between pores is a major structural feature of porous media, as can be indicated by examining sandstone thin sections. Interconnected pores were indeed included in the sphere pack model, but the shape of the pores were so complex that analyzing flow through them would be very demanding.

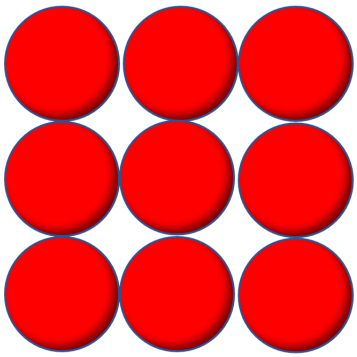


Figure 2.5: Illustration of a simple two-dimensional sphere pack model.

When it comes to the sphere pack model, it oversimplified the structure, but was still often too complex for theoretical studies. As a result of this, Fatt (1956) wanted to come up with a

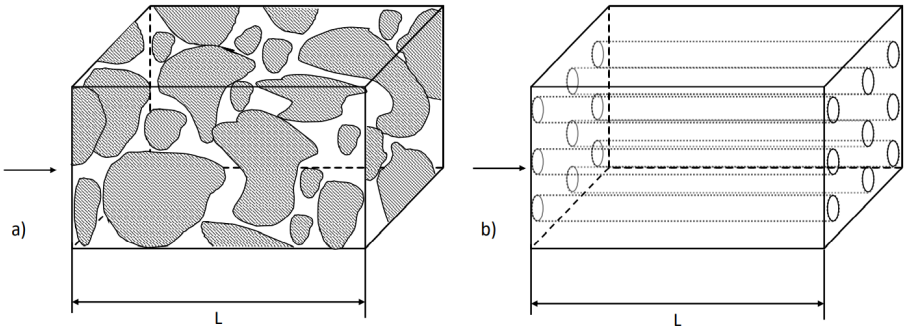


Figure 2.6: Illustration of the bundle of tube model. Image "a)" is a complex porous medium, while image "b)" is a simplified bundle of tube model of the same porous medium.

model that resembled most real porous media more closely but still simple enough to derive some useful information from it. Driven by the desire to develop a better understanding of the various measurable properties of porous media, and the relation between them, he came up with the network model.

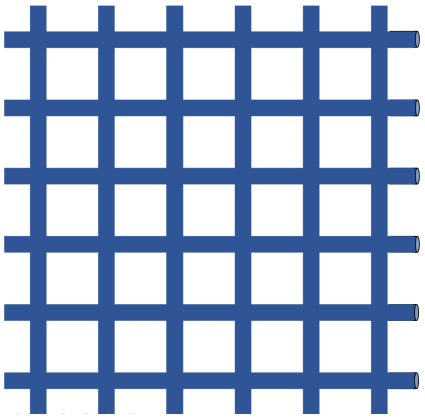


Figure 2.7: Two-dimensional square network of tubes.

It has cylindrical tubes, but Fatt (1956) argues that cylindrical pores could be a fair simplification of complex shapes for some applications. Fatt (1956) combined these methods, by keeping the structure of the sphere pack model, but substituting the complex pore shapes with uniform cylinders. The goal was to create a model which resembled porous media accurately, yet simple enough to be used for flow calculations. Combining the bundle of tubes model and the sphere pack model resulted in the three-dimensional network of tubes, as illustrated in Fig. 2.7.

Even though the three-dimensional network of tubes was a step in the right direction, the calculations still were a bit complex. To make the calculations simpler, Fatt (1956) replaced the three-dimensional network by a two-dimensional network. By assuming that a very thin slice of the porous medium would has the same properties as a cube of the same structure, and compensating for lost cross-connections by introducing additional channels within the two-dimensional

network, Fatt (1956) justified replacing the three-dimensional network by a two-dimensional network.

Chatzis and Dullien (1977)

A number of scientists continued the work on two-dimensional network models of capillary segments after Fatt (1956) published the first significant contribution to the field. Eventually, a three-dimensional model was introduced.

Two decades after Fatt's publication, Chatzis and Dullien (1977) introduced a three-dimensional network model, after discovering room for improvement in the two-dimensional model introduced by Fatt (1956). During a study of the properties of two-dimensional and three-dimensional network models of capillary tubes, Chatzis and Dullien (1977) found that the model proposed by Fatt (1956) was unable to give accurate estimates of three-dimensional flow. In Fatt's model, the effect of cross-connected pores was not covered well enough. Chatzis and Dullien (1977) introduced an average coordination number Z, which defines the average number of connections for each pore. An actual network of pores needs to be simplified for the purpose of analysis. The goal was to replace the irregular network of pores with geometrically ordered networks characterized by the number Z, defined as

$$Z = \sum_r Z_r \mathrm{f}_r, \tag{2.1}$$

where

$$Z_r = \left(\sum m\right)_r /2 + 1. \tag{2.2}$$

where $(\Sigma m)_r$ is the number of pores, connected to a pore of type r, at both ends, and f_r is the relative frequency of such pores in the lattice (Chatzis and Dullien 1977). It is clear that Z indicates the pore interconnectivity of the network (Chatzis and Dullien 1977). A Z value of 2 means that the average pore is connected to 2 pores. A square network lattice would have $Z = 4$.

To analyse the precision of the models, the two-dimensional and three-dimensional saturation curves and experimental curves were compared. The three-dimensional saturation curves were obtained by approximating the data using three-dimensional bond percolation results. These results were provided by (Frisch *et al.* 1962). The percolation problem was introduced by the mathematicians Broadbent and Hammersley (1957). On a lattice, each bond connecting two nodes has a constant probability p of being "open". If a bond opens in this case, it means it will transmit fluid. The "critical bond percolation" probability is the minimum fraction of the bonds that needs to be open for fluid to be transmitted from one end to the other. In other words, the probability for a breakthrough (Broadbent and Hammersley 1957).

Chatzis and Dullien (1977) discovered that the spatial interconectivity of the two-dimensional model was not efficient for modelling flow calculations. The investigation was done by comparing the breakthrough time of a two-dimensional square and three-dimensional tetrahedral network, both with equal coordination number $Z = 4$ and equal tube/capillary diameter distribution. They discovered that the breakthrough of a three-dimensional network occurs at a lower capillary pressure, P_c, than for a two-dimensional network. The Young-Laplace equation defines the capillary pressure as

$$P_c = \frac{2\sigma \cos\theta}{r}, \tag{2.3}$$

where σ is the interfacial tension, r is the radius of the capillary, in this case a pore, and θ is the angle between the two fluids and the capillary tube.

Chatzis and Dullien (1977) concluded their investigation stating that the existing two-dimensional and three-dimensional network models of cylindrical tubes yielded unrealistic simulations of sandstone pore structures. In addition, two-dimensional network models was not suited for two-phase flow simulations, due to the fact that bicontinua cannot exist in two-dimensional network models.

Pore Network Modelling

In 1992, Bryant and M. Blunt (1992) introduced another major breakthrough building on the work of Fatt (1956). Using random packed equal sized spheres, they were able to make predictive calculations of two-phase relative permeabilities. The predictions of their model closely matched existing results form experiments. Although their model was restricted to simple, well sorted porous media, M. J. Blunt, Jackson *et al.* (2002) described it as a major triumph for pore scale modeling.

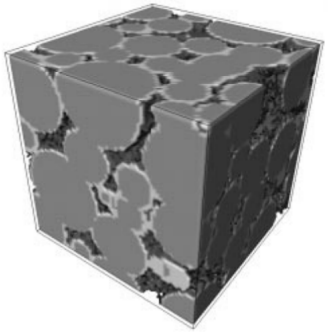


Figure 2.8: Visualization of the simulated rock by Bakke and P.-E. Øren 1997. The model is based on a Bentheimer sandstone.

Bakke and P.-E. Øren (1997) extended the method proposed by Bryant and M. Blunt (1992) by making the spheres randomly sized. Grain-size of each individual grain was randomly selected from grain-size distribution curves. They were obtained from BSE-images from thin sections of the actual sandstone. Not only did the method cover the sedimentation of the model better than the previous methods, it also included modelling of compaction and diagenesis Øren *et al.* (1998).

Compaction is modelled by a linear shift in the vertical direction, while diagenesis is modelled to simulate quartz overgrowth, clay and cement precipitation. This method for generating 3D pore-scale models was both cost-effective and rapid, at the time of publication (Bakke and P.-E. Øren 1997).

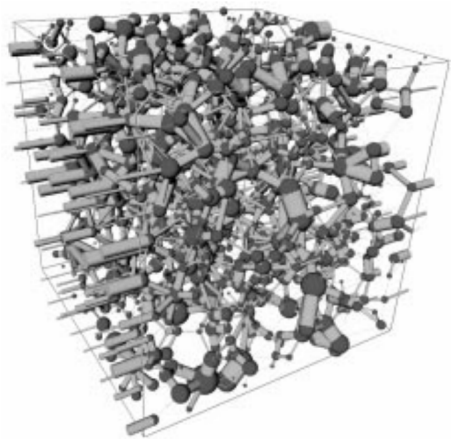


Figure 2.9: Ball-and-stick representation of the same Bentheimer sandstone as in fig 2.8. Image is taken from Bakke and P.-E. Øren 1997.

Pore network modelling was divided into two categories when Øren *et al.* (1998) introduced the quasi-static model. They assumed that capillary forces are dominant at pore scale, and claims that it is reasonable for capillary numbers of 10^{-6} or less (Øren *et al.* 1998). A capillary number is a dimensionless number describing the interface between a liquid and a gas, or two immiscible liquids. It represents the relative effect of viscous drag forces versus surface tension forces acting across an interface between the two fluids, defined as $Ca = \mu V/\sigma$, where V is a characteristic velocity, μ is the dynamic viscosity of the liquid and σ is the interfacial tension between the two fluid phases. Afew years later, Ding and Kantzas (2004) stated that capillary forces dominate the flow in porous media for low capillary numbers. Their rule of thumb was that $Ca < 10^{-5}$ is low, agreeing with the assumption of Øren *et al.* (1998). The quasi-static model simulates displacement of one fluid while another fluid fills one pore at a time (Gjennestad *et al.* 2018). If a pore is being filled or not is determined by the capillary entry pressure. Narrow pores demand higher pressure than a broader. In situations where flow rates are low and the viscous pressure is small enough to be neglected, capillary forces are assumed to be dominant (Gjennestad *et al.* 2018). The dynamic models accounts for the viscous pressure drop that is neglected in the quasi-static model. It captures the interaction between capillary and viscous forces. Pore network modelling has seen a rapid improvement when it comes to handling of complexity, hand in hand with the rapidly improving computer technology. Koplik and Lasseter (1985) modelled fluid propagation and displacement dynamically. They used approximate solutions of the Navier-Stokes equations to calculate two-phase flow in random networks. The motivation was to investigate how the macroscopic characteristics of fluid flow in porous media are affected by the microscopic geometry.

The art of network modelling has surely advanced since Fatt (1956) took advantage of the analogy between flow in porous media and a resistor network. Now, there are models that can handle irregular lattices, arbitrary wettability,

non-Newtonian fluids, non-Darcy flow, two- and three-phase flow and much more (M. J. Blunt, Bijeljic *et al.* 2013). At the end of Section 2.4, two recent studies concerning digital porous media will be introduced. Both of the studies is done using convolutional neural networks, which will be further explained in Section 3.1.

2.4 Machine Learning

Imagine being given the task of writing a program that would be able to decide if an image contains a turtle or a tortoise. The program should be able to differentiate between the two animals, which may look quite the same. To be able to classify the animals, the person writing the program would have to be very creative with coming up with rules for edges, curves and colors. In addition, the program would have to take into account all possible angles that the animals could be seen from, as well as color scaling and maybe even inverted colors. A program like that would end up being extremely complex. If the program suddenly would have to be able to classify a terrapin as well, the program would need to be extended, which would possibly be a lot of work. With machine learning (ML), this kind of problems can be solved by less manual programming. The machine learning model would, in this case, be given thousands of images of turtles, tortoises and terrapins. From these images, the model extracts information from patterns in the images. There are a lot of subcategories in ML, and ML itself is a subset of artificial intelligence.

2.4.1 Artificial Intelligence

Artificial intelligence (AI) is described as the development or study of computer systems able to copy human-like behaviour. These behaviours may be in learning, reasoning or self-correction (Kok *et al.* 2009) One advantage of AI is increased efficiency. Computers do not get tired, and can therefore be leveraged to perform small, repetitive tasks or computational heavy tasks to enable humans to be more efficient. Another advantage is the decreased rate of human error. A repetitive task can be particularly prone to human error, as the human brain gets tired after a long time of focus. AI systems trained to perform one specific task will not have to deal with fatigue, and will therefore be able to perform a task for as long as needed. Several AI methods are developed for different areas of application. Fig. 2.10 illustrates subsets of AI. Each of the different subcategories has its own subcategory. In Fig. 2.10, this is only illustrated for ML, since that is the focus in this thesis.

AI methods for recognizing human speech and language is often used for systems using voice control, e.g. Google Assistant and Apple's Siri. Some other subsets of AI is is robotics, planning, vision, expert systems and machine learning, which now will be explained more closely.

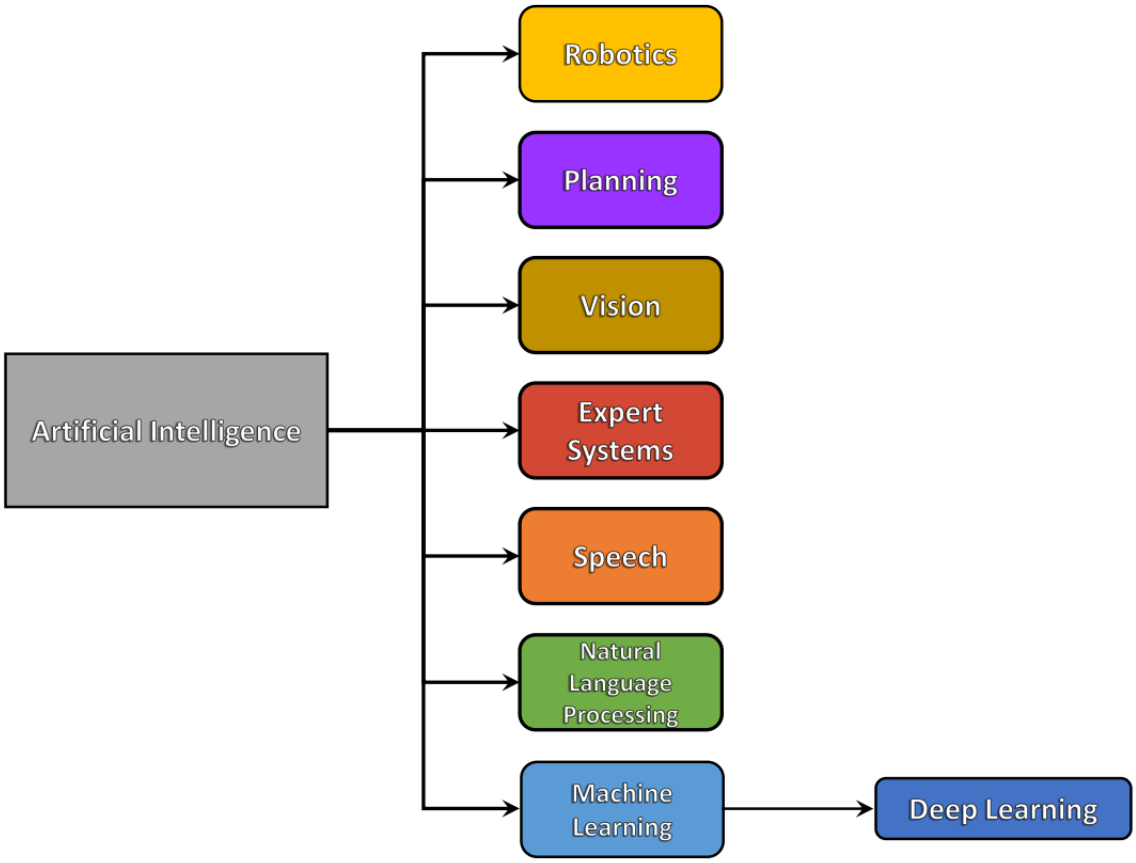


Figure 2.10: Illustration of different subcategories of AI. There are a lot of subcategories to all categories, but the focus in this project will be on deep learning as a subcategory of machine learning.

Machine Learning

Machine learning is a technique for building a model that is able to make decisions or predictions. The model is built based on sample data (or training data), and makes predictions or decisions based on experience from the data (Koza *et al.* 1996). Tom Michael Mitchell, an American computer scientist stated as early as in 1997 that ML is one of the ways we expect to achieve AI. Mitchell was a pioneer in ML, and describes ML as the study of computer algorithms that allow computer programs to automatically improve through experience (Mitchell 1997).

Machine learning is a collective term for several methods. ML as a subset of AI provides statistical tools to explore and analyse data, that improve at tasks by training. Which of the different methods that should be used is dependent on the problem trying to be solved. Two different approaches would be, e.g., to use a ML model to estimate a value, or to use it to classify an object in an image. A *Regressor* returns a value or a vector. A *Classifier* returns a class label, and would be suitable for the problem in the introduction of Section 2.4. It is common to split machine learning into three major learning paradigms: Supervised learning, unsupervised learning and reinforcement learning. There are variations and combinations, but these three are viewed as the main paradigms (Lampropoulos and Tsihrintzis 2015).

Supervised Learning

Supervised learning may be used for both prediction and classification tasks. In supervised learning the input is paired with the output, which means that the machine receives an input and its associated target value, or label (Lampropoulos and Tsihrintzis 2015). Supervised learning may be used to build a model that is able to classify data based on numerical or categorical values, or predict trends using previous labeled data (Kotsiantis *et al.* 2006). Labeled data means that the target value, or expected output, of a problem is known. When training, the machine gets both the input and the output. For example, let us say that you train a model using weather data from many days, combined with the outdoor clothing that you wore each day. After training is finished, the model should be able to predict which clothing you are going to wear the coming days, based on the weather forecast. There are many algorithms of supervised learning, but the most common ones are Support-vector Machines, Linear Regression, Logistic Regression, Decision Trees and Neural Networks (Daniel Westrich *et al.* 2009).

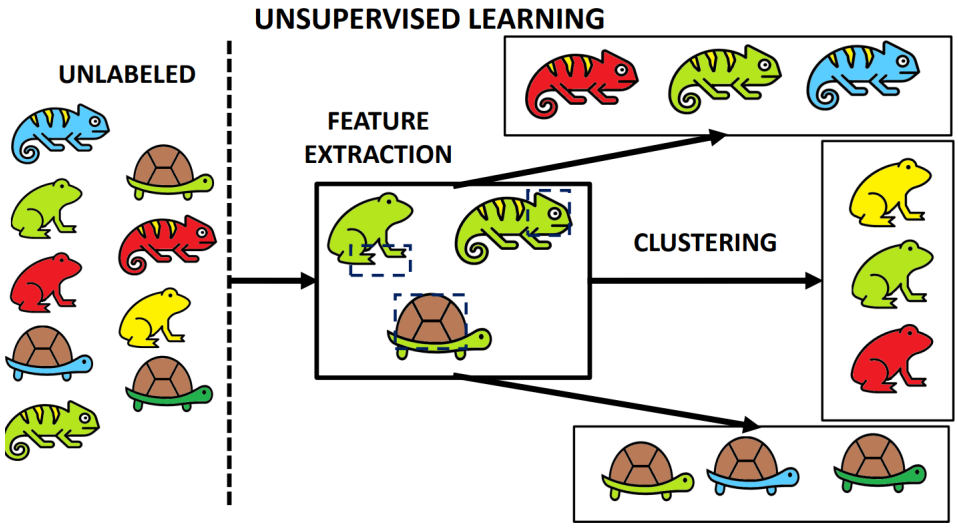


Figure 2.11: Cartoonish illustration of an unsupervised learning clustering algorithm. In this example, three different colored chameleons, frogs and tortoises are given as input. The model extracts features from the images, discovering that some of the images contain a round, brown shell, some contain flat, duck-like feet, and some images contain a head with a bony protrusion. The images are then clustered based on the features found in the images.

Unsupervised Learning

Unsupervised learning is when the model is trained without labeled data. The model then attempts to group, or cluster, the data based on similarities (Lampropoulos and Tsihrintzis 2015). A typical application for unsupervised learning is recognition-tasks, situations where the label is unknown or even situations where picking out characteristic features is difficult (Kotsiantis *et al.* 2006). Returning to the example from the intro of Section 2.4. Let us say it is difficult for a human to specify which features that makes the difference between a turtle and a tortoise. By passing enough images of turtles and tortoises, the model may extract the important features and cluster the images based on those features. The model has never seen the label of the images, so it will cluster the images based on common features it extracts from the images. Fig. 2.11 illustrates an example of what unsupervised learning may look like. A downside of unsupervised training is that the accuracy of the model is not measured during training. In supervised training, there are several tests for validation of the training. These tests are few when it comes to unsupervised learning, which makes it more difficult to ensure that the outcome of the model is accurate (R. Sathya and Annamma Abraham 2013). As the machine is creating its own outcome based on the features it has extracted itself, the result is a less controllable environment (Dike *et al.* 2018). As for the supervised learning method, there are a lot of different approaches to unsupervised learning. Clustering methods, anomaly detection methods and neural networks

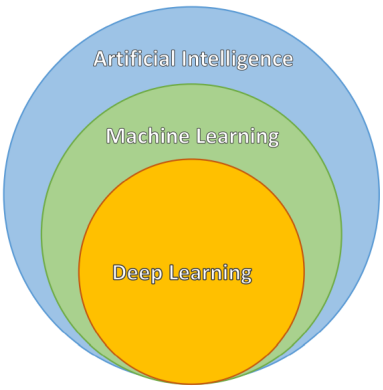


Figure 2.12: Illustration of the relation between artificial intelligence, machine learning and deep learning. DL is a subset of ML, which again is a subset of AI.

are some of them. Neural networks will be explained further in Section 2.5.

Reinforcement Learning

Reinforcement learning is based on reward and punishment. It is an efficient method to develop goal directed action strategies (Lampropoulos and Tsihrintzis 2015). The programmer defines which actions that should be rewarded and which actions that should be punished, while the model is told to hunt for the highest possible reward. The model then learns how to achieve the rewards by trial and error (D. Zhang *et al.* 2018). A common example when explaining reinforcement learning is teaching a model to play *Pac-Man*. Pac-Man is a famous video game where a yellow, circular head moves around within a dark maze. The character hunts for rewarding items while dodging ghosts in the maze. For a model to learn how to play Pac-Man using reinforcement learning, the model may be told to hunt for the highest possible total numeric reward. Then, when the model plays the game, it is given a numeric reward for collecting items along the way, with some items dealing a higher reward than others. If the model hits a ghost inside the maze, it leads to a deduction in total reward. Through trial and error, the model will gradually develop strategies to win the game.

Hyperparameters

Hyperparameters are parameters that are used to control the learning process. An example of a hyperparameter is the learning rate. When a ML model is training, it updates its model parameters depending on how well the model predicts during training. The learning rate decides how much the model parameters are

influenced by the results from its predictions during training. Learning rate will be explained further in Section 3.1.2. Optimal hyperparameteres are vital for the training of a ML model (Jia Wu *et al.* 2019). Usually, the hyperparameters are set by training the model on a small subset of the data set, a *mini-batch*. This is done before the actual training begins (Jianxin Wu 2017). Setting the hyperparameters is called hyperparameter optimization or hyperparameter tuning (Jia Wu *et al.* 2019). Hyperparameters and model parameters are not the same. Hyperparameter tuning is done before training the model, and are used in the process of deciding the model parameters. Hyperparameters affect both the quality and the speed of the training process. Different training algorithms may require different hyperparameters. Some require none (Jia Wu *et al.* 2019). Some hyperparameters may be dependent on other hyperparameters, e.g., the size of some layers can depend on the overall number of layers. A detailed introduction to layers can be found in Section 2.5.1 and Section 2.5.2.

2.5 Artificial Neural Networks

In the brain of a human or animal, connected nerve cells communicate by transmitting electrical signals between each other. The receiving nerve cell processes the signal and may transmit a signal to its connected nerve cells. Artifical Neural Networks (ANNs) are inspired by the way nerve cells in a biological brain communicates (S.-C. Wang 2003). In an ANN, often called neural network (NN), a collection of artificial nerve cells (neurons) are connected by links in various patterns, and they transmit real numbers. One neurons output is an other's input. Each neuron may receive input form multiple other neurons. Each link connecting the neurons has an associated weight which determine the impact of the received input. The weights are adjusted through out the training. To train a model is actually trying to optimize the weights within the model (S.-C. Wang 2003). Fig. 2.13 illustrates the communication of three neurons.

Sections concerning ML and its subsets will from now be focused towards supervised learning, because supervised learning is what has been used in this thesis. Supervised learning commonly include three different data sets that are used for three different purposes. Training data, validation data and test data. A brief explanation is that the model i trained using the training data set, during training the training of the model is validated using data from the validation set, and after training is finished, the model is tested using the test set. As mentioned in Section 2.4.1, a model is usually built by first tuning hyperparameters on a mini-batch of training data (Jia Wu *et al.* 2019). During hyperparameter tuning, the validation set is used to give an estimate of the predictive skills of the model. After hyperparameter tuning is done, the model is trained on *batches* of training data. A batch is the number of training samples utilized in each *epoch*. The number

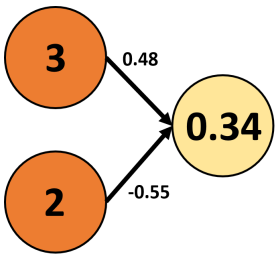


Figure 2.13: The output of the left neurons is 3 and 2. Both neurons transmit information to the same neuron. The receiving neuron is connected to the two neurons by links. Each link has its own weight, which impact the output value. Output value 3 is multiplied by 0.48, and output value 2 is multiplied by -0.55 before being received by the neuron on the right. The input value is the sum of the weighted outputs, which in this example is 0.34.

of *epochs* defines the number of batches processed before the training is finished. When the training is completed, the model is set to predict the test data set. This is done to give an estimate of the predictive skills of the final model (S.-C. Wang 2003).

2.5.1 Components of an Artificial Neural Network

Layers

An ANN model has an input layer and an output layer. External data, such as an image, is input for the input layer. The output for the output layer is the output of the model. The type of input differs depending on the task. Let us say the input is an image containing 5 wolves. For a classifier, the output could be a class label. For example, the model recognizes the animal in the image as a wolf and outputs “wolf”. For a regressor, the output is commonly a real number. For example, it could be used to estimate the weight of the wolves in the image, and the output would be a vector of weights. Between the input and the output layer, the model usually have multiple *hidden layers*, which are layers that are not directly observable. Their role is to perform nonlinear transformations of the input. The layers are typically build up of neurons (S.-C. Wang 2003). Only layers built up of neurons will be covered in this thesis.

Neurons

A neuron in an ANN model imitates the function of a nerve cell in the human brain. A layer is usually a group of neurons. Each neuron receives input from the neurons in the immediately preceding layer, and transmit output to the neurons in the immediately following layer (S.-C. Wang 2003). The output from a neuron is a real number, an is sometimes called the *activation*. A neuron may transmit its output to multiple neurons it is connected to.

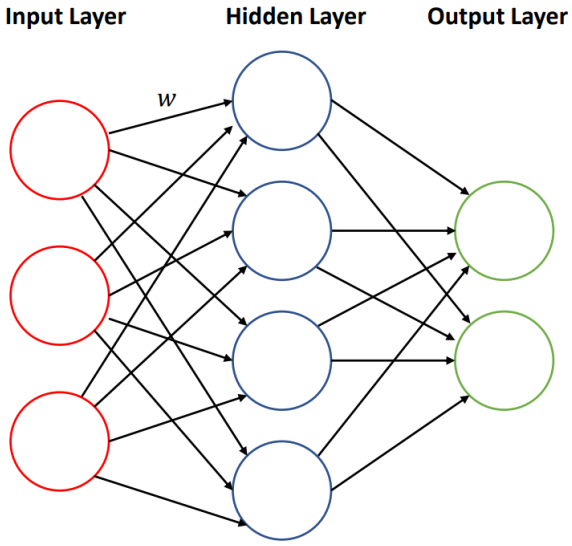


Figure 2.14: Illustration of a neural network with one hidden layer. All the connectors between the nodes has its own weight w. To preserve the clarity of the illustration, only one connector weight is visualized.

Weights

Each neuron may receive the output of multiple neurons, but it may only transmit one single output. An output may be transmitted to multiple neurons through each neurons individual connectors. Each connection to a neuron has a weight, which determines the impact of each neuron on the neuron it transmits its output to. A neuron receives the output of one or multiple neurons, while the input of the neuron is a weighted sum of all the outputs from the connected neurons (Shiruru 2016). The weights are tuned frequently during training. To train a model may be seen as an optimization problem, where the goal is to optimize the weights (Jia Wu *et al.* 2019).

2.5.2 Architecture

An artificial neural network typically has multiple layers. The input layer is the first layer, and the output layer is the last layer. Between them, there may be several layers, hidden layers, as mentioned in Section 2.5.1. There are networks with only one or even no hidden layers (S.-C. Wang 2003). Networks containing few layers will have faster learning speed and less data than a multi layered network. The downside is that a single layered network can only learn to solve linear problems, while a multi-layered perception can learn to solve non-linear problems (Gurney 1997).

After a neuron has received the weighted sum of the activations of all its connected neurons, the weighted sum is passed through an activation function to produce the output of the neuron. The activation function is usually non-linear (S.-C. Wang 2003). A commonly used activation function is the ReLU activation. ReLU, or rectified linear unit, has the function $f(x) = max(0, x)$. For an input less than 0, the ReLU activation function returns 0, setting the output of the node to 0. Positive inputs are passed through. Neurons with an output of 0 s not activated in the network. The ReLU activation function contributes to activating less neurons by excluding every neuron with an activation less than of equal to zero. In that way, only a certain number of neurons are activated. This makes ReLU computationally efficient (Jia Wu *et al.* 2019). The sigmoid function is another activation function worth mentioning. Before the ReLU activation function was introduced, the sigmoid function was seen as the main activation function (Nair and Hinton 2010). Sigmoid and ReLU will be explained in more detail in Section 3.1.4.

There exists a bunch of layers, when it comes to neural networks. Many of the layers consists of neurons that receive input from the immediately preceding layer and transmit its output to the neurons in the immediately following layer, but that is not the case for every type of neural networks. Different layers perform different transformations on their inputs. A fully connected layer, or a dense layer, is a commonly used layer. Every neuron in a dense layer is connected to every neuron in the following and in the preceding layer. In a recurrent layer, neurons may feed their input back to them selves or back to neurons in preceding layers. Recurrent layers are often used to handle time series data (Savarese and Maire 2019). Normalization layers are layers that scale the values in a training set. This is done to ensure that all the different inputs has a somewhat similar data distribution (Ba *et al.* 2016). Convolution layers is the main feature of Convolutional Neural Networks (CNNs). CNN will be described in more detail in Section 3.1, as this thesis is focused on applying CNN to study porous media.

Different input require different amount of neurons in the input layer. Usually, the number of neurons equals the number of features in the input data (S.-C. Wang 2003). The model configuration determines the number of neurons in the output layer. A regressor typically has one neuron in the output layer, since it returns a single number. A classifier may return a probability for each of its class labels. This means that the number of neurons in the output layer equals the number of class labels known by the classifier.

Other important features of neural networks, like error-back propagation, will be introduced in Section 3.1.

2.5.3 Pruning and Augmentation - Techniques to Improve Learning

There are plenty of techniques when it comes to improve the predictive abilities of a NN model. Some techniques are applied before building the model, and some techniques are applied while or even after the model is built. Pruning is a technique that involves removing as many neurons as possible without making a significant impact on the predictive skills of the model (Yan Lecun 1990). The idea is that if all neurons are ranked based on their contribution, some neurons will be superfluous. The least contributing neurons could be removed without the model suffering any distress for that reason. The main goal when applying pruning, is to build a smaller and faster NN, though maybe a bit less accurate (Molchanov *et al.* 2019).

Data augmentation is a technique that is common to apply when the input data is images. The reason for augmenting images used in training is to increase the amount of training data. This is done by, e.g., flipping, rotating, cropping, zooming or changing the colors. It is common to apply image augmentation to decrease the chance for the model to overfit the data (Perez and J. Wang 2017).

2.5.4 Overfitting and Underfitting

Overfitting is a result of poor generalization to data. It can be revealed when the model is able to predict well on the training data, but the general predictive skills of the model is bad. If the model yields good results on the training data, but bad results on the test data, it has probably overfit to the training data (Baiyang Wang and Diego Klabjan 2017). Increasing the amount of training data may be a solution to the problem. As mentioned, data augmentation can be applied for the purpose of creating additional training data. Another technique to reduce overfitting is to apply *dropout*. During training, each neuron in a layer has a probability of being left out. This is done to prevent the neurons from reaching a too high degree of co-adaption (Srivastava *et al.* 2014).

Underfitting may be the explanation if the model is unable to predict the data it was trained on. This may be caused by a too simple network. A solution could be to increase the number of layers or neurons to increase the complexity of the network (H. Zhang *et al.* 2019). If dropout is applied, it may be an idea to reduce the dropout probability. A high dropout probability demands a higher number of neurons, which eventually will slow down the training speed and lead to underfitting (Srivastava *et al.* 2014).

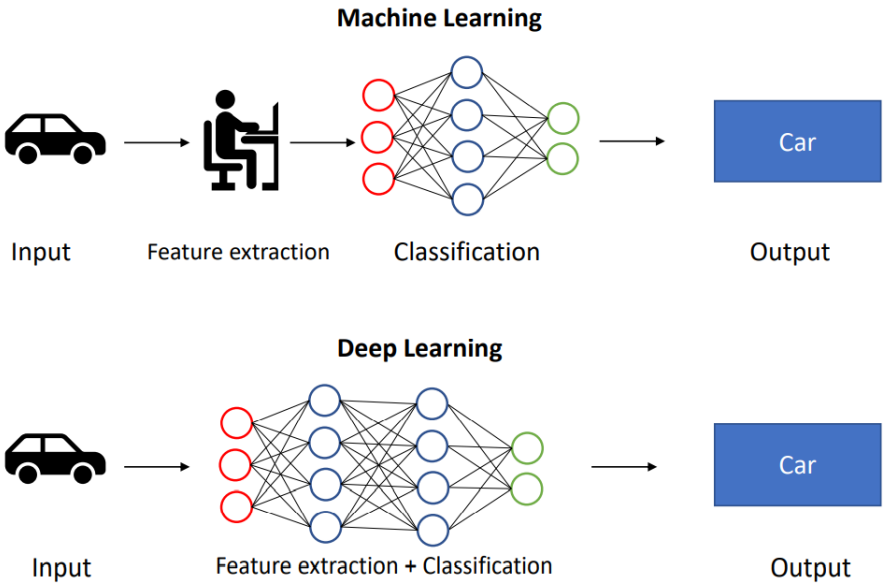


Figure 2.15: Feature extraction is done manually in ML. In DL, the feature extraction is done by the machine. The illustration shows a ML classifier and a DL classifier.

2.6 Deep Learning

Deep learning (DL) is a subset of ML. As ML, it is inspired by the way human and animal brains learn. It involves exposing a multilayered NN to a large amount of data. Chassagnon *et al.* (2020) defines the main difference between classical ML and DL as a difference in feature extraction. In ML the feature selection is done before training the model, while in DL, the features are generated during training. Hidden layers are a crucial part of DL, since they are required for feature extraction from raw data (C. Zhang *et al.* 2019). As mentioned earlier, some ML does not have any hidden layers. CNN is a type of DL that will be explained in Section 3.1.

Recent years, the use of artificial intelligence has escalated in porous media related research. Chawshin *et al.* (2021) used artificial intelligence for lithology classification of whole core CT scans. As such CT scans can provide valuable information about cores extracted from wells. The purpose of the research was to investigate the possibilities for building a CNN model whose purpose is to automatically predict the lithology of a well in the Norwegian continental shelf (Chawshin *et al.* 2021). Very recently Elmorsy *et al.* (2022) developed a model for permeability prediction of digital porous media. It was made using CNN, and it is built in such a way that it elevate the models accuracy beyond that of the existing state-of-the-art 3D CNN models (Elmorsy *et al.* 2022).

Chapter 3

Theory

3.1 Convolutional Neural Networks - CNN

A convolutional neural network is a class of artificial intelligence. It is common to apply CNN for visual analysis, such as image classification. In this thesis, an algorithm for building 2D and 3D CNN models was used.

3.1.1 Architecture

It is common for a CNN to take a 3rd order tensor as input. A tensor is a matrix of order 3 or higher. A common input for a CNN would be an image with M rows, N columns and 3 channels, (R, G, B color channels). Common numbers for the input is 32×32, 64×64, 128×128 and 224×224 (O'Shea and Nash 2015). In this project a single channel was used, since the images used were binary. As explained in 2.5.2, there are many different layers used in an artificial neural network.

An abstract illustration of how a forward pass CNN may be structured is shown with the following equation.

$$\boldsymbol{x}_1 \longrightarrow \boxed{\boldsymbol{w}_1} \longrightarrow \boldsymbol{x}_2 \longrightarrow ... \longrightarrow \boldsymbol{x}_{L-1} \longrightarrow \boxed{\boldsymbol{w}_{L-1}} \longrightarrow \boldsymbol{x}_L \longrightarrow z, \tag{3.1}$$

where $\boldsymbol{x}_1$ is the input tensor, usually an image. The boxed $\boldsymbol{w}_1$ is the first layer. This is where $\boldsymbol{x}_1$ is processed and forwarded to the next layer as $\boldsymbol{x}_2$. This forwarding process goes on until all layers in the CNN has finished, and $\boldsymbol{x}_L$ is returned as an output. During training, the output is evaluated, and the weights in the network are adjusted by a method called back-propagation. The loss layer z calculates the discrepancy between the prediction $\boldsymbol{x}_L$ and the actual value, Val. Usually more complex loss functions are used, but just to give an example, $z = 1/2||Val - \boldsymbol{x}_L||^2$

could be used as a simple loss function. There are many different loss functions for both regressors and classifiers. The back propagation layer $\mathbf{w}_L$ and the loss function is only applied during training. When predicting, the tensor $\mathbf{x}_i$ is processed by the weights $\mathbf{w}_i$, and the final output is $\mathbf{x}_L$ (Jianxin Wu 2017).

3.1.2 Stochastic Gradient Descent

Optimizing the model is to minimize the loss by tuning the weights. A common method for updating the values of the weights is called *Stochastic Gradient Descent*, or SGD for short. First the input $\boldsymbol{x}_1$ is forward passed until reaching the output $\boldsymbol{x}_L$. After achieving the loss z, it is used as a supervision signal to guide the updating of the parameters (Jianxin Wu 2017). The way that SGD updates the parameters is generalized by the equation:

$$\boldsymbol{w}_i^{t+1} = \boldsymbol{w}_i^t - \eta \frac{\partial z}{\partial \boldsymbol{w}_i^t}, \tag{3.2}$$

where the layers are denoted by i, referring to the layer number, and superscript t referring to the epoch number. The learning rate η is usually set to a small number, e.g. $\eta = 0.001$ (O'Shea and Nash 2015). The gradient $\partial z/\partial \boldsymbol{w}_i$ measures the change of z with respect to $\boldsymbol{w}_i$. To decrease the value of z, $\boldsymbol{w}_i$ should be updated along the opposite direction of the gradient. This way of updating is called *gradient descent*. In fig 3.1 the gradient is denoted by g. As can be seen from the figure is that a too big of a change of the weight might lead the weight to pass by the minimum point. Keeping η too high might lead the weights to jump back and forth without reaching the minimum. A wrong update of a weight might lead to an increase of z. That is why the learning rate η usually is very small, so that the weights are updated by a small amount (Jianxin Wu 2017). After using all the training examples in a batch to update parameters, one epoch has been processed. During one epoch, the average loss will be reduced while the model learns, and after the model does not improve for a given *patience* number the training is terminated. If the patience number is 5, the model would have to show no signs of improvement for 5 epochs in a row, no matter how fractional the improvement, to stop the training before it has completed all epochs (Jianxin Wu 2017).

If the gradient descent is used to update the parameters using only one training example, the loss function will be very unstable. Usually the gradient is estimated from a small subset of the

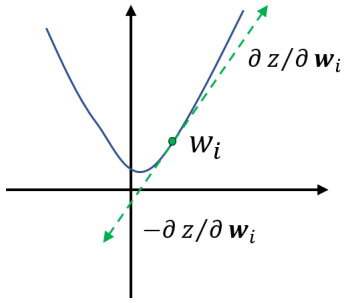


Figure 3.1: Illustration of the gradient descent method, where w_i is optimized based on the gradient $\partial z/\partial \mathbf{w}_i$.

training data, instead of using the whole batch from the training data. The parameters are updated only once in an epoch, and if the training examples are large, this demands a lot of computations. When the gradient used for updating the parameters is estimated from a small *batch* of data, it is called *Stochastic Gradient Descent*. A typical batch size for learning the CNN's parameters is a set of 32 or 64 examples (Jianxin Wu 2017).

3.1.3 Error Back Propagation

For each layer the partial derivative of z with respect to the layer $\boldsymbol{w}_i$ and with respect to the layers input $\boldsymbol{x}_i$ needs to be computed. To do this for the i'th layer, the partial derivatives from the $(i+1)$'th layer needs to be known. This is called the Error Back Propagation, since the parameter update starts at the last layer, and propagates backwards towards the first layer. For the last layer, $i = L$, $\partial z/\partial \boldsymbol{w}_L$ and $\partial z/\partial \boldsymbol{x}_L$ are easy to compute. For the loss function given as an example, $z = 1/2||Val - \boldsymbol{x}_L||^2$, the partial derivative $\partial z/\partial \boldsymbol{x}_L = \boldsymbol{x}_L$ - Val, where Val is the target value or the "true value" (Jianxin Wu 2017).

Further explanation is done with an example: The propagation process for the $(i+1)$'th layer is finished, and the process moves on to the i'th layer. Since layer $(i+1)$ is finished, this means that $\partial z/\partial w_{i+1}$ and $\partial z/\partial x_{i+1}$ are computed and stored in memory. The task now is to compute $\partial z/\partial w_i$ and $\partial z/\partial x_i$. By applying the chain rule, we get

$$\frac{\partial z}{\partial(\text{vec}(\boldsymbol{w}_i)^T)} = \frac{\partial z}{\partial(\text{vec}(\boldsymbol{x}_{i+1})^T)} \frac{\partial \boldsymbol{x}_{i+1}}{\partial(\text{vec}(\boldsymbol{w}_i)^T)} \tag{3.3}$$

$$\frac{\partial z}{\partial(\text{vec}(\boldsymbol{x}_i)^T)} = \frac{\partial z}{\partial(\text{vec}(\boldsymbol{x}_{i+1})^T)} \frac{\partial \boldsymbol{x}_{i+1}}{\partial(\text{vec}(\boldsymbol{x}_i)^T)} \tag{3.4}$$

where vec is a reshaping operation converting the matrix into a column vector, and T is the transpose operator. Notice that $\boldsymbol{x}_i$ is directly related to $\boldsymbol{x}_{i+1}$, and that the function that relates the two has the parameters $\boldsymbol{w}_i$. That is why it is easier to compute $\partial \boldsymbol{x}_{i+1}/\partial(\text{vec}(\boldsymbol{w}_i)^T)$ and $\partial \boldsymbol{x}_{i+1}/\partial(\text{vec}(\boldsymbol{x}_i)^T)$ than to compute $\partial z/\partial(\text{vec}(\boldsymbol{w}_i)^T)$ and $\partial z/\partial(\text{vec}(\boldsymbol{x}_i)^T)$ directly (Jianxin Wu 2017).

3.1.4 ReLU

From here on, the notation is changed. Assume $\boldsymbol{x}^l$ where $\boldsymbol{x}^l \in \mathbb{R}^{H^l \times W^l \times D^l}$. The order 3 tensor $\boldsymbol{x}^l$ is the input for the l'th layer. The triplet index set (i^l, j^l, d^l) helps locating any specific element in the tensor $\boldsymbol{x}^l$, by referring to the element at

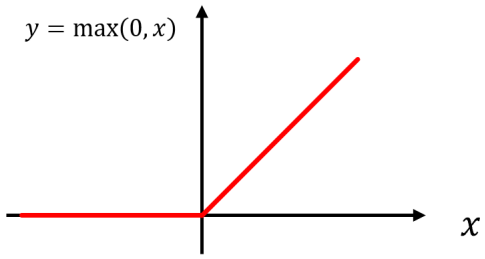


Figure 3.2: Illustration of the ReLU activation function. A positive x-value will be kept as it is, while all negative x-values will be set to zero.

the i'th row, j'th column in the d'th channel. To make future notations simpler, the zero-based indexing convention is used: $0 \leq i^l \leq H^l$, $0 \leq j^l \leq W^l$, $0 \leq d^l \leq D^l$. An assumption is that the output $\boldsymbol{x}^{l+1}$, transformed from the input $\boldsymbol{x}^l$, has the size $H^{l+1} \times W^{l+1} \times D^{l+1}$. Rectified Linear Unit, ReLU, is an elementwise activation function, meaning that it applies to every element in x^l (O'Shea and Nash 2015). There are no parameters inside the ReLU layer, which means that there is no need for parameter learning in a ReLU layer. It does not change the size of the input, hence the size of y equals that of x^l. This means that after applying the ReLu function, which is

$$y_{i,j,d} = \max(0, x^l_{i,j,d}), \tag{3.5}$$

the dimensions are $0 \leq i^l \leq H^l = H^{l+1}$, $0 \leq j^l \leq W^l = W^{l+1}$, $0 \leq d^l \leq D^l = D^{l+1}$ (Jianxin Wu 2017).

Let's say that $\boldsymbol{x}^l_{i,j,d}$ is one of the $H^l W^l D^l$ features extracted by the layers 1 to $l-1$. It can be positive, zero or negative. A positive $\boldsymbol{x}^l_{i,j,d}$ may indicate that the region inside the image contain a certain pattern, for example a platypus beak or other patterns. If that region does not contain any of the specific patterns, $\boldsymbol{x}^l_{i,j,d}$ is negative or zero. In the ReLU layer, all negative input values are set to zero. This means that an image possessing a pattern will be amplified, such that the presence of the pattern is made more apparent. If one feature is activated, and that feature is "beak", the probability for the statement "the input image contains a platypus" is small, unless other features like "beaver-tail", "fur", "webbed foot" are detected. If multiple features are activated after the ReLu layer, the probability increases. As mentioned in 2.5.2, the logistic sigmoid function is a much used activation function:

$$y = \sigma(x) = \frac{1}{1 + \exp(-x)} \tag{3.6}$$

Compared to ReLU, it tends to be less effective than ReLU for CNN learning. The derivative dy/dx can be written as $\frac{dy}{dx} = y(1-y)$, where $0 < y < 1$. It is clear that $\frac{dy}{dx} \leq 1/4$, which reduce the impact of the gradient $\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}$. After several

sigmoid functions during back propagation, the components of the gradient will be close to zero. Gradient based learning like SGD with a low impact gradient is difficult (Jianxin Wu 2017). An important change in CNN is the way ReLU significantly reduces the difficulty in learning parameters. ReLU benefits SGD learning by not changing the activated features, and not activating the features that are not of interest (the gradient is set to zero) (Jianxin Wu 2017).

3.1.5 Convolution Layer

A convolution layer takes in an input that is usually an image and outputs a set of feature maps, or a single map. The feature map is a result of the layer trying to extract different features using convolution kernels. A kernel is a matrix that "overlaps" the input image and calculates the product of the kernel and the image values it overlaps. In a convolution layer, the kernels are the only learnable parameters (O'Shea and Nash 2015). One typically start from the top left, and move to the right by a number of pixels defined by the *stride s*. A stride of *n* means that the kernel moves *n* pixels to the right after each calculation. When reaching the right end, it moves back to the end on the left and moves *n* pixel down. The scalar products will form a new matrix. Usually multiple kernels are used to extract different features. A typical kernel size is therefore 3×3. This is because the size of the output matrix is reduced more compared to the input, the bigger the kernel size. It is crucial to keep the output matrix large enough to capture important features from the input, as well as keeping the input small enough to not demand too much computational power (O'Shea and Nash 2015). As well as keeping the kernel small, it would be beneficial to keep the kernel odd-sized. If the kernel is odd-sized, all matrix elements used in the kernel convolution is symmetrically around the element in the center of the kernel. Without the symmetry, we would have to account for distortions (Sahoo 2018).

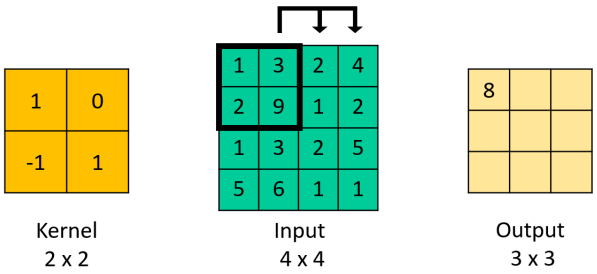


Figure 3.3: A simple illustration of how kernel convolution works. The kernel with a size $2\times$ moves with a stride $s = 1$.

Fig. 3.3 illustrates how kernel convolution works. The figure illustrates a 4×4 input matrix (image) and a 2×2 kernel. After the kernel has moved through the entire input matrix, the output matrix in this case has the size 3×3. If the convolution kernel had the size 3×3, the output would have the size 2×2.

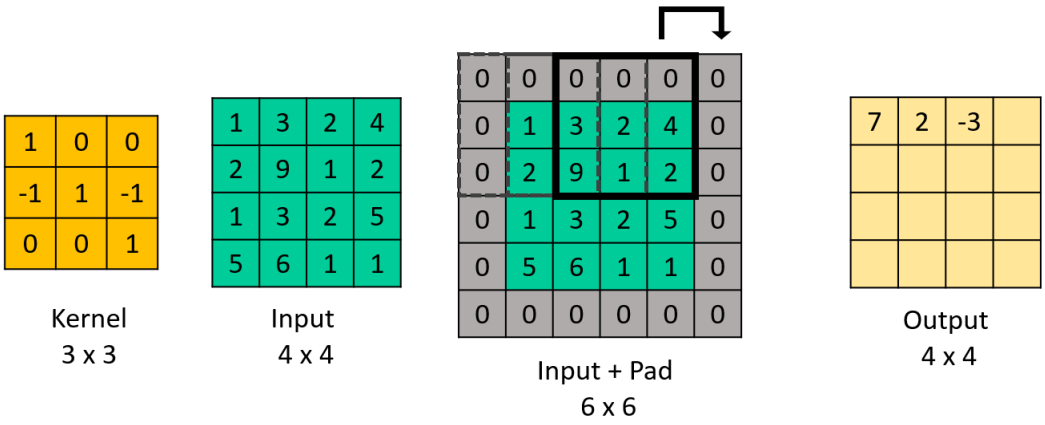


Figure 3.4: Input of the size 4×4 and 3×3 kernel with stride 1. The input is zero-padded, which results in a 4×4 output.

Given an input with the dimension $H \times W \times D$, where the dimension of the l'th layer is $H^l \times W^l \times D^l$, where H^l is the number of rows, W^l is the number of columns and D^l is the number of channels in the layer. A convolution kernel has the dimension $H_k \times W_k \times D^l$, which makes it a 3rd order tensor as well. When the kernel overlap the spatial coordinates $(0,0,0)$, the convolution result at that spatial location in the output will have the value of the sum of the $H_k W_k$ numbers in each of the D^l channels, i.e. the sum of $H_k W_k D^l$ scalar products. Assuming multiple kernels, N_k, of the same dimension, and denoting them as $\boldsymbol{f}$, makes $\boldsymbol{f}$ a 4th order tensor in $\mathbb{R}^{H_k \times W_k \times D^l \times N_k}$. To help defining specific elements in the kernels, the four dimensions have the index variables: $0 \leq i \leq H_k$, $0 \leq j \leq W_k$, $0 \leq d^l \leq D^l$ and $0 \leq n \leq N_k$ (Jianxin Wu 2017).

As mentioned earlier, the size of the output matrix is smaller than the input. This holds for as long as the size of the convolution kernel is bigger than 1×1. A method called *padding* is often used to preserve the dimensions (Jianxin Wu 2017). Padding adds extra rows and columns on each side of the input matrix for every channel of the input. A padding layer element usually has the value zero, but it may be given any value. Fig. 3.4 illustrates how padding works. If the size of the convolution kernel is $H_k \times W_k \times D^l$ and the size of the input matrix is $H^l \times W^l \times D^l$, then the output will have the size $(H^l - H_k + 1) \times (W^l - W_k + 1)$. To produce an output of the same size as the input, $H^l + 2p_H - H + 1 = H^l$, where p_H is the size of the padding layer in the H-direction for both top and bottom of the matrix. It is clear that $p_H = \frac{H_k - 1}{2}$. The same holds for W-direction, where $p_W = \frac{W_k - 1}{2}$ (Jianxin Wu 2017). However, if the stride s is larger than 1, the size of the output matrix is reduced (O'Shea and Nash 2015). In that case we have an output with the size $(\frac{H^l + 2p_H - H_k}{s} + 1) \times (\frac{W^l + 2p_W - W_k}{s} + 1)$. In this case, the padding height and width has to be $p_H = \frac{(H^l - 1)s - H^l + H_k}{2}$ and $p_W = \frac{(W^l - 1)s - W^l + W_k}{2}$, to keep the same dimensions.

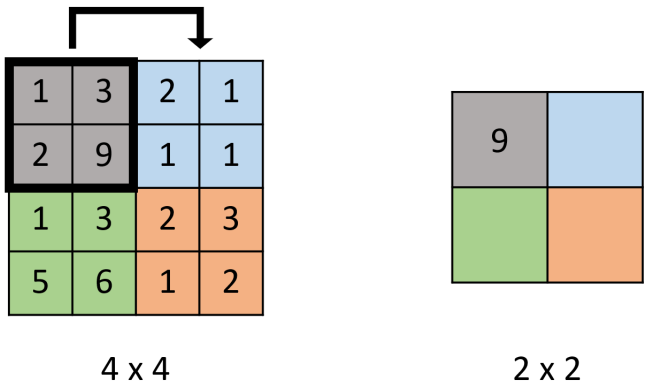


Figure 3.5: Illustration of max-pooling. Input with dimension 4×4 with kernel of 2×2 and stride $s = 2$. Output is becomes a smaller matrix then the input with the maximum values from the regions where the kernel has overlapped the input. In this case the output matrix would be 2×2 with elements $[[9, 2], [6, 3]]$

A beneficial side of CNN compared to other ANNs is the use of *parameter sharing*. Assuming that features that are useful to compute at a given spatial region in the input, are also useful for other spatial regions, there is no need for learning parameters at each spatial location for the same pattern-recognition. By sharing weights across all kernels in the same layer the number of produced parameters in the convolution layer will be massively reduced. For example a kernel that detects horizontal edges, could be useful for the whole image, not just one part of the image (Jianxin Wu 2017).

3.1.6 Pooling Layer

Pooling layers are used for the purpose of reducing the size of the input to reduce the computational complexity and number of parameters, while at the same time maintaining the quality of the representation. In max-pooling, the kernel moves across the matrix and returns the maximum value for each stride. This is illustrated in Fig. 3.5. There are mainly two types of max-pooling. Kernel with size 2×2, and stride of 2, and *overlapping pooling*, where the stride is 2 and the kernel size is 3×3. A larger kernel size will typically lead to decrease in model performance, due to the destructive nature of pooling (O'Shea and Nash 2015). A kernel of the dimensions 2×2 with a stride $s = 2$ will reduce the size of the input by 25 %.

There exists several different kinds of pooling. General pooling layers are build of neurons capable of performing many different common operations. An example is the *average pooling* layer, which unlike the max-pooling layer, does not find the maximum value, but the average. Max-pooling highlights the features, while average pooling smoothens them out (Jianxin Wu 2017).

3.2 Hydraulic conductivity

In the case of sedimentary rocks as porous media, the hydraulic conductivity is a physical property that describes the ability for a rock to transmit fluid through its pore space. To transmit fluid, a hydraulic gradient need to be present. Poiseuille's law, that describes pipe flow, can be expressed as

$$q = \frac{\Delta P}{L}\frac{\pi R^4}{8\mu} = g\Delta P, \tag{3.7}$$

where ΔP is the pressure gradient over the length, L, of the pipe with radius R. Flow rate is q, and the dynamic viscosity of the fluid is μ. Hydraulic conductivity is denoted as g. Qingrong Xiong *et al.* (2016) explained how Poiseuille's law is valid for laminar flow, and is is therefore applicable for describing flow in pores. Poiseuille's law expresses the relation between change in pressure and the flow rate through a cylindrical pipe. A pore throat may be seen as a cylindrical pipe. In this thesis, the hydraulic conductance will be defined as

$$q = \frac{g\Delta P}{L} \tag{3.8}$$

Geoffery Mason and Norman R. Morrow (1990) related the area A to the inscribed radius by $r_{\text{insc}} = 4GA$. The inscribed radius is the radius of a circle that tangent all sides of a triangle, illustrated by Fig. 3.6.

In Eq. 3.9, Øren *et al.* (1998) expresses the hydraulic conductivity as a function of the shape factor, G. It is defined as $G = A/p^2$, where A is the cross section area of the pore element and p is the perimeter length (Bakke and P.-E. Øren 1997). For a slit shaped pore, the shape factor is zero, while for equilateral triangle-shaped pore, the shape factor is 0.0481. Network elements, pore throats or bodies, may be approximated as shapes with circular, rectangular or triangular cross-sections. In order to calculate the hydraulic conductivity, when assuming most network elements have triangular cross-sections, Øren *et al.* (1998) suggests the functional dependency between g and G is close to linear and sufficiently approximated by

$$g = \frac{3}{5}\frac{GA^2}{\mu} \tag{3.9}$$

where A is the cross section area of a network element.

By combining Geoffery Mason and Norman R. Morrow (1990) definition of r_{insc} and Eq. 3.9, hydraulic conductivity can be expressed as

$$g = \frac{3}{80}\frac{r_{\text{insc}}^4}{\mu G} \tag{3.10}$$

Equation 3.10 was used to calculate the hydraulic conductivity of the network

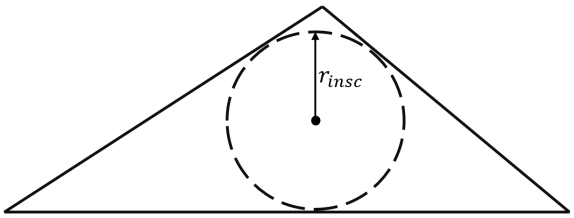


Figure 3.6: Illustration of the inscribed radius of a triangular pore throat.

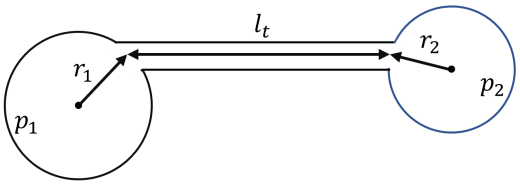


Figure 3.7: Two pore bodies, p_1 and p_2 with the corresponding radii r_1 and r_2, and a pore throat with the length l_t. The radii r_1 and r_2 corresponds to l_1 and l_2.

elements in this project. Two pore bodies p_1 and p_2 are connected by a pore throat, and each of the three elements are likely to have different hydraulic conductances due to their different geometries and sizes. To calculate the *effective* hydraulic conductance between p_1 and p_2, a length-weighted harmonic average of the three elements is said to be an representative average. The length-weighted effective hydraulic conductance can be expressed as (Ramstad *et al.* 2019)

$$g_{1,2} = l_{1,2}\left(\frac{l_1}{g_1} + \frac{l_t}{g_t} + \frac{l_2}{g_2}\right)^{-1}, \tag{3.11}$$

where the network elements p_1, p_2 and the pore throat are denoted by 1, 2 and t, respectively. The effective length $l_{1,2} = l_1 + l_t + l_2$, and the effective hydraulic conductivity is $g_{1,2}$.

The CNN models will be trained using images of equal dimension. A consequence of this is that the actual sizes of the features in the images is unknown for the machine. Hence, the hydraulic conductances need to be scaled by a factor derived from the spatial extent of the pore body-throat-body system. A more detailed explanation on how this is done is found in Section 5.5.1.

3.3 Navier-Stokes

The Navier-Stokes equations are some well known partial differential equations used to describe fluid motion. Stokes equations are linearizations of the Navier-Stokes equations. These equations will be the basis when scaling the hydraulic

conductances.

3.3.1 Stokes

The Navier-Stokes equation is a non-linear, partial differential equation that describe the flow of viscous fluids.

$$\rho\left(\frac{\partial\vec{u}}{\partial t}+\vec{u}\cdot\nabla\vec{u}\right)=-\nabla P+\nabla\cdot\mathbb{T}+\vec{f} \tag{3.12}$$

Here, ρ is the fluid density, $\vec{u}$ is the fluid velocity vector and ∇P is the pressure gradient. $\mathbb{T}$ is the stress tensor, which for a newtonian fluid is $\mathbb{T}=\mu(\nabla\vec{u})$. By assuming that there will only be newtonian fluids in the porous medium, the term $\nabla\cdot\mathbb{T}$ may be replaced by $\mu\nabla^2\vec{u}$. The last term, $\vec{f}$, is an applied body force, for example gravity, $\vec{f}=\rho\vec{g}$. As the differences in gravitational force in the rock samples we are considering are negligible, we can assume $\rho\vec{g}=0$. The left side of the equation is $\rho\left(\frac{\partial\vec{u}}{\partial t}+\vec{u}\cdot\nabla\vec{u}\right)$. This gives $\rho\frac{D\vec{u}}{Dt}$, where $\frac{D\vec{u}}{Dt}$ is the material derivative of the velocity vector.

$$\rho\frac{D\vec{u}}{Dt}=-\nabla P+\mu\nabla^2\vec{u} \tag{3.13}$$

If inertial effects are negligible, then $\rho\frac{D\vec{u}}{Dt}=0$. This means that the fluid is assumed to be incompressible. The result is that the Navier-Stoke's equation becomes the Stoke's equation

$$\mu\nabla^2\vec{u}-\nabla P=0 \tag{3.14}$$

Eq. 3.14 may be written out as

$$\frac{\partial P}{\partial x}=\mu\left(\frac{\partial^2u}{\partial x^2}+\frac{\partial^2u}{\partial y^2}+\frac{\partial^2u}{\partial z^2}\right)$$

$$\frac{\partial P}{\partial y}=\mu\left(\frac{\partial^2v}{\partial x^2}+\frac{\partial^2v}{\partial y^2}+\frac{\partial^2v}{\partial z^2}\right)$$

$$\frac{\partial P}{\partial z}=\mu\left(\frac{\partial^2w}{\partial x^2}+\frac{\partial^2w}{\partial y^2}+\frac{\partial^2w}{\partial z^2}\right)$$

which gives the flow in the three spatial dimensions based on the pressure changes in the same directions. An explanation of how this is used for scaling the images is found in Section 5.5.1

Chapter 4

Materials

4.1 Grid Model

In this project a grid model created from a cuboid subvolume of a rock sample is used. The grid model is generated from a 3D CT-scan of a rock sample. The cuboid subvolume that is used in this project has the size 4.81 mm × 4.64 mm × 7.17 mm. As a grid, the subvolume has the dimensions 792×764×1180 voxles, where each voxel has a cell size of $6.0772 \cdot 10^{-6}$ m. It is stored as a `.raw`-file, and has the name *grid_MM_792×764×1180.raw*. Pore space, grains and clay is represented by the values 0, 9 and 1, respectively. Clay is a phase with unresolved micro-porosity. For simplicity, it is treated as grain in this project. In the images used for training the model, both grain and clay is changed to have the value 1, to make the images binary instead of gray scale. A cross section image of the grid is shown in Fig. 4.1.

4.2 Link & Node - files

Four different network data files were used. The files are ASCII files used to describe the pore network in SI-units. The format of the files are the same as those used by Imperial College London, sometimes referred to as the Statoil format. Two of the four files holds information about the pore bodies in the grid, while the other two holds information about the pore throats. Pores are stored in the node-files, and throats are stored in the link-files. The files and the information they hold will now be presented.
The file names are:

- `node1.dat`

- `node2.dat`
- `link1.dat`
- `link2.dat`

There are a total of 177753 nodes and 317813 links in the network. Total number of pores will be referred to as P, and total number of throats will be referred to as T. The file `node1.dat` stores positional information specific to each node. This information is stored in $P+1$ lines, where the first line contains the total number of pores, P, total length in the directions x, y and z. Each of the ensuing P lines contain the following properties: A Pore index between 1 and P, x-, y- and z-coordinates for the center of the pore and the number of throats connected to the pore. If the pore is connected to C pores, then the ensuing numbers will be the indices of the connected pores. Entry $C+1$ is 1 if the pore is connected to the inlet, 0 if not. Entry $C+2$ is 1 if the pore is connected to the outlet, 0 if not. Then follows C entries that are the indices of the connected throats. Geometric and volumetric properties are stored in `node2.dat`. It contains P lines. Each line contains: The index of the pore (1 to P), the pore volume, inscribed radius of the pore, shape factor of the pore and the micro-porosity volume in the pore.

Each throat is connected to two pores only. These two pores will be referred to as Pore A and Pore B. The first line of the $T+1$ lines in the `link1.dat`-file contains the number T. The T following lines is structured in this way: First the throat index (1 to T), followed by the index of Pore A and Pore B, the inscribed radius and the shape factor of the throat and the length of the link, from center of Pore A to center of Pore B. In `link2.dat`, there is T lines which starts with the index of the throat, Pore A and Pore B. Then follows the length of Pore A, Pore B and the throat, throat volume and micro-porosity volume in the throat. Some of the scripts developed for this thesis use information from the `.dat`-files. Those scripts can only handle files organized in this specific way.

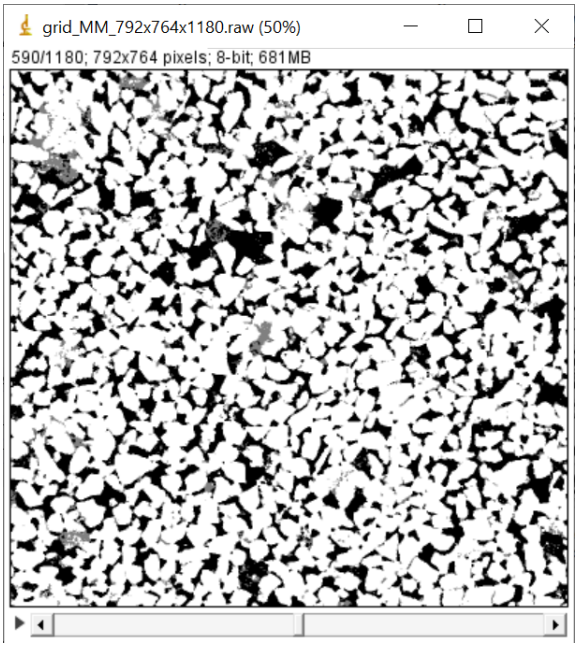


Figure 4.1: Visualization of the grid-file. The cross section shows the 792 × 764 voxels at the z-value 590.

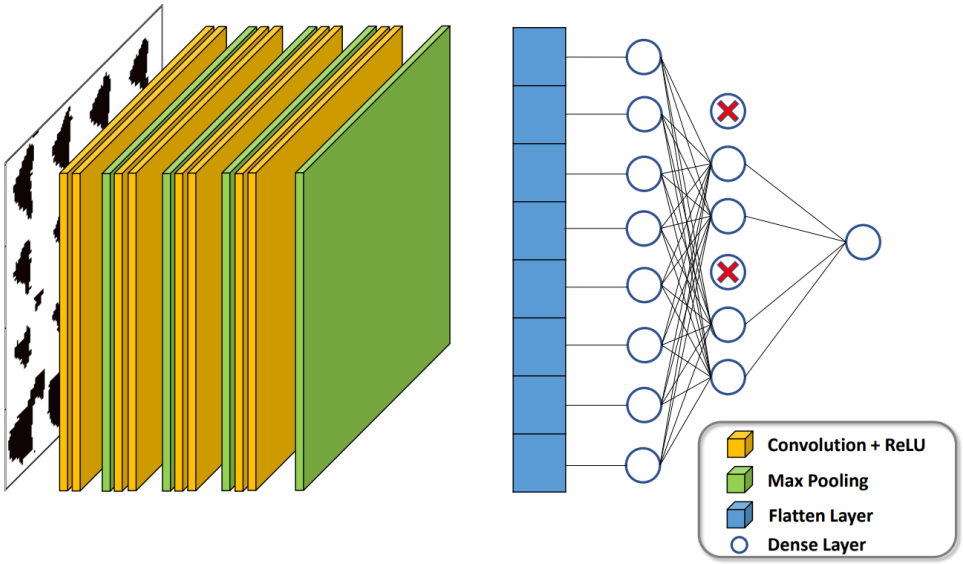


Figure 4.2: Illustration of the architecture used for the 2D CNN model.

4.3 CNN-codes

Two different CNN-modelling codes were used and slightly modified in this project. Both codes were made by Kurdistan Chawshin as a part of her PhD (Chawshin *et al.* 2022). One of the codes is made for estimation of porosity from 2D CT-scan images. Since it is used for numeric estimation, it builds a regressor. The second code is used for facies classification of 3D images. Since it it is used for classification, the code builds a classifier. For the 3D code to be able to return a predicted value for the hydraulic conductivity, the code needed to be changed from a classifier to a regressor. Both codes are written in Jupyter Lab, and is therefore a .ipnyb-file. The codes that was used for building the models will be presented now.

4.3.1 2D CNN Architecture

The model is a sequential regressor, which means that the architecture is arranged layer by layer and that the output layer only consists of a single neuron. The input layer of this 2D model is a convolutional layer. This convolutional layer, as well as the rest of the convolutional layers mentioned in this architecture description, has a semi-random number of output units. The number of output units form each convolutional layer is randomly chosen to be a number between 16 to 256 with a step size of 32. This means that the number of output units are randomly picked between [16, 48, 80, 112, ..., 224, 256]. Each of the convolutional layers has a ker-

nel size of either 3 or 5, also randomly selected. The input is given a zero-padding to make the output the same size as the input. Following the first convolutional layer comes a ReLU and a max pooling layer. The max pooling layer has a pool size of 2, and it also ads a zero-padding. After these first layers, the model builds three more convolutional layers followed by a ReLU and a max pooling layer.

The dense layer needs the input to be one dimensional. Therefore the output from the last max pooling layer is flattened, which means it is turned in to a single array. This array is transmitted to a dense layer with dropout. In this layer, each neuron has a dropout probability of either 0, 0.2, 0.4 or 0.6. the dropout probability is chosen by random. The number of hidden neurons in the dense layer are also chosen by random. It is a number of 32 to 256 with a step size of 32. Then follows the output layer, a dense layer with one single neuron. Both dense layers uses a linear activation function.

The model use the mean absolute error (MAE) as loss function, and optimizing is done using Adam with a learning rate of $1e-2$, $1e-3$ or $1e-4$.

Hyperparameter tuning is done for 30 epochs with a batch size of 32. Early stopping is applied with a patience of 5 epochs, so that the machine avoids spending much time and computational power on tuning hyperparameters that stops improving.

When tuning is done, the best model is trained for 300 epochs with a batch size of 32. Early stopping is again applied, but this time the patience is set to 30. This is done for both the 3D and the 2D models.

4.3.2 3D CNN Architecture

This model is a sequential regressor as well. And just like the 2D model, the input layer is a convolutional layer with a number of input units between 16 and 265 and with a kernel size of 3 or 5. The difference is that this models input layer is a 3D convolutional layer. Following the convolutional layer comes a ReLU and a 3D max pooling layer with size 2. This layer also ads a zero-padding to the input, to give the output the same size as the input. Then the model builds three more convolutional layers followed by a ReLU and max pooling layer built in the same way as the input layer. Then the output is flattened and passed to a dense layer with a minimum of 32 and a maximum of 256 hidden neurons and a linear activation function. The output is activated by a ReLU layer and passed to the output layer, which is a single neuron. The loss function is MAE, while Adam is used as an optimizer with the same learning rates as for the 2D model.

Chapter 5

Methodology

As mentioned, this thesis's primary goal is to build a CNN model that can predict the hydraulic conductivity of a pore body-throat-body system based on images of the pore throat and -bodies. The first step is to generate images for all the links in the network. The predictions will eventually be used for the estimation of the permeability of the network model.

The property of interest when training the CNN-model, is the hydraulic conductivity between pairs of interconnected pore bodies, hereby referred to as nodes. The overall goal is to train a model to predict the hydraulic conductivity between two nodes. To do this we need to generate images of the subvolume between the nodes and links, from which the ML model will be able to extract structural information defining the hydraulic conductance. The following method was used to generate this data.

5.1 Two Methods for Generating Input Data - Cubic and Cylindrical

Points in the xyz-space need to be generated from the xyz-coordinates provided in the `node1.dat`-files. These points will be transformed to grid-coordinates to extract the grid values at the location given points. These grid values will be structured and stored as images. The CNN model demands images of the same dimension. Therefore, the method for generating points should be able to generate the same amount of points as well as number of images for all node-pairs, despite differences in distance between them. Two methods are going to be explained. There are a lot of similarities between the methods, and the first steps are equal. Therefore, the first steps will be explained before digging deeper in to the different methods.

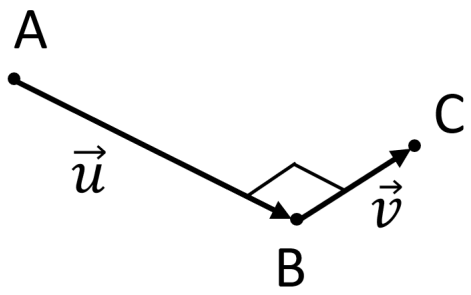


Figure 5.1: Illustration of what will be the starting point of each of the three methods. A and B are nodes, while C is a point at the distance s from B. Vector $\vec{u}$ and $\vec{v}$ are perpendicular.

5.1.1 Orthogonal Vectors

Assume two pore bodies P_1 and P_2 connected by a pore throat. Let the coordinates of the center of the pore bodies be $\vec{P}_a = (x_a, y_a, z_a)$ and $\vec{P}_b = (x_b, y_b, z_b)$. Let $\vec{u} = \vec{P}_b - \vec{P}_a$ be the vector defined by the two pore centers. The following is a general example on how to find the vectors.

To generate our images, we need to find points $\vec{q} = (x, y, z)$ such that $\vec{u} \perp \vec{v}$, when $\vec{u} = \vec{P}_b - \vec{P}_a$ and $\vec{v} = \vec{q} - \vec{P}_b$ and $||\vec{v}|| = s||\vec{u}||$ for a given scaling factor $s \in [0, 1]$.

The two vectors are perpendicular to each other if

$$\vec{u} \cdot \vec{v} = |\vec{u}||\vec{v}| cos\theta = 0. \tag{5.1}$$

The first assumption gives us the following relation

$$u_x v_x + u_y v_y + u_z v_z = 0,$$

$$(x_b - x_a)(x - x_b) + (y_b - y_a)(y - y_b) + (z_b - z_a)(z - z_b) = 0. \tag{5.2}$$

The second assumptions gives

$$s\sqrt{(x_b - x_a)^2 + (y_b - y_a)^2 + (z_b - z_a)^2} - \sqrt{(x - x_b)^2 + (y - y_b)^2 + (z - z_b)^2} = 0, \tag{5.3}$$

where x, y and z are the coordinates of the point $\vec{q}$. The solution of these two equations with three unknowns gives a circle, as illustrated in Fig. 5.2. We now have two equations and three unknowns. The process of getting rid of one of the unknowns will be explained by another example.

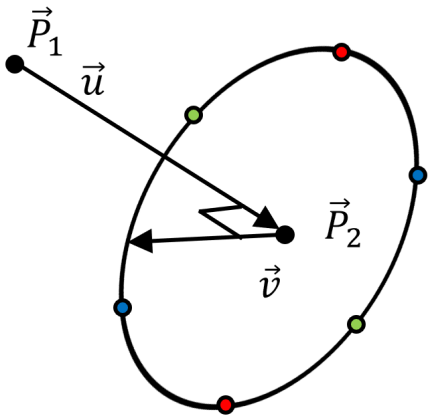


Figure 5.2: Illustration of the possible solutions for the two equations 5.2 and 5.3. The red points share x-coordinate, blue points share y-coordinate and green points share z-coordinate.

Cosider two points in $\mathbb{R}^3$, $\vec{P}_a = (x_a, y_a, z_a)$ and $\vec{P}_b = (x_b, y_b, z_b)$ where $x_a \neq x_b$, $y_a \neq y_b$, $z_a \neq z_b$ and $\vec{u}$ is the vector $\vec{P}_b - \vec{P}_a$. There exists 6 points equidistant to $\vec{P}_b$ and orthogonal to $\vec{u}$ that shares one spatial coordinate with the point $\vec{P}_b$. Two of these points can collapse, giving only 4 points. If we want to calculate the coordinate of one of the six points, say $\vec{q} = (x_c, y_c, z_c)$, this means that for our two equations 5.2 and 5.3 we have two possible solutions for each of the coordinates x_c, y_c and z_c. By for example choosing $z_c = z_b$, the number of possible solutions for $\vec{q}$ is cut down from six to two. The chosen coordinate is in a way chosen to be left out of the equations. If two of the coordinates (x_a, y_a, z_a) equal those of (x_b, y_b, z_b), let us say for example $x_a = x_b$ and $y_a = y_b$, the coordinate $z_c \neq z_b$. Here is why:

Lets say $\vec{P}_a = (x_a, y_a, z_a)$ and $\vec{P}_b = (x_b = x_a, y_b = y_a, z_b)$, where $\vec{u} = \vec{P}_b - \vec{P}_a$. We want to calculate the coordinates of a point $q = (x_c, y_c, z_c)$ such that $\vec{u} \perp \vec{v}$, when $\vec{v} = \vec{q} - \vec{P}_b$ by using the equations 5.2 and 5.3.

From 5.2 we get

$$(x_b - x_a)(x_c - x_b) + (y_b - y_a)(y_c - y_b) + (z_b - z_a)(z_c - z_b) = 0,$$
$$(z_b - z_a)(z_c - z_b) = 0.$$

If we in this case chose $z_c = z_b$, Eq. 5.2 gives $0 = 0$, and we are left with solving for x_c and y_c using Eq. 5.3.

$$s||\vec{u}|| - \sqrt{(x_c - x_b)^2 + (y_c - y_b)^2} = 0,$$

By in this case choosing $x_c = x_b$, we would get

$$(z_b - z_a)(z_c - z_b) = 0,$$

and

$$s||\vec{u}|| - \sqrt{(y_c - y_b)^2 + (z_c - z_b)^2} = 0,$$

which is solvable. This tells us that if the solution circle illustrated in Fig. 5.2 is in the z-plane, $z_c \neq z_b$. If the circle is in the x- or y-plane, $x_c \neq x_b$ and $y_c \neq y_b$.

If we transfer this to our example, one can predetermine either of the spatial coordinates of $\vec{q}$ by putting it equal to the coordinate of $\vec{P}_b$. It seems natural to start with, e.g., $z_q = z_2$. If the solution circle is in the z-plane, $y_q = y_2$. By doing this, the number of possible solutions is reduced from 6 to 2. And since we now know one of the coordinates of the two points, this leaves us with two equations 5.2, 5.3 and two unknowns. Solving these equations will give two solutions. Both solutions are equally applicable for calculating $\vec{v}$. Once the coordinates of $\vec{q}$ is known, the vector $\vec{v}$ can be calculated.

The vector $\vec{u}$ is scaled down using a scaling factor $\gamma = \vec{u}/n_c$, where n_c is the number of cross sections. Then point $\vec{P}_b = \vec{P}_a - i\gamma$ for $i \in [-n_c, 0]$. The n_c equals the number of images created between two nodes.

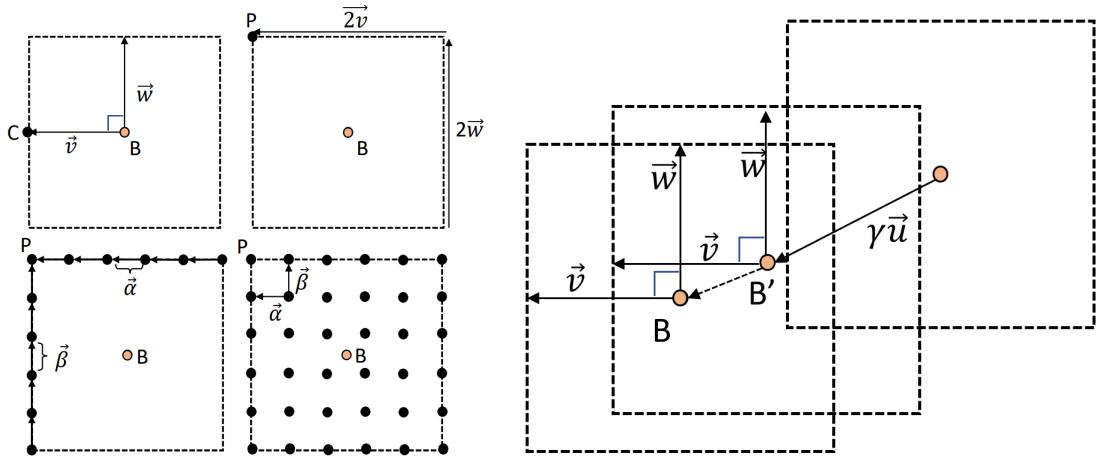
Cubic Extraction

This method is used to generate points in a cubic-shape, with square cross sections. This method is used to generate squares with any number of points, with the consequence of not having the node it self as a point for even numbered cross section-sides. As explained in Section 3.1.1, it is common to train CNN-models on images with a dimension that are recursive divisible by 2, such as 32×32, 64×64, etc.

The number of points and number of cross sections should therefore be large enough to give an adequate description of the area between the nodes. The procedure will continue where Section 5.1.1 left off.

The vector $\vec{w} = \frac{\vec{u} \times \vec{v}}{||\vec{u} \times \vec{v}||}||\vec{v}||$ is a vector perpendicular to the plane spanned by $\vec{u}$ and $\vec{v}$, and of length equal to $\vec{v}$. The first corner point of the cross section is found by adding together $\vec{v}$, $\vec{w}$ and the coordinated of the cross section center point, as illustrated in 5.3a. This corner point, $q_c = \vec{w} + \vec{v} + \vec{P}_b$, will be the starting point for the calculation of the rest of the points. A vector $\vec{\alpha}$ is defined as $2\vec{v}/n_p$ and $\vec{\beta}$ is defined as $2\vec{w}/n_p$, where n_p is the number of points equal to the square root of the number of points in each cross section.

Each point is calculated by the formula $p_{i,j} = p_n - i\vec{\alpha} - j\vec{\beta}$, where $i = [0, n_p - 1]$ and $j = [0, n_p - 1]$. After scaling down $\vec{u}$, as described previously, a new point p_n is calculated using the same $\vec{v}$ and $\vec{w}$, but with a new point $\vec{P}_2$. At the end a



(a) Vector $\vec{v}$ and $\vec{w}$ is added the coordinates of B to find first corner point, here named P. Then $\vec{v}$ and $\vec{w}$ is multiplied by 2 and divided by the number of points for each side to give the vectors $al\vec{p}ha$ and $\vec{\beta}$.

(b) Illustrating down scaling of vector $\vec{u}$ to find the new center point B'.

Figure 5.3: Procedure for the method explained in Section 5.1.1.

$n_p \times n_p \times n_c$ sized cube of points is generated. The next step is to extract the grid values of these points.

Cylindrical Extraction

There is no reason for limiting this method to generate cubes only. Actually, generating cylinders allows for more image augmentation, by yielding an increased amount of rotation-possibilities. This method is used to generate coaxial circular cross sections along the vector between two connected nodes. Each cross section consists of multiple concentric circles of points. All the circles in each cross section have different radii. In this method, the vector $\vec{v}$ is rotated around $\vec{u}$, generating a new point for

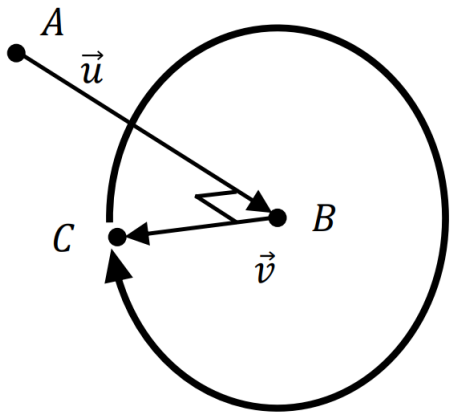


Figure 5.4: The vector $\vec{v}$ is rotated around the axis-vector $\vec{u}$. The rotation starts in the point C.

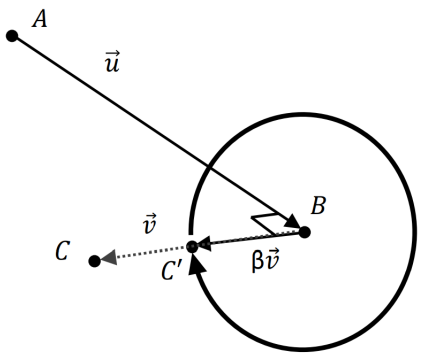


Figure 5.5: After one full rotation, $\vec{v}$ is scaled down using a scalar β. The scaled vector is then rotated around the same axis as $\vec{v}$.

each rotation by a given angle. After a full circle, $\vec{v}$ is scaled down, and rotated again. This is repeated for a defined number of layers n_l and a given number of cross sections n_c.

The transformation matrix for rotating an angle θ about the axis defined by the unit vector (l, m, n) is

$$\begin{bmatrix} ll(1-\cos\theta)+\cos\theta & ml(1-\cos\theta)-n\sin\theta & nl(1-\cos\theta)+m\sin\theta \\ lm(1-\cos\theta)+n\sin\theta & mm(1-\cos\theta)+\cos\theta & nm(1-\cos\theta)-l\sin\theta \\ ln(1-\cos\theta)-m\sin\theta & mn(1-\cos\theta)+l\sin\theta & nn(1-\cos\theta)+\cos\theta \end{bmatrix}. \tag{5.4}$$

In our case we use the unit vector gven by $l = \frac{u_x}{||\vec{u}||}$, $m = \frac{u_y}{||\vec{u}||}$, $n = \frac{u_z}{||\vec{u}||}$, and $\theta = 2\pi/n_p$. The vector $\vec{u}$ is the axis vector for all rotations of $\vec{v}$, hence the unit vector is the same for all rotations. From here, the transformation matrix with our unit vector for an angle θ will be referred to as $\vec{r}_{rot}(\theta)$.

A point P_θ at an angle θ from $\vec{v}$ is found by $P_\theta = B + \vec{v} \cdot \vec{r}_{rot}(\theta)$.

After completing a full rotation, the vector $\vec{v}$ is scaled down using a scaling factor $\beta = ||\vec{v}||/(n_l)$. A new number of points is chosen, and the new vector $\vec{v} = (\vec{v} - i\beta)$ is rotated until reaching full circle to form a new layer of points. The variable $i \in [0, n_l]$, starting as 0 for the outer layer, and reaching n_l for the inner layer.

When one cross section of points is finished, $\vec{u}$ is scaled down as described earlier. Thereafter, the process for finding an orthogonal point q is much simpler. To find the coordinates of the point q, we have the relation $\vec{q} - \vec{P}_a = \alpha\vec{u} + \vec{v}$. A quick rewrite gives us that the coordinates of $\vec{q}$ is found by calculating

$$\vec{q} = \alpha\vec{u} + \vec{v} + \vec{P}_a. \tag{5.5}$$

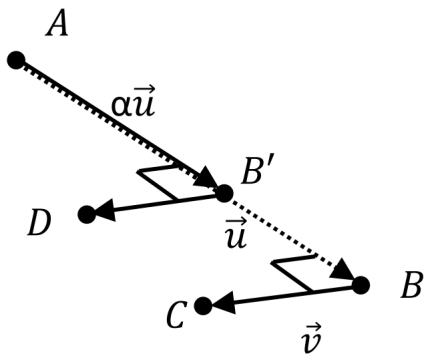


Figure 5.6: The vector $\vec{u}$ s scaled down using a scalar α.

After calculating the new point $\vec{q}$ and rotating $\vec{v}$ to complete a new cross section, the process is repeated for as many times as determined by n_c.

5.2 Grid

Consider a three-dimensional cuboid grid with X voxels in the x-direction, Y voxels in the y-direction and Z voxles in the z-direction, where $X, Y, Z \geq 0$.

Grid values are extracted by transforming the cartesian x, y, and z coordinates of a point into grid coordinates. Each voxel in the grid has a value, depending on the structure it represents. A voxel representing pore space holds the value 0, matrix holds the value 9, and clay, a phase with unresolved micro-porosity, has the value 1. Some of the generated points may be outside of the grid dimension for nodes close to the grid boundaries.

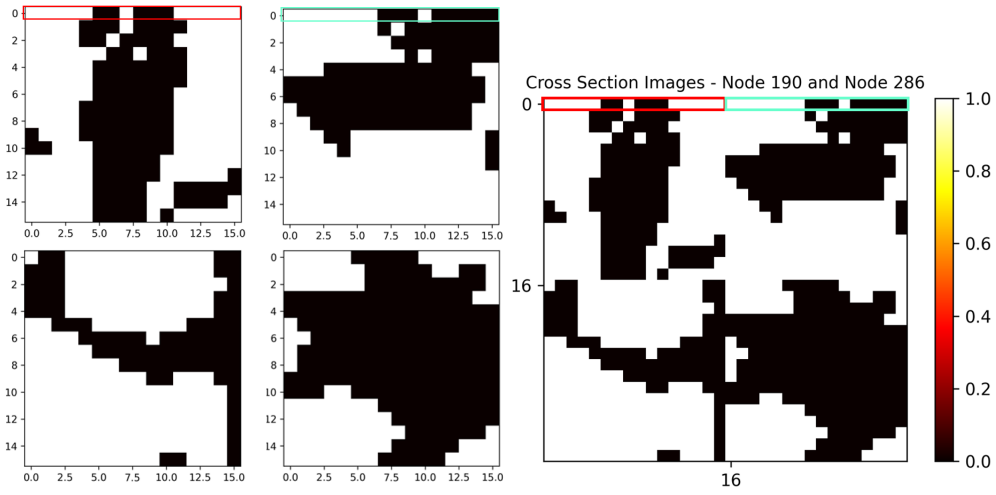
Therefore the points are compared to the grid dimensions before finding the grid value. If a point is outside of the grid, the link is not included so that no image will be created from the node connection. A point that is within the grid boundaries is stored. Then the value corresponding to the stored positional point is extracted and stored. If the grid value is 9, it is changed to 1 before storing. The reason for changing the value of clay is to only have pores and solid matrix. This is done to simplify the problem. In addition to images outside the boundaries, inlet and outlet nodes are excluded as well.

5.3 Point Structuring

To train a model on the points, they must be structured in such a way that the model is able to extract some sort of pattern that relates the images to the corresponding hydraulic conductance. Organizing the points from the cubic extractions was more or less straight forward. Different methods were used for the 2D and the 3D CNN. Organizing the points from the cylindric extraction was more of a challenge. The points corresponds to a circular image, without a way to organize the points as a square that is obviously better than another.

For the 2D method, all the cross sections were put together to form one image, as illustrated in Fig. 5.7. This is a simple way to structure the images, and possibly also understandable for a CNN model. The result is a $\sqrt{n_c} \times \sqrt{n_c}$ image, where the first line of each of the first $\sqrt{n_c}$ cross sections are put together to form the first line in the compound image. Second line in the compound image is the second line of the same $\sqrt{n_c}$ cross sections. This continues until reaching the end of the firs cross sections. Then follows the first line of the second $\sqrt{n_c}$ cross section, etc.

Different methods of point-structure was used depending on if the images were cylindrical or cubic, and depending on whether the images was going to be used for a 2D or 3D CNN-model.



(a) Images from a cubic extraction with 4 cross sections. Each cross section has sides of 16 pixels.

(b) Image of all cross sections united. The first line of the first cross section followed by the first line of the second cross section, two make the first line of the final image.

Figure 5.7: Visualization of how the points are structured for training of the 2D CNN model.

5.3.1 Point Structuring for 2D-CNN

For 2D CNN the points are stored as an array of 0 and 1, organized in the same way as illustrated by the Figure 5.8.

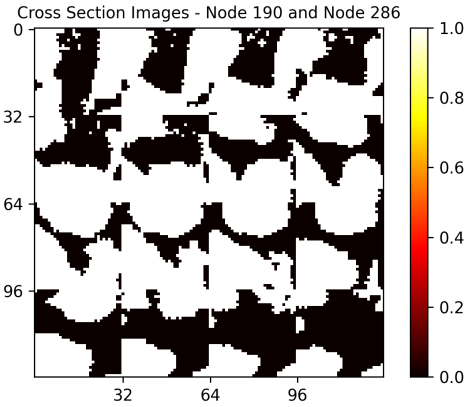


Figure 5.8: Example of an image-representation of how the grid values are organized for 2D CNN training. Top left image is the area around Node 190, and bottom left is the area around Node 286.

The image is a visualization of the cross sections generated between the two nodes. In this image, $n_c = 16$ and each cross section contains 32×32 points. The cross sections are organized from left to right with the area around Node 286 located between 0 and 32 on the horizontal and vertical axis, and Node 190 is located between 96 and 128.

It is important that the number of cross sections is a square number, to obtain the square shape of the points. The size of the total image is not arbitrary. According to O'Shea and Nash (2015), images with the dimension of 32×32, 64×64, 128×128 or 224×224 are common.

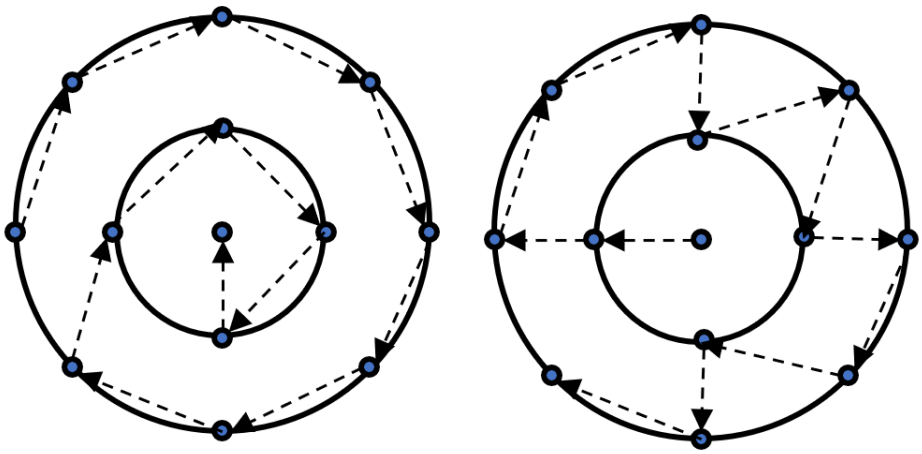
5.3.2 Point Structuring for 3D-CNN

For 3D CNN, the points need to be stored as 3D images. In this case that means that the points are stored as an array of arrays, where each cross section is stored in separate arrays. Let us say the data set contains 500 links, i.e. 500 node-connections, and the parameters are the same as the 2D example, 16 cross sections of 32×32 points, the shape of the array would be (500, 16, 1024).

5.3.3 Cylinder-points Structures

The input of the CNN-model is usually an image with the dimension $n \times n$, though it could be possible that the image is $n \times m$.

A challenge is to organize the circular cross sections in such a way that they make sense as a square or rectangular image. This doesn't necessarily mean that they have to make sense for us human beings, but there need to be a relationship between the patterns in the images and the corresponding hydraulic conductance



(a) The points are organized from outer layer to inner layer.

(b) Organizing the points based on the nearest point, starting in the center point. The four points in the inner layer is at equal distance to the center, so which point is chosen as the second point is arbitrary.

Figure 5.9: "Follow-the-dots"-visualization of the two methods used to organize the points in a cross section from the cylindric-method.

to make sense for the machine. Two different ways of structuring the points are suggested:

Outer-to-Inner

The outer-to-inner-method starts out with the first orthogonal point that was generated, that ended Section 5.1.1, and adds the following points in a clockwise order. When all the points in a layer is stored, the next layer is stored in the same way, as illustrated in Fig. 5.9a. The last point from a cross section to be stored is the center point.

Next-Nearest-Point

A second structuring possibility is to start in with the center point, and always let the next point be the nearest to the current point. This is illustrated in Fig. 5.9b. Which point is chosen as the second point is arbitrary. All the points in the inner layer is equidistant from the center point. A new array to store the points in a structured order, $\vec{aP}_o$ are created and points are transferred from an array containing all the points of the cross section, $\vec{aP}$, to the new array. During struc-

turing, all unstructured points are stored in the array $\vec{aP}$. To find the point $\vec{P}_i$ in $\vec{aP}$ closest to the current point $\vec{P}_{current}$, the following equation is used:

$$\vec{d}_i = (\vec{P}_{i,x} - x)^2 + (\vec{P}_{i,y} - y)^2 + (\vec{P}_{i,z} - z)^2. \tag{5.6}$$

When iterating through $\vec{aP}$, Eq. 5.6 is applied to each point $\vec{P}_i$. The distance, d_i, from the point $\vec{P}_i = (\vec{P}_{i,x}, \vec{P}_{i,y}, \vec{P}_{i,z})$ to the current point point $\vec{P}_{current} = (x, y, z)$ is calculated for all the points in the current cross section and stored in the array $\vec{d}_i$. The point $\vec{P}_i$ with the smallest distance $\vec{d}_i$ is added to $\vec{aP}_o$ and removed from $\vec{aP}$.

5.4 Tube Model

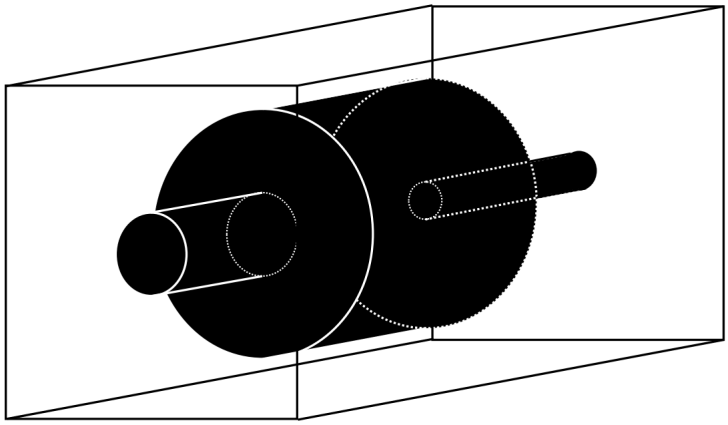


Figure 5.10: Illustration of how the imaginary cylindrical pore space could look like.

During the specialization project, none of the attempts to train a CNN-model indicated that the model was learning. Since the code used to build the CNN-model in this project was made with the purpose to estimate porosity of CT-scan images of rocks, it would be useful to ensure that the CNN architecture is not part of the problem. For the purpose of testing the CNN algorithm, images of simpler geometries than the images generated from the grid was made. The simplest way would be to generate images of cylinders with a constant radius. To make the learning slightly more difficult, the images were chosen to be cross sections of a tube system of three connected cylinders with a semi-random radii. The reason why the radii is semi-random is because they are restricted by a minimum and a maximum value that is dependent on the length of the tube model. To have the same structure as the images generated from the grid, the training images generated from the tube models consists of sixteen circles of three different sizes, as illustrated in Fig. 5.11.

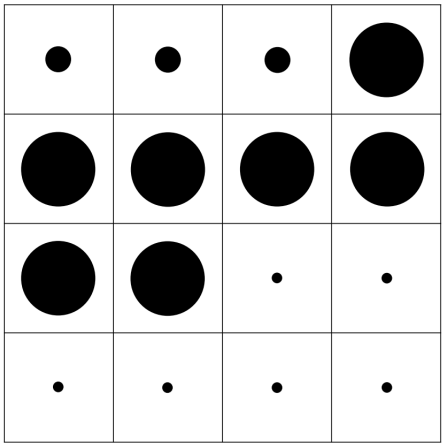


Figure 5.11: Illustration of the training image that could be generated from the tube model in Fig. 5.10

The images should be structured in the same way as the original images, meaning that the nodes, or inlet/outlet, should be placed in the top left and bottom right corner, with the throat going from left to right, line by line. Each image were generated by semi-randomly generating three real valued lengths, l_1, l_2, l_3 $\in [10, 120]$, where $l_{tot} = l_1 + l_2 + l_3$. Then three radii, r_1, r_2, r_3 are randomly generated, restricted by r_1, r_2, $r_3 \in [10, 120]$ or $\in [10, 0.85 l_{tot}]$. The number of images of each of the three radii is decided by the ratio l_i/l_{tot}, where l_i represents one of the three lengths. A high l_i/l_{tot} means that the circle of radius r_i will be included in a high number of the images.

Fig. 5.11 illustrates what the training image for the pore space in Fig. 5.10 could look like. Each of the individual cross section images has the spatial dimension $l_{tot} \times l_{tot}$, hence the total image has a dimension of $4l_{tot} \times 4l_{tot}$.

5.5 Hydraulic Conductivity - Labels

The hydraulic conductivity, g, is a measure of how easily fluids can flow through the pores of a porous media. The relation between the flow rate, Q and g is described by

$$Q = g\Delta P. \tag{5.7}$$

Two sets of images are illustrated in Fig. 5.12, hereby referred to as set a) and set b). Set a) and b) are of equal size, with cross section images of the same pore-throat-pore system. The only difference is that the images in set b) includes voxels from a larger area of the same cross section. The prediction for set a) and set b) is referred to as G_a and G_b. For the images in Fig. 5.12, $G_a > G_b$, even though the two sets contains the exact same pore elements.

The labels need to be scaled based on the images spatial extent. In this project Eq. 3.10 was used to calculate the hydraulic conductivity for each of the network elements. Eq. 3.11 was used to calculate the effective hydraulic conductivity. As the images them selves give no clue about the actual size of the pore or the distance between the two connected pores, the hydraulic conductivities need to be scaled. Each of the labels are scaled based on its associated images spatial extent, before they can be used for training.

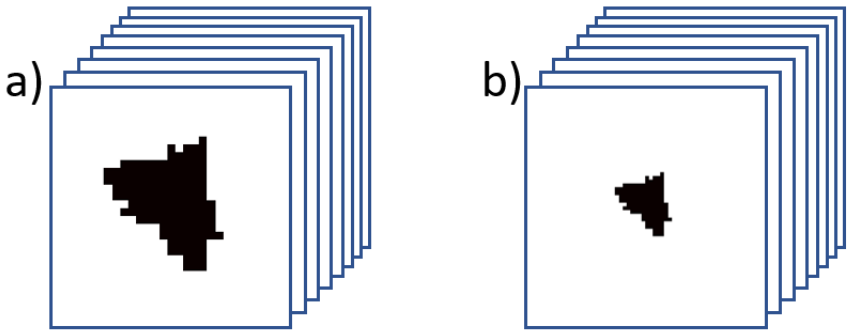


Figure 5.12: Two data sets of the same pore at different scale. The predicted conductivity for set a) would most likely be higher than for set b).

The hydraulic conductivity calculated from Eq. 3.10, $g = \frac{3}{80}\frac{r_{insc}^4}{\mu G}$, has the unit $[m^5 s/kg]$, compared to Eq. 3.7, $g = \frac{\pi r_{insc}^4}{8\mu L}$, which has the unit $[m^4 s/kg]$. The two equations give different units, which means they will have to be treated differently when calculating the flow rate, Q. A change in pressure can lead to a change in flow. To calculate Q when g is found by using Eq. 3.7, the relation $Q = g\Delta P$ is used. While using equation Eq.3.10 to calculate g, the relation is $Q = g\nabla P$, i.e. the pressure gradient instead of the pressure difference. The unit difference is due to the fact that the length of the medium is included in Eq. 3.7, while it is not included in Eq. 3.10.

After multiplying G_a and G_b by their individual scaling factors, the predictions for the hydraulic conductivities should be equal, $g_{ha} = g_{hb}$.

5.5.1 Scaling the Labels

Figure 5.13 illustrates a cuboid that represents an arbitrary subvolume between two nodes in the rock sample. The spatial extent of all the subvolumes in the rock sample may vary a lot. These differences are not present in the cross section images that are used to train the model. Each subvolume has its own hydraulic conductivity, g. To eliminate the differences in g caused by the different sizes of the subvolumes, g must be scaled based on these differences. A scaled variable will be denoted by " $'$ ". Two factors a and b will be used to scale properties of the cuboid.

The area of the yz-plane faces is scaled in the following way: $A'^2 = b^2 A^2$. Flow rate is a function of the cross section area, and is scaled by $Q' = b^2 Q$. Darcy's law, $q = -\frac{k}{\mu L}\Delta P$, describes the flow through a porous media for a single phased,

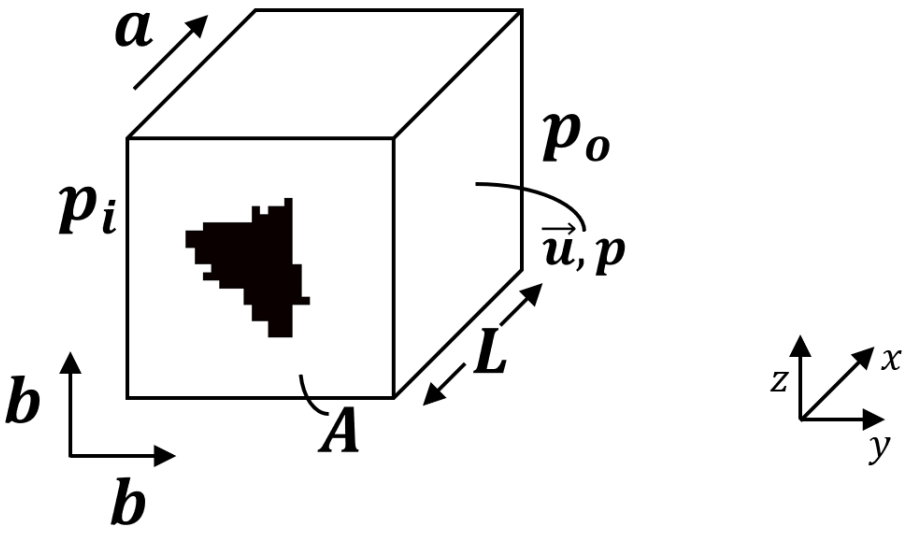


Figure 5.13: An illustration of a subvolume between two nodes. Illustrated on the front face of the cuboid is the center of a pore body, a node. The pore body is connected to a pore throat, which is connected to another pore body. The cross section area of the cuboid is A, and the length is L. Inlet and outlet pressure is p_i and p_o. The velocity of the fluid inside the subvolume is described by the velocity vector $\vec{u}$, which also describes the pressure.

stationary, incompressible fluid flow. Fluid flux, q, is a function of permeability, k, dynamic viscosity μ, length of the medium, L and the pressure change, ΔP. By rearranging, ΔP becomes a function of L. It is scaled by the factor a: $\Delta P' = a\Delta P$. On gradient form, Darcy's law is $q = -\frac{k}{\mu}\nabla P$. The pressure gradient is independent of L.

To solve Stokes equation, Eq. 3.14 need some assumptions: 1) The pressure change in z-direction is assumed to be close to zero. This is due to the small height of the rock sample. With a height of 7.17 mm, the maximum hydrostatic pressure difference in the rock sample would be:
$\rho gh = 1000\ kg/m^3\ 9.81\ m/s^2\ 7.17 \cdot 10^{-3}\ m \approx 70\ Pa$.
2) The flow is assumed to be mainly in the x-direction. The pressure change is assumed to be mainly in the flow-direction, $\partial P/\partial y = \partial P/\partial z = 0$. Stokes equation is then reduced to $\frac{\partial P}{\partial x} = \mu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right)$.

In the fluid viscosity vector $\vec{u}$, v and w are cancelled out by the 2nd assumption. Since flow is only assumed to be in the x-direction, u only works on the surface where the scaling factor b^2 is applied. Applying the scaling factors give

$$\frac{\partial aP}{\partial ax} = \mu\left(\frac{\partial^2 b^2 u}{\partial (ax^2)} + \frac{\partial^2 b^2 u}{\partial (by)^2} + \frac{\partial^2 b^2 u}{\partial (bz)^2}\right)$$

If the velocity in x-direction is constant, then $\partial b^2 u/\partial(ax^2) = 0$.

$$\frac{\partial aP}{\partial ax} = \mu\left(\frac{\partial^2 b^2 u}{\partial(by)^2} + \frac{\partial^2 b^2 u}{\partial(bz)^2}\right)$$

, and scaling factors cancels out.

$$\frac{\partial P}{\partial x} = \mu\left(\frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right)$$

If P and u is a solution, then aP and $b^2 u$ is a solution.

$Q = u^2 A = Pg$, $Q' = b^2 Pg$

$$g = \frac{Q}{\nabla P} \tag{5.8}$$

since

$$g' = \frac{Q'}{\nabla P} = \frac{b^2 Q}{\nabla P} = b^2 g \tag{5.9}$$

$A' = b^2 A$

$$\frac{g'}{A'^2} = \frac{g'}{b^2 A^2} = \frac{g}{A^2} \tag{5.10}$$

To sum up this section: the labels for the data sets used to train the model needs to be scaled based on the spatial extent of the subvolume the images are generated from. Each label is scaled by dividing it by the cross section area squared, A^2, of the cuboid subvolume.

5.6 Logarithmic Labels

The plan was originally to train the models using the scaled g's, $\hat{g}$'s, as labels. For the rock sample subvolume used in this thesis, the maximum $\hat{g}$ is 1.89 $[m^5 s/kg]$, while the minimum value is $2.03 \cdot 10^{-8}$ $[m^5 s/kg]$. This means that the labels of the images spread across 8 magnitudes, which is a huge variation. To get precise and accurate predictions demands a very well trained model. After training a model, some critical problems were discovered.

One of the problems was that the model gave the same prediction for a large number of the samples in the test set. For over 15 % of the test set, the model predicted the $\hat{g}$ to be $9.98 \cdot 10^{-5}$. This was also the lowest prediction the model gave. Another problem was that the model gave negative predictions, which from a physical perspective is preposterous. As an attempt to solve these two issues, the

model was trained, validated and tested using log10-values of the labels. In this way, the problem with the large spread when it comes to number of magnitudes is removed and replaced with a need for precision in decimal numbers. It also removes the possibility for predictions to negative (unphysical) values. The labels will therefore be logarithmic.

5.7 Choosing an Efficient Number and Resolution of Cross Sections

There are multiple factors to take into account when choosing the number of cross sections included in the data set images. A high number of cross sections and a high image resolution demands a lot more computational power compared to fewer cross sections and poorer resolution. On the other hand, it is crucial to include an efficient amount of information from the subvolume in the images. As described earlier, the cross section images represents an average of the cross section it is generated from. Both the number of cross sections and the cross section image resolution could possibly affect the results of the predictive skills of the CNN-model. The optimal number of cross sections may vary from link to link. A plot showing the distribution of the lengths is shown in Fig. 5.14. From the plot it is clear that there is a great differences in distance. The red line marks the 95th percentile. It tells us that 95% of the nodes connected by a link is located at a distance of equal to or less than 32 voxels. On the other hand, 5 % has a distance between 32 and 1400 voxels. For the image structure suggested in Section 5.3.1, the number of cross sections needs to be a square number. As mentioned in Section 3.1.1, it is common for the inputs of CNN networks to have a dimension of 32×32, 64×64, 128×128 or 224×224. To choose a number of cross sections that suits every link is ambitious. It should primarily suit the great majority of the links.

Some of the links in the network may be too long to be properly represented by the chosen number of cross sections. An interesting topic to investigate is how these links influence the training of a model. During hyperparameter tuning and model training, the model is tuned based on the input samples. If a link is poorly represented due to too few cross section images, the accuracy and precision of the model may be influenced. The model would probably be tuned to try to handle such images as well. If it handles those images probably would depend on the amount of such images in the data set.

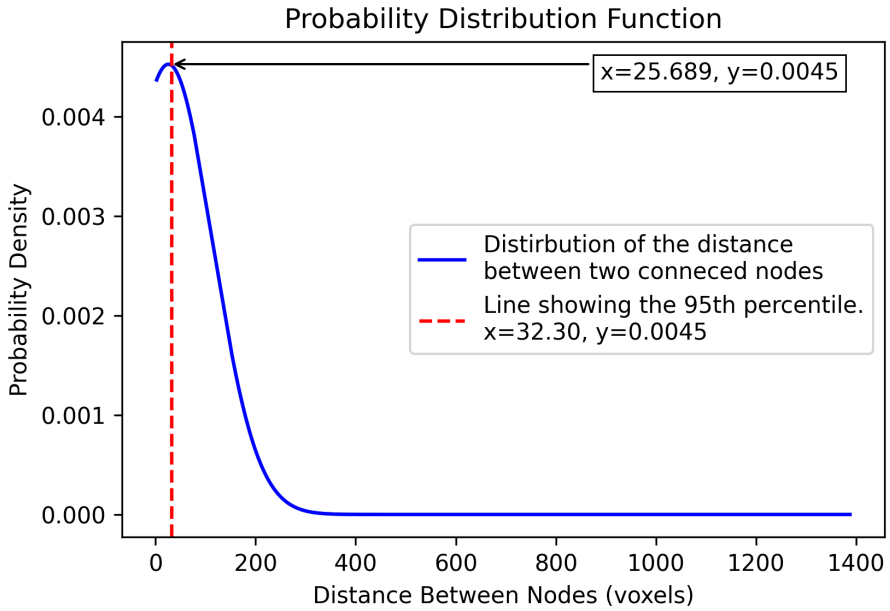


Figure 5.14: The node-lengths are distributed as a right-skewed normal distribution. The red dotted line shows the 95-percentile. It has a value of 32.3 with a probability density of 0.0045. The arrow points out the maximum value, which is found at $x = 25.7$.

5.8 Handling the wide spectre of labels

5.8.1 Choosing the Number of Cross Sections Image Resolution

Another thing that may be worth investigating is the impact of large variations in $\hat{g}$. Fig. 5.15 shows a normal distribution plot of the labels in the data set. If the model is only exposed to a few images of high and low labels during training, it might impact the predictive skills of the model. This meaning that the model might not get to adapt to those cases well enough. The overall predictive skills of the model might also be impacted if the model tries to adapt to the extreme cases, but not adapts well enough. A cumulative distribution function of the labels is shown in Fig. 5.16. The 5th percentile and 95th percentile are marked by a dotted line. The plot tells us that there is a high probability that the models will be trained with an insufficient amount of high- and low-conductance links if the training images are picked by random. The models will therefore be trained using data sets that somewhat matches the normal distribution curve of the complete data set.

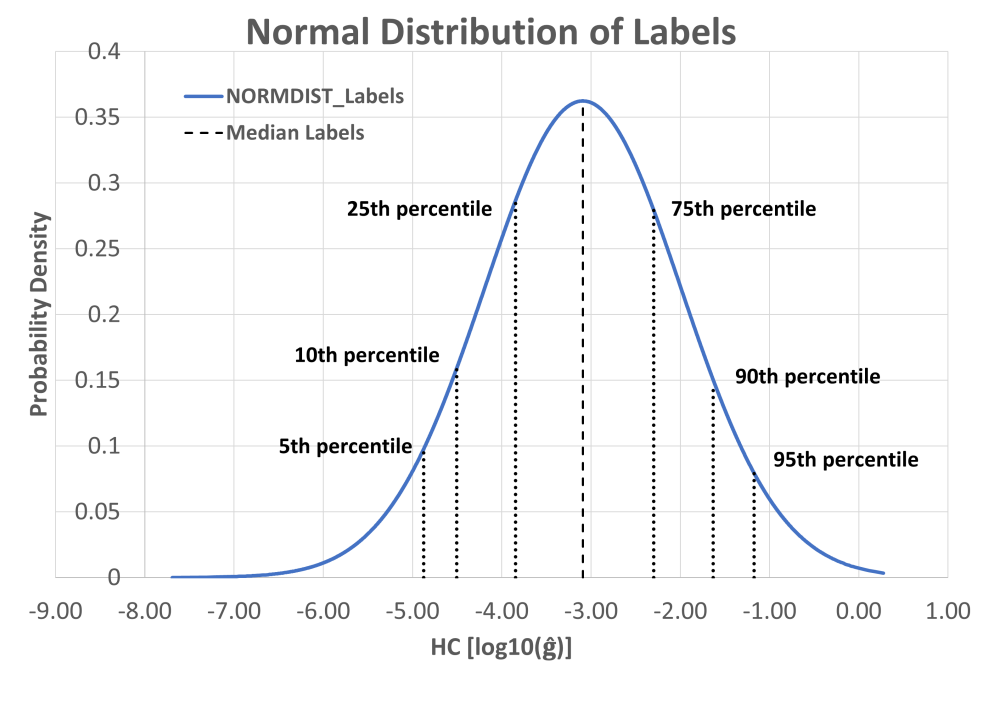


Figure 5.15: The distribution curve for the data set labels. The median label is slightly lower than -3.00.

5.8.2 Multiple Models for Different Cases

It could be worth investigating the result of training three different models - one for the highest labels, one for the lowest and one for the majority. One model will therefore be trained using 5th percentile images only, one model will be trained using only images above the 95th percentile, and one model will be trained using only images between the 5th percentile and 95th percentile. The goal is to investigate if training three specialized models will give better predictions than a model trained on the whole spectre of labels.

5.9 Post-Process Image Augmentation

Image augmentation is a technique that is used for altering existing data to create more data. In this case, rotation and flipping of images will be done to investigate if it the quality of the predictions are effected. The label, $\hat{g}$, of an image will be same, independent of rotation or flipping, due to the way $\hat{g}$ is defined. All parameters used for calculating $\hat{g}$ for a link is the same no matter the orientation of the images. The amount of training images is believed to be sufficient, and the image

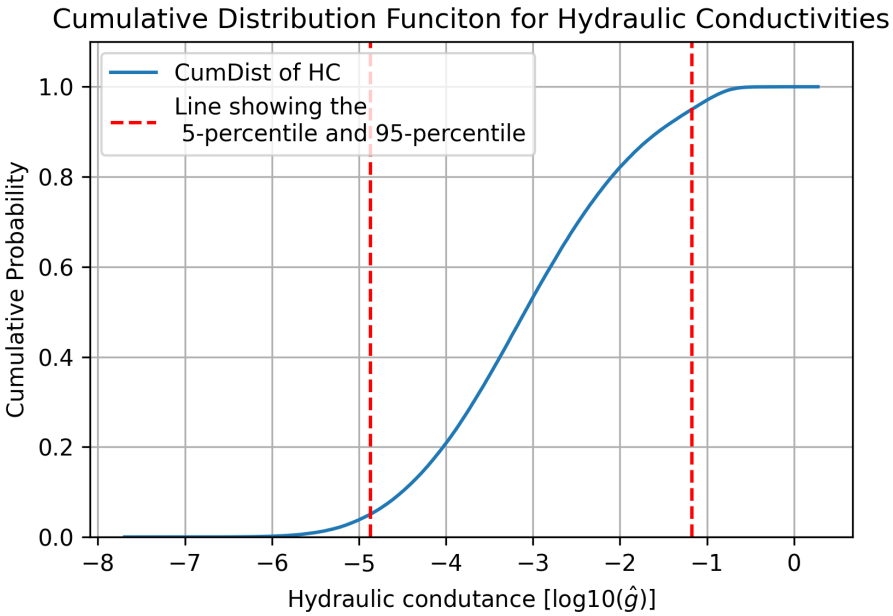


Figure 5.16: A cumulative distribution function for the labels in the data set. The dashed, red lines shows the 5th and 95th percentile.

augmentation will therefor be done only to investigate if the predictions of the models are influenced. The way this will be done is by rotating a data set by 90°, 180° and 270°. Then the four data sets will be flipped horizontally, which gives a total of 8 sets of images for each label. The model will be set to make predictions for all the 8 data sets, and an average of all the predictions for each label will be calculated. There may be a lot of different outcomes of this investigation. The model could give the same prediction for the different orientations of the images. This means the accuracy of the model is well-adjusted. Another possibility is that the model gives different predictions for the different orientations. In that case, the assumption is that an average of the predictions will be a better prediction than the prediction for the image that is not augmented.

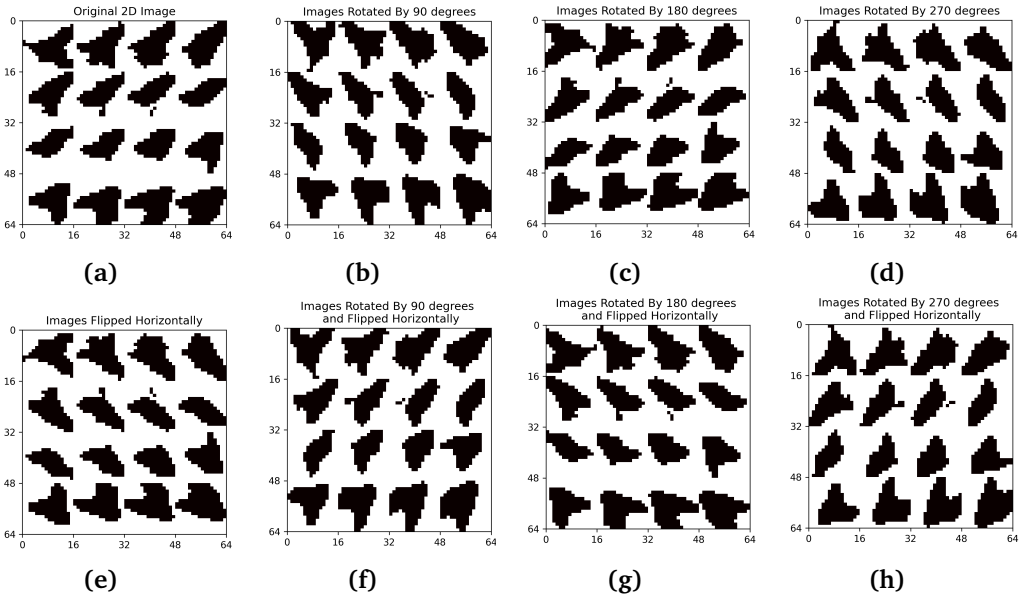


Figure 5.17: Illustration of post-processing augmentation applied to the images. The original image is **a)**. Image **b)**, **c)** and **d)** rotates each individual image in **a)** by 90°, 180° and 270° counterclockwise. After rotation, the images in **a)**-**d)** are flipped around the horizontal axis, **e)**-**h)**. The label, $\hat{g}$ is preserved, and the result is eight images with the same label.

5.10 Error Functions

The CNN models were trained to minimize the MAE of its predictions. This method will now be introduced. A method for calculating the precision of each individual prediction will also be introduced. This method is called the relative error (RE).

Mean Absolute Error

A frequently used error function to measure accuracy in CNN is the error function Mean Absolute Error (MAE).

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|p-l| \tag{5.11}$$

where p is the predicted value l is the true value, and the number of predictions is n. This method works fine as long as it is used while labels and predictions are on logarithmic form, $log10(\hat{g})$. If the labels are on the form of $\hat{g}$, the labels differ by many orders of magnitude. Then the MAE will not be able to pick up the correct error. Let us illustrate this by an example:

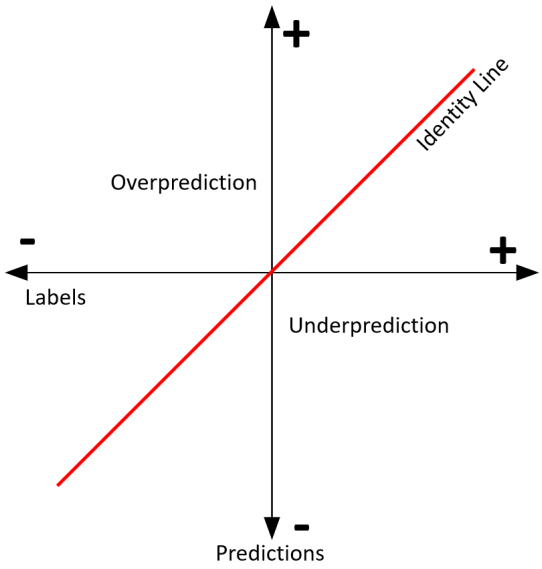


Figure 5.18: Illustration of overprediction and underprediction with the use of *RE*. A predicion to the right of the identity line is an underprediction. A prediction to the left is an overprediction.

- Let us say the label for an image is $l = 10^{-6}$ and that the prediction is $p = 10^{-8}$. Its contribution to the MAE will be $|p - l| = 9.9 \cdot 10^{-7}$.
- Now, let us change p from an underprediction by magnitude 2 to an overprediction by magnitude 2. If $p = 10^{-4}$ and $l = 10^{-6}$, the contribution will be $|p - l| = 9.9 \cdot 10^{-5}$.

This illustrates the fact that if the model underpredicts or overpredicts by the same magnitude, the effect of overprediction is influencing the MEA by a higher degree. In addition, the higher magnitude of the label, the greater impact it potentially has on the MAE.

Relative Error

Relative Error is an error function that will be used to analyze the predictions of the models. It gives the error relative to the expected value, in this case the label. *RE* is usually defined as

$$RE = \frac{p - l}{l} \tag{5.12}$$

As for the MAE, this method is useful as long as the labels are on the form $log10(\hat{g})$. Otherwise, it will weight an underprediction and an overprediction

differently. To illustrate this by an example:

- Let us say that $p = 10^{-3}$ and $l = 10^{-5}$, $RE = (10^{-3} - 10^{-5})/10^{-5} = 99$, which is an overprediction by 9900%.
- Now, let us say that $p = 10^{-5}$ and $l = 10^{-3}$, $RE = (10^{-5} - 10^{-3})/10^{-3} = -0.99$. This is an underprediction with the absolutevalue of 99%.

In both examples the prediction misses by a magnitude of 2. Both the underprediction and the overprediction should ideally be treated equally. For log10-values of labels, this method works fine. For labels that are on exponent form a different method is needed.

Fig. 5.18 illustrates the zones for an underprediction and an overprediction. If $Label < 0$, $RE < 0$ for an overprediction and $RE > 0$ for an underprediction. If $Label > 0$, $RE < 0$ for an underprediciton $RE > 0$ for an overprediction.

Modified Relative Error

A possible solution for calculating the error of the labels on exponential form is to use the relative error (RE), but to change the denominator to be $Prediction + Label$. In this way, it is the difference between the $Prediction$ and the $Label$ that determines the error.

$$RE = \frac{p - l}{p + l} \tag{5.13}$$

In this case, an underprediction and an overprediction of the same magnitude is weighted equally. Let us use the same example as the example used for RE.

- $RE = (10^{-3} - 10^{-5})/(10^{-3} + 10^{-5}) = 0.99$, which is an overprediction of 99%.
- $RE = (10^{-5} - 10^{-3})/(10^{-5} + 10^{-3}) = -0.99$, which is an underprediction -99%,

where the absolute value of these two examples is equal. This method is more of a measure of the difference between the prediction and the label. A great underprediction will result in the same RE as an equally great overprediction.

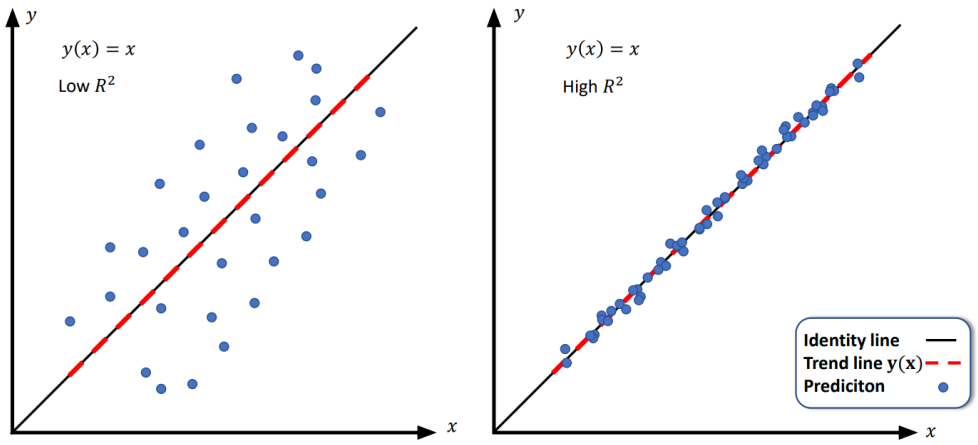


Figure 5.19: Comparison of two cross plots. They give an identical trend line, but different R^2. The graph to the left is accurate, but not precise. The graph to the right is both precise and accurate.

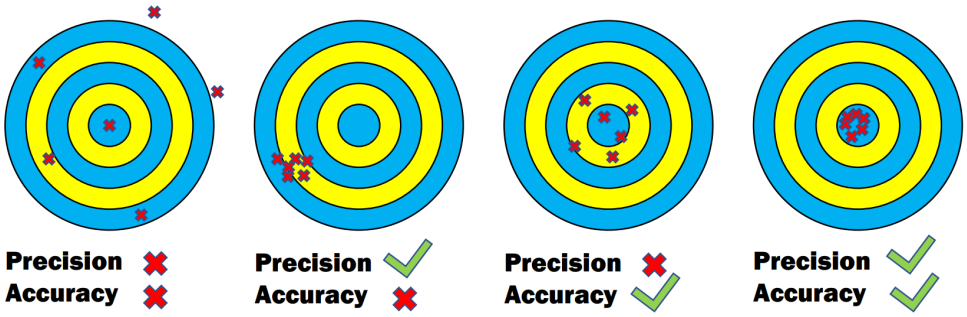


Figure 5.20: Illustration of the difference between a high R^2 and a good trend line. A trend line of $y(x) = x$ would indicate that the data points are oriented around the identity line, the accuracy. While a low or high R^2 would indicate if the data points have a high or a low variability.

5.10.1 Trendline and R^2

After building a CNN model, it predicts a test set to provide an evaluation of the model. The predictions may be plotted against the corresponding labels as a scatter plot to inspect the quality of the predictions. Ideally, the scatters orient around the identity line. A precise model is able to make more or less equally accurate predictions for most of the images. An accurate model is able to make predictions close to or equal to the labeled value. A precise and accurate model is able to give predictions more or less equal to the labeled value.

A perfectly trained model would give predictions with a trend line function of $y(x) = x$ and a $R^2 = 1$. R^2 describes the precision of the trend line, which again describes the accuracy of the estimations. High precision means that the predic-

tions are "predictable" or "collected", etc. High accuracy means that the predictions give results close to the expected value. Predictions oriented around the identity line will have a trend line close to $y(x) = x$, with a slope $m \approx 1$, as the right plot in Fig. 5.19. A noisy, high-variability plot as the left plot in Fig.5.19 can have a significant trend. The trend indicates that the model is able to extract information from the images, and make predictions that is reasonable as a collective, and not individual. The low R^2 indicates that the trend line does not fit the data points.

Chapter 6

Results & Discussion

In this chapter, the results will be presented and discussed. First, an example of a 2D and 3D image will be presented. Then different CNN models will be introduced and compared. As mentioned in Section 4.3.1, the models were trained to minimize the *MAE*. A further investigation of the predictive skills of the models will be done by comparing $|RE|$ and RE, as introduced in Section 5.10. For the models where it is possible, both the results of a 2D CNN and 3D CNN model will be investigated. Finally, the predictions of one of the CNN models will be used for flow property estimation of a network model.

6.1 Results - Images

6.1.1 Cubic Images

As mentioned in Section 5.2, some links was not included when generating images. All the links that generated points outside of the grid were ignored. This also includes all the links directly connected to the inlet or outlet. Fig. 6.1 shows a distribution curve for the number of voxels between two interconnected pores. As the plot shows, the amount of long-distance links are drastically reduced when compared to Fig. 5.14. With this distribution, it could be fair to assume that 16 images could be an appropriate number of cross section images for each link. This means that a majority of the links will have approximate one image for each layer of voxels. For some links with a short distance between them, this might be superfluous, as several cross sections can be equal. For other long-distance links, this will be too few layers to capture all the details, however, it might be sufficient for the models to give good predictions.

To generate both the 2D and 3D images, the cubic extraction method with 16

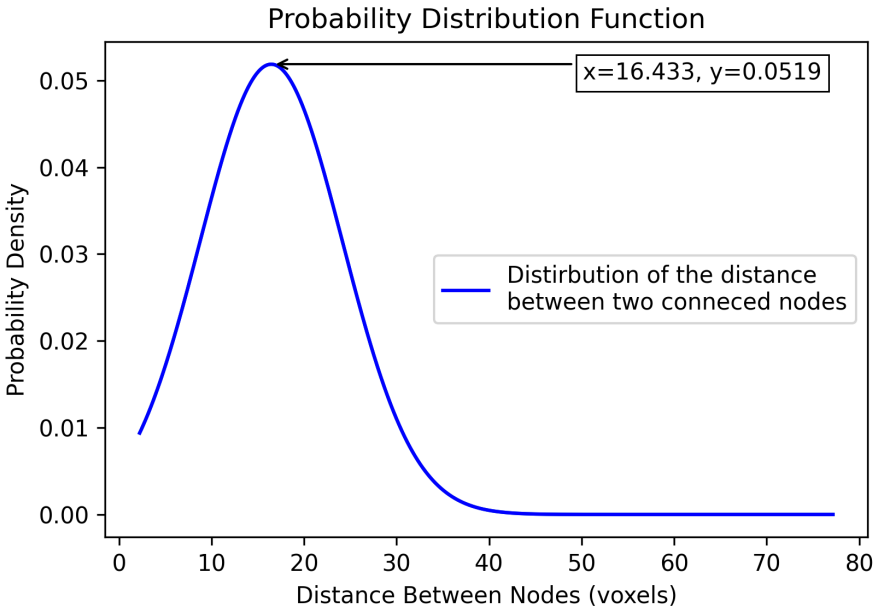


Figure 6.1: A plot showing the distribution of the links that was used for genereating images.

cross sections was used. An example of a 2D image and one 3D image cross section are provided in Fig. 6.2. In this thesis, cross section images with the dimension 16×16 and 32×32 has been used. An arbitrary dimension is therefore referred to as $n \times n$.

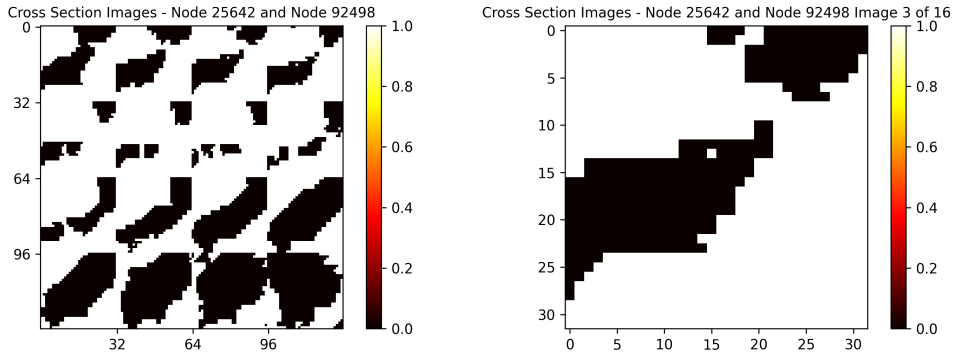
6.1.2 2D images

The 2D images were generated with cross sections of the size $n \times n$. Structured in the way described in Section 5.3.1, the total image has the size $4n \times 4n$. Each image contains a node in the upper left corner, between 0 and n on both the vertical and horizontal axis and a node in the bottom right corner, between $3n$ and $4n$ on both axis.

6.1.3 3D images

The 3D model stores each of the 16 cross section as an individual image. Each image has the size $n \times n$. Figure 6.2b shows one of the 16 images generated for the same node pair as in Figure 6.2a. Fig. 6.2a is an image of the third cross section, and can be seen in 6.2a between 0 and 32 on the vertical axis and 64 and

96 on the horizontal axis.



(a) Example of a training image used for the 2D-model. The dimension of the cross section images is 32×32, while the total dimension of the images is 128×128. The image represents the connection between the node 25642 and node 92498.

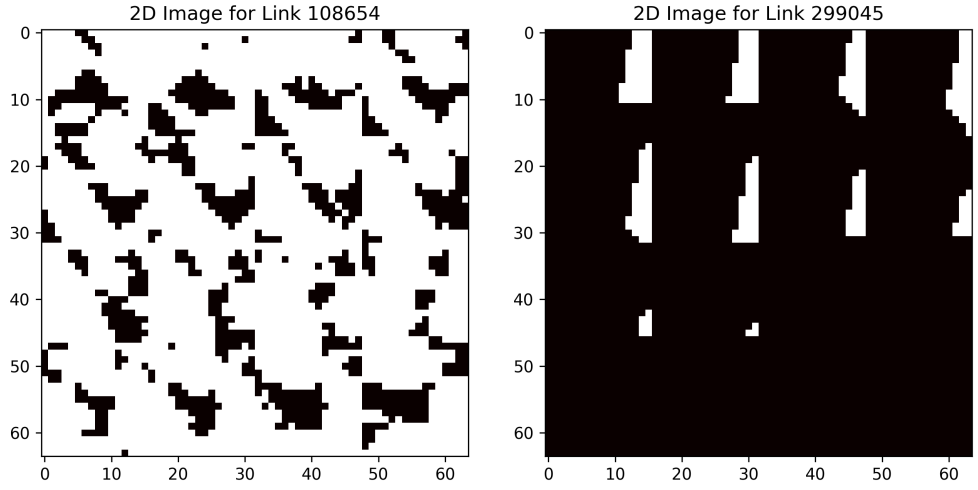
(b) An example of a training image used for the 3D-model. The dimension of the image is 32×32 This is the image created from the third cross section between node 25642 and node 92498.

Figure 6.2: Two different images created for the connection between node 25642 and node 92498. The image to the left is an example of the images used for the 2D-model. The image to the right is one of the 16 images used for the same connection for the 3D-model.

6.2 Discussion - Images

The way the images are generated may be suboptimal. Fig. 6.3a illustrates the data set image for link 108654. From this image, it is hard to tell what part of the pore pixels that belong to the node. It seems like the cross section images include pore from surrounding pore elements. In that way, the image does not represent the link it is supposed to represent. Thus the model is making predictions based not only on the current link. The image in Fig. 6.3a has a label of $log10(\hat{g}) \approx -7.69$, and is one of the samples with the lowest label.

One of the samples with the highest label is shown in Fig. 6.3b. It illustrates the training image for link 299045, with a label of $log10(\hat{g}) \approx -0.43$. The last six cross section images for this sample are all dark. With this great amount of pore-pixels, it is clear that details about the size and structure about the pore elements are missing. This could be crucial for the predictions, since the model does not know the extent of the pore. It is possible that the pore taking up the complete or almost complete cross section leaves some information about the size of the $log10(\hat{g})$, but a precise prediction would be difficult for such an image.



(a) Data set image for link 108654. For this link, $log10(\hat{g}) < -7$.

(b) Data set image for link 299045. For this link, $log10(\hat{g}) > -0.5$.

Figure 6.3: Comparison of an image with a low label and an image with a high label.

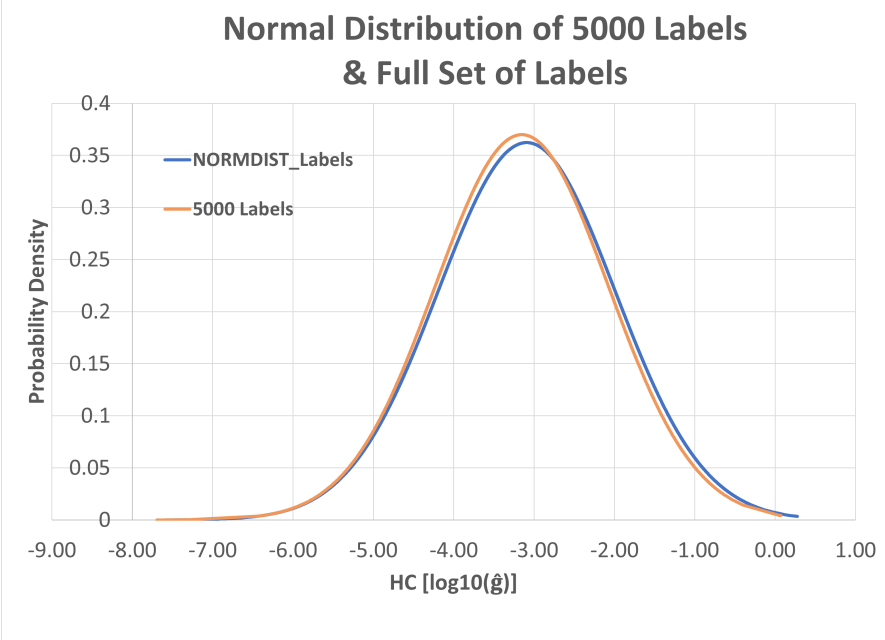


Figure 6.4: Distribution plot of the full set of labels, and the 5000 labels used as a common test set.

6.3 CNN Setup

6.3.1 Training and Validation Sets

For all the CNN models, the training and validation set have a distribution similar or equal to the distribution of the labels for the complete data set. The distribution curve of the complete data set is found in Fig. 5.15. The validation sets have had a size of 20 % of the training set size. The number of cross sections and the dimensions of the cross section images was 16×16 unless otherwise specified.

6.3.2 A Common Test Set

When comparing the predictive skills of the models, a common test set is used. The test set consists of 5000 images. None of the images in the test set have been included in a training set or validation set for any of the models. A distribution curve of the test set is shown in Fig. 6.4. It shows a similar distribution as the full set of labels.

6.3.3 Underpredictions & Overpredictions

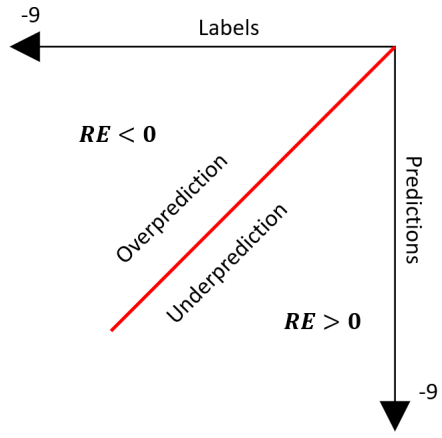


Figure 6.5: A simplification of Fig. 5.18. This illustration is only valid for labels and predictions < 0. In the test set, there are only 2 labels and predictions that is not within these boundaries.

As illustrated in Fig. 5.18, a prediction that fall to the right of the identity line is an underprediction, and a prediction that fall to the left of the identity line is an overprediction. For an underprediction, RE may be both positive or negative, depending on the label and prediction. To make it easier to analyse, the sign of RE will be modified depending on if it is an underprediciton or an overprediction.

Fig. 6.5 shows a modified version of Fig. 5.18. For negative predictions and negative labels, $RE > 0$ for an underprediction, and $RE < 0$ for an overprediction. The distribution curve in Fig. 6.4 shows that a few labels in the test set are positive. An underprediction for a positive label would usually mean $RE < 0$, while $RE > 0$ for an overprediction. For simplification, and because there are only two positive labels in

	Model Names		
of Training Samples	**5000**	**25000**	**50000**
2D CNN	2DM5k	2DM25k	2DM50k
3D CNN	3DM5k	3DM25k	3DM50k

Table 6.1: The models are named based on the amount of training data used, and whether they are used for predicting 2D or 3D input.

the test set, the sign of the *RE* for the positive labels will be changed. This is done to make all overpredictions $RE < 0$ and all underpredictions $RE > 0$. This will eventually make the error analysis a bit more intuitive.

6.4 Different Amount of Training Data

This section will present and discuss the results of three 2D and 3D CNN models. The models have been trained using different amount of training images. For both 2D and 3D, one model was trained using a training set of 5000 samples, one was trained using 25000, and one using 50000 training samples. The data samples included in the three sets are the same for both 2D and 3D CNN. The different models are named, and the names of the models in this section are found in Table 6.1. In this section, the effect of an increased amount of training data will be investigated. Theoretically, increasing the amount of data will build a better model.

6.4.1 2D CNN Models

The *MAE* for each model and descriptive statistics regarding the different models are included in Tab. 6.2. The $|RE|$ is calculated using Eq. 5.12. The table also includes characteristics of the trendline of the predictions. Fig. 6.6 shows a cross plot of the predictions of model 2DM5k for the test set. The trendline of the cross plot has the slope $m = 0.8244$ and the reliability $R^2 = 0.7738$. The cross plots for every model will not be presented, but the trendline characteristics will.

Table 6.3 contains information about the $|RE|$ and the *RE* of 2DM25k and 2DM50k. It compares the error of the models in three different categories. **Low** and **High** are the labels within the 5th percentile and above the 95th percentile of the labels. **Rest** refers to the labels between the 5th and 95th percentile.

Let us first have a look at the *MAE*, which is the function that the models have tried to minimize during training. The *MAE* decreases gradually as the amount of training data increases. The improvement from model 2DM25k to 2DM50k is

	2D CNN Models		
MAE	**2DM5k**	**2DM25k**	**2DM50k**
MAE	0.4066	0.3984	0.3800
|RE|	**2DM5k**	**2DM25k**	**2DM50k**
Average	14.41 %	14.00 %	13.21 %
Median	11.41 %	10.73 %	10.14 %
Max	1357.49 %	1455.90 %	1568.65 %
Min	0.0039 %	0.0062 %	0.0017 %
Trendline	**2DM5k**	**2DM25k**	**2DM50k**
R^2	0.7738	0.7902	0.787
m	0.8244	0.8717	0.8557

Table 6.2: Comparing *MAE*, $|RE|$ and trendline characteristics of three different 2D CNN models.

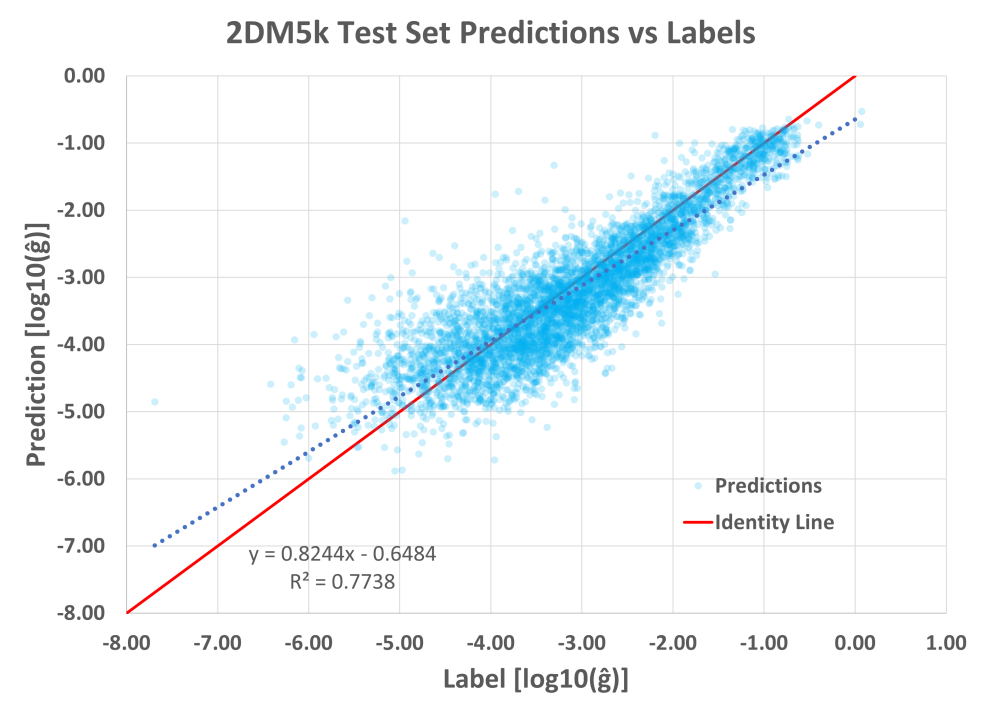


Figure 6.6: A crossplot of predcitions vs labels for model 2DM5k. The dotted blue line is the trendline for the predictions.

	Comparison of Different Amount of Training Data					
2D Model	**2DM25k**			**2DM50k**		
|RE|	**High**	**Low**	**Rest**	**High**	**Low**	**Rest**
Average	30.78 %	13.64 %	13.10 %	31.39 %	15.51 %	12.08 %
Median	13.11 %	13.02 %	10.50 %	13.71 %	14.28 %	9.72 %
Max	1455.90 %	47.35 %	68.20 %	1568.65 %	46.12 %	77.77 %
Min	0.03 %	0.18 %	0.0062 %	0.19 %	0.39 %	0.0017 %
RE	**High**	**Low**	**Rest**	**High**	**Low**	**Rest**
Average	24.04 %	-12.63 %	6.79 %	25.75 %	-14.82 %	4.33 %
Median	7.61 %	-13.01 %	6.39 %	10.63 %	-13.99 %	3.96 %
Max	1455.90 %	23.39 %	68.20 %	1568.65 %	20.55 %	77.77 %
Min	-36.61 %	-47.35 %	-58.14 %	-37.03 %	-46.12 %	-62.73 %

Table 6.3: A comparison of the relative error of the two models. **High** and **Low** refers to the 95th and the 5th percentile labels in the test set of 5000 samples. **Rest** is the remaining samples.

actually higher than the improvement from 2DM5k to 2DM25k. This does not necessarily mean that the predictions of model 2DM50k will always be better than the predictions of model 2DM25k. In Table 6.2, the average and median of the $|RE|$ for the three models indicate that increasing the size of the training set improves the prediction error. Although 2DM50k has the lowest minimum error, it also has the highest maximum error.

Increasing the size of the training and validation set will theoretically improve the results, as the model gets to adapt to more cases. It is therefore interesting to see that the trend line of 2DM25k seems to be the best among the three models. Its R^2 is marginally better than 2DM50k, while m is better by a fair amount. An explanation for this can be that 2DM50k has been exposed to twice as many "extreme-case" images - images with very high or very low label-value. For each input sample, the model is exposed to during training, it adjusts its weights to handle all cases as well as possible. If the model is exposed to several extreme cases, but not many enough to adapt completely, it will surely influence how it solves other cases as well. 2DM50k may have adapted to the extreme cases to a higher degree than 2DM25k, which may have influenced its predictive ability. To investigate this theory, we need to isolate the predictions for higher and lower labels.

Table 6.3 contains information about the $|RE|$ and the RE of 2DM25k and 2DM50k. It compares the error of the models in three different categories. **Low** and **High** are the labels within the 5th percentile and above the 95th percentile of the full data set of labels, as shown in Fig. 5.15. The category **Rest** refers to the labels between the 5th and 95th percentile.

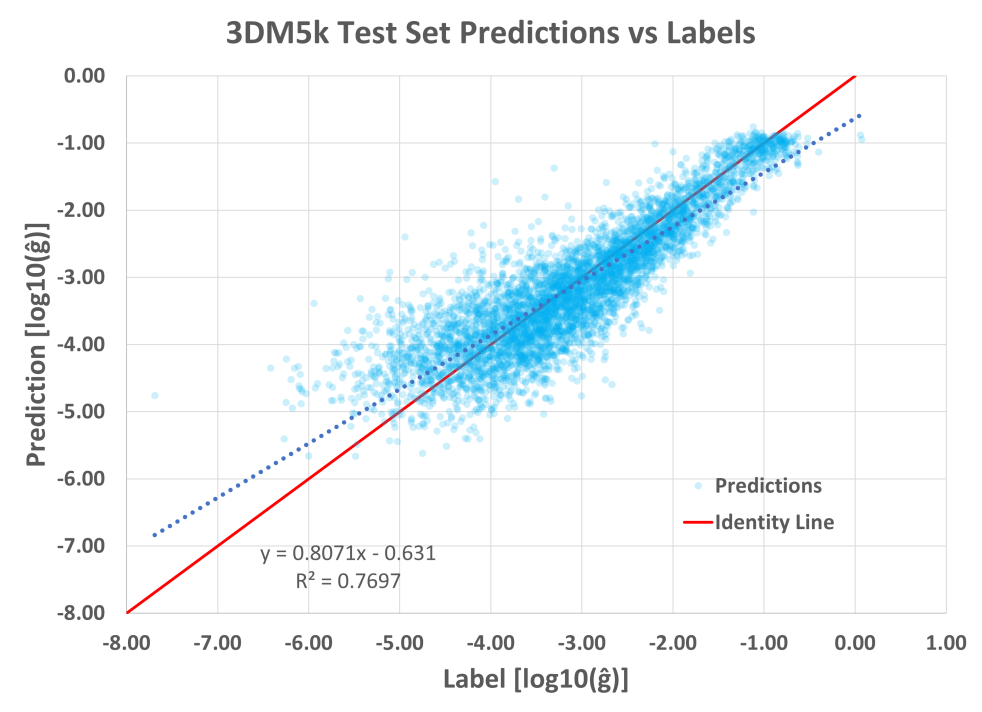


Figure 6.7: A cross plot for the predictions of model 3DM5k vs labels. The blue dotted line is the trendline. The function for the trendline and the reliability of the trendline is also included in the plot.

Both models have similar results for all three categories in Table 6.3. The average and median *RE* for **High**-labels indicate that the models underpredict, and for **Low**-labels they overpredict. This could mean that the both models have a spectre of predictions more narrow than the actual spectre of labels. The *RE* of 2DM50k for **High** and **Low** are respectively higher and lower than the *RE* of 2DM25k. A higher degree of overpredictions for **Low** labels, and a higher degree of underpredictions for **High** labels will naturally tilt the trendline more horizontally, causing a more gentle slope. Table 6.2 shows that the slope m for model 2DM25k is steeper than for 2DM50k.

To compare the performance of the models excluding the labels below the 5th percentile and above the 95th percentile, a cross plot of the **Rest** for both models was made. It gave the trendline characteristics:

- 2DM25k: $R^2 = 0.7476$, $m = 0.9128$
- 2DM50k: $R^2 = 0.7467$, $m = 0.9038$

These trendlines are not that different, but 2DM25k still has the best fit. In Table

	3D CNN Models		
MAE	**3DM5k**	**3DM25k**	**3DM50k**
MAE	0.3939	0.3577	0.3883
|RE|	**3DM5k**	**3DM25k**	**3DM50k**
Average	13.87 %	12.26 %	13.39 %
Median	10.82 %	9.46 %	10.25 %
Max	1633.15 %	1324.77 %	1294.46 %
Min	0.0118 %	0.0008 %	0.0002 %
Trendline	**3DM5k**	**3DM25k**	**3DM50k**
R^2	0.7697	0.8001	0.7785
m	0.8071	0.8226	0.8594

Table 6.4: Comparing the $|RE|$ and trendline characteristics of three different 3D CNN models.

6.3, both the median and the average $|RE|$ for 2DM50k are better than for 2DM25k for the **Rest**-category. This indicates that model 2DM50k actually is better than 2DM25k at predicting the majority of the test samples. Model 2D50k has both a higher maximum and a lower minimum RE, which gives a wide spread of points, i.e., a lower precision. This is also indicated by its R^2, which is lower than the R^2 for model 2DM25k. The difference between the maximum RE of 2DM25k and 2DM50k is higher than the difference between the minimum, which is also indicated by the slope of 2DM25k, which is steeper than the slope of 2DM50k.

Model 2DM25k has a better trendline than 2DM50k, but 2DM50k has a better average and median error. This may be explained by the fact that 2DM50k has been exposed to twice as much data, and thereby, twice as many extreme-valued data samples than 2DM25k. Model 2DM50k may have adapted to these extreme cases, without luck, causing a decreased predictive skill for the extreme cases, and a wider spread of predictions. Its wide spread of predictions may in the end be the reason for not having a better trend line than 2DM25k, even though its predictions are better in general. It is also suggested by the *MAE* that 2DM50k benefits from its increased amount of training data.

6.4.2 3D CNN Models

This section will present the results of the 3D CNN models 3DM5k, 3DM25k and 3DM50k. Descriptive statistics about the $|RE|$ and trendline characteristics for the models are found in Table 6.4. To compare the predictive skills of the two models 3DM25k and 3DM50k, a table containing the $|RE|$ and RE for **High** and **Low** labels

	Comparison of Different Amount of Training Data					
3D Model	**3DM25k**			**3DM50k**		
|RE|	**High**	**Low**	**Rest**	**High**	**Low**	**Rest**
Average	27.32 %	16.95 %	11.16 %	28.84 %	15.34 %	12.43 %
Median	11.99 %	16.30 %	9.12 %	14.93 %	13.84 %	10.00 %
Max	1324.77 %	47.20 %	75.11 %	1294.46 %	51.19 %	77.80 %
Min	0.2056 %	0.0754 %	0.0008 %	0.0002 %	0.4390 %	0.0035 %
RE	**High**	**Low**	**Rest**	**High**	**Low**	**Rest**
Average	18.84 %	-16.79 %	1.37 %	18.09 %	-14.86 %	4.88 %
Median	4.77 %	-16.30 %	0.90 %	6.75 %	-13.84 %	4.44 %
Max	1324.77 %	10.63 %	75.11 %	1294.46 %	9.87 %	77.80 %
Min	-29.13 %	-47.20 %	-59.72 %	-40.15 %	-51.19 %	-56.78 %

Table 6.5: A comparison of the $|RE|$ and RE of the two models. **High** and **Low** refers to the 95th and the 5th percentile labels in the test set of 5000 samples. **Rest** is the remaining samples.

are found in Table 6.5. A crossplot of predictions vs labels for model 3DM5k is found in Fig. 6.7.

A table comparing the MAE, the $|RE|$ and the trendline of the three models 3DM5k, 3DM25k and 3DM50k can be found in Table 6.4. From the table it can be seen that increasing the amount of training data also increases the slope of the trendline. What may be surprising is that the MAE is lowest for model 3DM25k. This is the error function that the models are trained to minimize, and it is therefore unexpected that the MAE of model 3DM25k is lower than the MAE of model 3DM50k.

Model 3DM50k has the steepest slope, with $m = 0.8594$. The most precise model is 3D25k, with $R^2 = 0.8001$. This means that 3DM50k is more accurate, whilst 3DM25k is more precise. Model 3DM5k has a a precision close to the one of 3DM50k, with $R^2 = 0.7697$. Model 3DM25k has the lowest average and median $|RE|$. The median of 3DM50k is closer to the median of 3DM5k than the one of 3DM25k. This, after being trained using 10 times as many images as for 3DM5k.

From Table 6.4 it is clear that both 3DM25k and 3DM50k has better $|RE|$ statistics and trendline characteristics than model 3DM5k. Therefore, only the two models 3DM25k and 3DM50k will be compared further. This is done to investigate the same hypothesis as for the 2D models: if the predictive skills of the model trained using a higher amount of data samples has adapted to the extreme cases to a higher degree, and thereby lost some predictive skills for the majority of the cases.

As for the 2D CNN, the predictions for 3DM25k and 3DM50k are presented in a table, Table 6.4. The predictions are divided into three categories, **High**, **Low** and **Rest**. From Table 6.4 it seems like both models have the same characteristics for each category. Both 3DM25k and 3DM50k tend to overpredict for low labeled input samples. Their maximum error is around 10 %, whilst their minimum error is around −50 %. For **High**-labeled samples, both models have an average *RE* of around 18 %. Model 3DM50k has a higher median $|RE|$ than 3DM25k, even though the median of both models indicate that the models underpredict for most of the labels. The rather high averages indicates that both models tend to underpredict by a great deal for a few samples. Both maximum values are extremely high, *RE* around 1300 %. For each model, there are only two predictions with a $RE > 1000\%$. These predictions have a great impact on the average, and a low impact on the median. Both of the samples are positive labeled, but both models tend to give a negative prediction. This leads to great underpredictions, which impacts the average *RE* by a great amount. Without the two positive labeled samples, the maximum *RE* for 3DM25k would be 117.72 %, and for 3DM50k it would be 109.95 %. The averages would then be 8.85 % for 3DM25k, and 8.78 % for 3DM50k.

The overall predictive skills of model 3D25k seems to be better. Model 3D50k is better at predicting samples with **Low** labels, but for both **High** and the **Rest**, model 3DM25k gives better predictions. For **High** and **Low** labels, the median *RE* is lower for 3DM25k, indicating that the higher average is caused by a few outliers. Model 3D50k has a better slope than model 3D25k, but it also has a wider spread.

For the 3D CNN models, increasing the amount of training data from 5000 to either 25000 or 50000 samples gave an improved model. Increasing the amount of training samples from 25000 to 50000 reduced the predictive skills of the model, except for the lowest labels. It could be that further increasing the amount of data would give a model better suited for the whole spectre of labels. For the models presented in this thesis, increasing the training data from 25000 to 50000 samples did not give an improved model.

6.5 Comparison of 2D CNN and 3D CNN

One could argue that estimating hydraulic conductance is a 3D problem, since g is a spatial property. It is therefore conceivable that 3D CNN works better than 2D CNN for the purpose of predicting hydraulic conductance.

In this section, the results from 2D CNN models and 3D CNN models will be compared. The models 2DM5k and 3DM5k will be compared, and the models 2DM50k and 3DM50k will be compared. In Table 6.6, statistics about the predictions of the two models can be found.

	2D CNN vs 3D CNN				
	5000 Training Images			**50000 Training Images**	
MAE	**2DM5k**	**3DM5k**		**2DM50k**	**3DM50k**
MAE	0.4066	0.3939		0.3800	0.3883
|RE|	**2DM5k**	**3DM5k**		**2DM50k**	**3DM50k**
Average	14.41 %	13.87 %		13.21 %	13.39 %
Median	11.41 %	10.82 %		10.14 %	10.25 %
Max	1357.49 %	1633.15 %		1568.65 %	1294.46 %
Min	0.0039 %	0.0118 %		0.0017 %	0.0002 %
Trendline	**2DM5k**	**3DM5k**		**2DM50k**	**3DM50k**
R^2	0.7738	0.7697		0.7870	0.7785
m	0.8244	0.8071		0.8557	0.8594

Table 6.6: Comparing the error of 2D and 3D models.

6.5.1 Training Set of 5000 Images

A comparison between the 2D and 3D model trained with a set of 5000 samples is found in Table 6.6. It shows that the 2D model has a better trendline. The average and median $|RE|$ of the 3D model is lower than for the 2D model. Its minimum and maximum is higher than for 2DM5k, and may be the reason for its lower R^2. The *MAE* of model 3DM5k is better than the *MAE* of 2DM5k, indicating that the 3D model actually have solved the optimization problem better. The general prediction of 3DM5k is better than for 2DM5k, indicated by the median and average $|RE|$, as well as the *MAE*, despite the weaker trendline. For this low amount of training data, the 3D CNN model seems to benefit slightly.

6.5.2 Training Set of 50000 Images

For the models trained with 50000 samples the 2D model has the best average and median $|RE|$, but only marginally. It also has the highest maximum and minimum error. The trendline of model 3DM50k has a higher m, while 2DM50k has a better R^2, while *MAE* of model 2DM50k is better than 3DM50k.

Table 6.7 compares the $|RE|$ and RE for the predictions of the models 2DM50k and 3DM50k to investigate if there are any clear differences. Both models seems to struggle with the same problem: underpredicting for high labels, and overpredicting for low labels. For **High**, 2DM50k has a lower median $|RE|$, but a higher average. The high average may be explained by the high maximum $|RE|$. Regarding RE, model 2DM50k seems to have a generally wider spread than model 3DM50k.

	Comparison of 2DM50k and 3DM50k					
	2DM50k			**3DM50k**		
|RE|	**High**	**Low**	**Rest**	**High**	**Low**	**Rest**
Average	31.39 %	15.51 %	12.08 %	28.84%	15.34 %	12.43 %
Median	13.71 %	14.28 %	9.72 %	14.93 %	13.84 %	10.00 %
Max	1568.65 %	46.12 %	77.77 %	1294.46 %	51.19 %	77.80 %
Min	0.19 %	0.39 %	0.0017 %	0.0002 %	0.4390 %	0.0035 %
RE	**High**	**Low**	**Rest**	**High**	**Low**	**Rest**
Average	25.75 %	-14.82 %	4.33 %	18.09 %	-14.86 %	4.88 %
Median	10.63 %	-13.99 %	3.96 %	6.75 %	-13.84 %	4.44 %
Max	1568.65 %	20.55 %	77.77 %	1294.46 %	9.87 %	77.80 %
Min	-37.03 %	-46.12 %	-62.73 %	-40.15 %	-51.19 %	-56.78 %

Table 6.7: A comparison of the models 2DM50k and 3DM50k.

For **Low** labels, model 3DM50k has a better average and median $|RE|$, though its maximum and minimum are higher than for model 2DM50k.

When it comes to the trendline and $|RE|$, it is difficult to say which model that is the best. Both models have their pros and cons. Model 2DM50k has the lowest *MAE* and the highest m, while 3DM50k has the highest R^2.

It is surprising to see that the *MAE* suggest that the 2D CNN benefits the most by the increased amount of training data. Hydraulic conductance is, as mentioned, a spatial property, and one could argue that the 3D CNN should benefit from that. It was discussed in Section 6.4.2 that model 3DM50k had a reduced predictive ability, compared to 3DM25k. This might also be reflected when comparing it to model 2DM50k. A quick comparison between model 2DM25k and 3DM25k shows that the 3D model is greater at predicting than the 2D model. Therefore, we have no basis for stating that 2D CNN are better than 3D for the purpose of predicting hydraulic conductance. The 3D models are better with a training set of 5000 and 25000 samples. For the two models 2DM50k and 3DM50k, the 2D model seems to be the best.

It is also worth mentioning that the computational time for the 3D model is significantly higher than for the 2D model. Almost 6 times as long computational time was needed for the 3D model.

6.6 Image Augmentation as Post-Processing

Image augmentation was introduced in Section 2.5.3. It is a way to create more data samples from the already existing samples. Sometimes the samples are used as additional training samples. In our case the images are rotated and/or flipped, and will be used as additional test data. The label of a data sample is the same no matter the orientation of the cross section images. If a model is perfectly tuned, it will give the same prediction for all orientations of an image. A model that is not perfectly tuned will most likely give different predictions for each orientation. The idea is to see if the average of those predictions gives a more narrow spread than the individual predictions, i.e., a lower R^2.

The test set from Fig. 6.4 was rotated by 90°, 180° and 270°, to give 4 differently orientated images. After rotating, the images were flipped around the horizontal axis to make a total of 8 data sets for each label. The model 2DM50k, introduced in 6.4.1, was set to predict eight data sets of 5000 images. An average of the eight predictions was calculated for each label. The $|RE|$ and RE for the average and for the unaugmented data set will be compared.

6.6.1 2D CNN - Original vs Augmented

Table 6.8 compares the predictions of 2DM50k. First of all, the average of the predictions for the augmented images has a drastically improved MAE. This indicates that the predictions in general have become more accurate. In addition, every asset in the table regarding $|RE|$ is better than the $|RE|$ for the model without post-processing. From the trendline characteristics, the precision seem the have improved, with an increased R^2. The slope, m, of the trendline has decreased slightly.

From Table 6.9 it seems like the augmentation tend to reduce the magnitude of the overpredictions and underpredictions. Fig. 6.8 illustrates this in a good way. It shows a plot of the predictions for the eight augmented data sets (blue), and the average of the predictions (red). What can be seen from the plot is that for the sample predictions that are evenly spread at each side of the identity line, the average prediction comes very close to a perfect prediction. For samples where some of the predictions are close to perfect and some predictions are far off, the average will naturally be dragged away from the identity line. As mentioned in the introduction to this section, augmentation was applied to investigate if it could improve the R^2 for the trendline of the predictions, and it clearly does. The maximum and minimum RE for all the categories are reduced, resulting in an improved R^2. For **High** labels, the average and median $|RE|$ and RE increases after augmentation, while it decreases for **Low** labels. The increased error for the overpredictions of **High** labels are higher than the increased error for the underpredictions of **Low**

	2DM50k	
MAE	**Original Image**	**Avg Augmented Images**
MAE	0.3800	0.3490
|RE|	**Original Image**	**Avg Augmented Images**
Average	13.21 %	12.35 %
Median	10.14 %	9.42 %
Max	1568.65 %	1486.96 %
Min	0.0017 %	0.0013 %
Trendline	**Original Image**	**Avg Augmented Images**
R^2	0.7870	0.8207
m	0.8557	0.8506

Table 6.8: Comparison between the predictions of the original test set of 5000 images, and the average of the predictions of the augmented images. The predictions are done by the model 2DM50k.

labels. This leads to a reduction in m, although the reduction is quite small compared to the increase of R^2. It is clear that applying augmentation as it has been done here will reduce the spread of points, i.e., improve R^2.

	Comparing the Results of Original Data vs Augmented Data					
2D	**Original Image**			**Avg Augmented Images**		
|RE|	**High**	**Low**	**Rest**	**High**	**Low**	**Rest**
Average	31.39 %	15.47 %	12.08 %	32.94 %	14.61 %	11.08 %
Median	13.71 %	14.25 %	9.71 %	16.81 %	12.77 %	9.02 %
Max	1568.65 %	46.12 %	77.77 %	1486.96 %	45.45 %	74.99 %
Min	0.1913 %	0.3876 %	0.0017 %	0.2263 %	0.0090 %	0.0013 %
RE	**High**	**Low**	**Rest**	**High**	**Low**	**Rest**
Average	25.75 %	-14.82 %	4.33 %	29.13 %	-14.10 %	5.01 %
Median	10.63 %	-13.99 %	3.96 %	14.12 %	-12.70 %	4.69 %
Max	1568.65 %	20.55 %	77.77 %	1486.96 %	17.49 %	74.99 %
Min	-37.03 %	-46.12 %	-62.73 %	-26.98 %	-45.45 %	-59.74 %

Table 6.9: Comparing the $|RE|$ and RE for different parts of the distribution curve. The predictions were done by model 2DM50k.

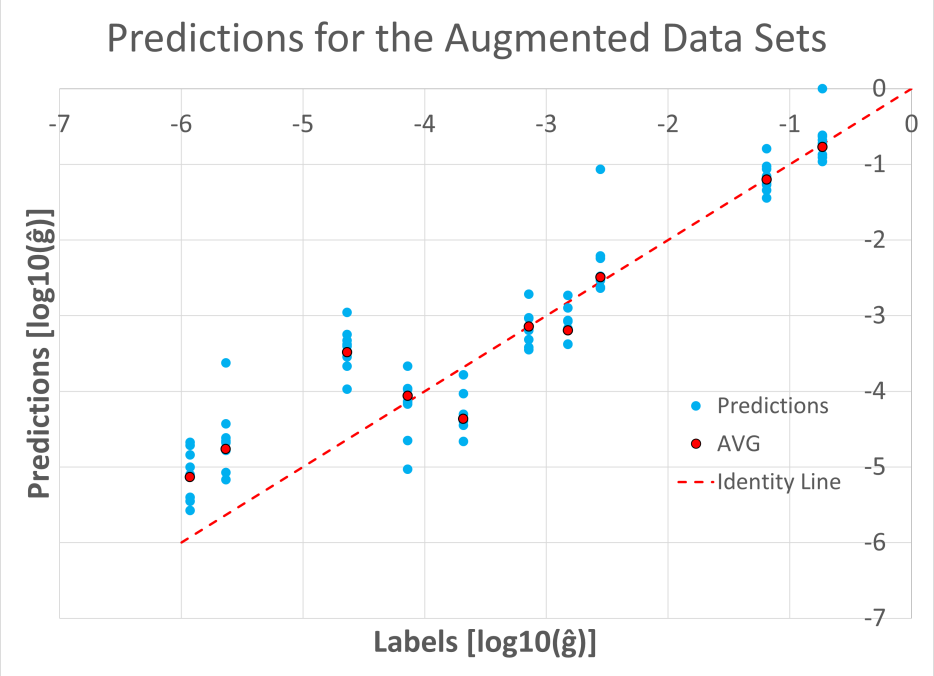


Figure 6.8: The plot shows the predictions (blue) for 10 of the 5000 samples in each data set, and the average of the predictions (red). The predictions were done by model 2DM50k.

6.6.2 3D CNN - Original vs Augmented

Table 6.10 compares the $|RE|$ of the predictions for the original image and the average $|RE|$ of the augmented images. It also includes trendline characteristics and MAE of the predictions. As for the 2D predictions, the average of the augmentated data sets lead to a drastic decrease in MAE. It also has a generally better $|RE|$. The only asset where the predictions for the original images is better is the minimum $|RE|$. For the trendlines, the average of the predictions for the augmented images has a better precision, R^2, while the accuracy, m, is somewhat lower than the slope for the original images.

From Table 6.11 it can be seen that the $|RE|$ and RE follows somewhat the same trend as for 2D in Table 6.9. The error of the underpredictions for **High** labels have increased and the error of the overpredictions for **Low** labels. In addition, the spread is reduced. For $|RE|$, the **Rest** indicates an improved predictive ability for the majority of the labels. This indicates that that this way of post-processing improves the general predictions of the model.

	3DM50k	
MAE	**Original Images**	**Avg Augmented Images**
MAE	0.3883	0.3482
|RE|	**Original Images**	**Avg Augmented Images**
Average	13.39 %	12.14 %
Median	10.25 %	9.25 %
Max	1294.46 %	1244.15 %
Min	0.0002 %	0.0017 %
Trendline	**Original Images**	**Avg Augmented Images**
R^2	0.7785	0.8212
m	0.8594	0.8480

Table 6.10: Comparison between the predictions of the original test set of 5000 images, and the average of the predictions of the augmented images. The predictions were done by model 3DM50k.

	Comparing the Results of Original Data vs Augmented Data					
3D	**Original Image**			**Avg Augmented Images**		
|RE|	**High**	**Low**	**Rest**	**High**	**Low**	**Rest**
Average	28.84 %	15.34 %	12.43 %	27.42 %	15.01 %	11.14 %
Median	14.93 %	13.84 %	10.00 %	13.93 %	14.33 %	8.87 %
Max	1294.46 %	51.19 %	77.80 %	1244.15 %	47.22 %	73.85 %
Min	0.0002 %	0.4390 %	0.0035 %	0.0242 %	0.0615 %	0.0017 %
RE	**High**	**Low**	**Rest**	**High**	**Low**	**Rest**
Average	18.09 %	-14.86 %	4.88 %	21.58 %	-14.75 %	5.44 %
Median	6.75 %	-13.84 %	4.44 %	9.08 %	-14.33 %	5.02 %
Max	1294.46 %	9.87 %	77.80 %	1244.15 %	7.27%	73.85 %
Min	-40.15 %	-51.19 %	-56.78 %	-24.43 %	-47.22 %	-56.85 %

Table 6.11: Comparing the $|RE|$ and RE for different parts of the distribution curve. The predictions were done by model 3DM50k.

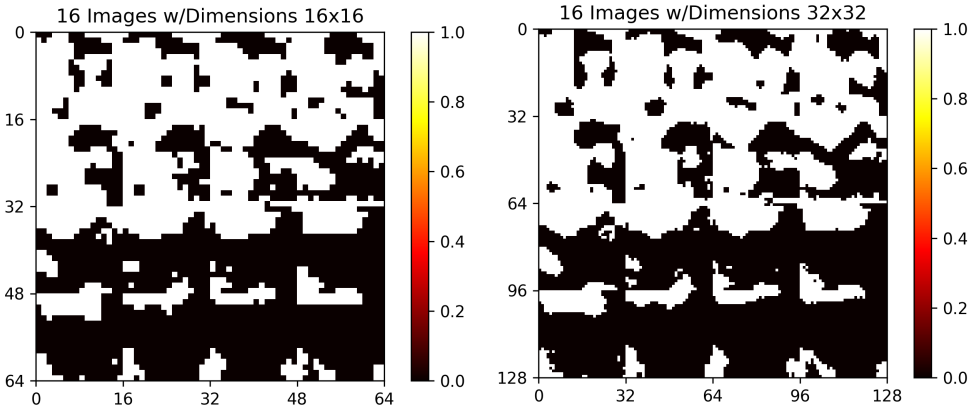
6.7 Increased Image Resolution

It is hard to say if the predictions of the CNN models are based on anything other than the amount of pore in the images. From Eq. 3.10, it is clear that the size and shape of the pore impact the hydraulic conductivity. If the CNN model is aware of this correlation, it would possibly be able to give better predictions if the amount of details about pore shape is increased.

Until now, all the models have been trained using cross section images with a dimension of 16×16 pixels. This means that the 2D images are a total of 64 × 64 pixels, while the 3D images contain 16 images of 16 × 16 pixels. It is conceivable that the models would yield better predictions if they had been trained on images with a higher image resolution, i.e., more detailed images. Fig. 6.9 shows that detail is lost due to the low resolution. The majority of the lost details concern the shape of the pores. If the model predicts based on the amount of grain vs. pore ratio, increasing the resolution will probably not make any difference.

6.7.1 2D CNN Increased Image Resolution

Two 2D CNN models were trained using cross section images with a resolution of 32×32. One of the models used a training set of 5000 samples and one model used a training set of 50000 samples. The models are named 2DM5kRes and 2DM50kRes, respectively. Table 6.12 compares the *MAE*, the |*RE*| and the trend-line of the predictions from 2DM5k and 2DM5kRes.



(a) Image used as input data for the 2D CNN model. The image consists of 16 images with the dimensions 16×16 pixels.

(b) Image used as input data for the 2D CNN model. The image consists of 16 images with the dimensions 32×32 pixels.

Figure 6.9: Two images illustrating the differences in image resolution.

According to the *MAE*, both 2DM5kRes and 2DM50kRes have better predictive abilities than the models 2DM5k and 2DM50k, respectively. For the 5k-models, average and median $|RE|$ shows that 2DM5kRes has generally better predictions. Its maximum and minimum $|RE|$ have increased, which may be the reason why its trendline is worse than the trendline of model 2DM5k. For the 50k-models, both the average $|RE|$ and R^2 are as good as equal. Model 2DM50kRes shows that increasing the resolution has lead to a decreased maximum $|RE|$ but an increased minimum. 2DM50k still has the best trendline, while 2DM50kRes has the best $|RE|$-statistics.

Increasing the cross section resolution has lead to a decreased *MAE*, but it has surely not provided any clear signs that indicate a major improvement. The improvement that are found could be the result of a more precise pore vs grain ratio in the images of higher resolution.

	2D CNN - Resolution Comparison				
MAE	**2DM5k**	**2DM5kRes**		**2DM50k**	**2DM50kRes**
MAE	0.4066	0.3949		0.3800	0.3779
|RE|	**2DM5k**	**2DM5kRes**		**2DM50k**	**2DM50kRes**
Average	14.41 %	14.11 %		13.21 %	13.02 %
Median	11.41 %	10.86 %		10.14 %	10.12 %
Max	1357.49 %	1701.82 %		1568.65 %	1286.69 %
Min	0.0039 %	0.0063 %		0.0017 %	0.0101 %
Trendline	**2DM5k**	**2DM5kRes**		**2DM50k**	**2DM50kRes**
R^2	0.7738	0.7640		0.7870	0.7867
m	0.8244	0.7829		0.8557	0.8481

Table 6.12: Comparing the $|RE|$ and trendline characteristics for the predictions of model 2DM5k and 2DM5kRes, and 2DM50k and 2DM50kRes.

6.7.2 3D CNN Increased Image Resolution

One 3D CNN model was trained using cross section images with the resolution 32×32. It was trained using 5000 images, and is therefore named 3DM5kRes. Training a 3D CNN model proved to be a lot more time consuming than training a 2D model. The model 2DM5kRes spent 5 days of hyperparameter and model tuning, whilst 3DM5kRes spent 20 days. 2DM50kRes spent six weeks, and the model that was supposed to be named 3D50kRes did not finish in time to be included in this thesis. Therefore, Table 6.13 compares the results of the two models 3DM5k and 3DM5kRes.

	3D CNN - Resolution Comparison	
MAE	**3DM5k**	**3DM5kRes**
MAE	0.3939	0.4581
|RE|	**3DM5k**	**3DM5kRes**
Average	13.87 %	16.19 %
Median	10.82 %	13.12 %
Max	1633.15 %	988.58 %
Min	0.0118 %	0.0098 %
Trendline	**3DM5k**	**3DM5kRes**
R^2	0.7697	0.7854
m	0.8071	0.9095

Table 6.13: Comparing the $|RE|$ and trendline characteristics for the predictions of model 3DM5k and 3DM5kRes.

Let us start with the error function that the models were trained to minimize, *MAE*. For model 3DM5k, $MAE = 0.3939$, and for model 3DM5kRes, $MAE = 0.4581$. This is a rather high increase. The increase of *MAE* is also reflected in the average and median $|RE|$ of model 3DM5kRes. Its trendline has improved a lot. Both R^2 and m is way better than those of 3DM5k. This could be due to the reduced maximum and minimum $|RE|$, which also indicate that the spread of the model is drastically reduced.

In Table 6.14, statistics regarding the $|RE|$ and RE are listed. From the minimum and maximum RE, it is clear that increased image resolution helps the model with reducing underpredicting, i.e., reducing maximum RE. This leads to overall more precise predictions. That said, the general error, both RE and $|RE|$ is worse than for model 3DM5k. This indicates that the precision, R^2, is improved, but the general error for each prediction has increased. This might be what is reflected in the *MAE*. It would have been interesting to see the results of model 3DM50kRes. As discussed before, increasing the amount of training data tend to improve the model. It is therefore possible that model 3DM50kRes would be able to lower the *MAE* and at the same time keep the R^2 high.

	Comparing the Results of Increased Image Resolution					
3D	**3DM5k**			**3DM5kRes**		
|RE|	**High**	**Low**	**Rest**	**High**	**Low**	**Rest**
Average	26.47 %	13.94 %	13.17 %	23.33 %	16.05 %	15.80 %
Median	13.32 %	11.82 %	10.73 %	13.97 %	12.72 %	13.10 %
Max	1633.15 %	107.21%	182.47 %	988.58 %	67.24 %	109.72 %
Min	0.2300 %	0.0615 %	0.0118 %	0.0217 %	0.0480 %	0.0098 %
RE	**High**	**Low**	**Rest**	**High**	**Low**	**Rest**
Average	15.36 %	5.90%	3.31 %	16.64 %	12.02 %	10.06 %
Median	2.66 %	5.34 %	2.64 %	8.45 %	10.29 %	9.54 %
Max	1633.15 %	107.21 %	182.47 %	988.58 %	67.24%	109.72 %
Min	-42.85 %	-32.73 %	-60.20 %	-33.56 %	-33.12 %	-61.05 %

Table 6.14: Comparing the $|RE|$ and RE for different parts of the distribution curve. The predictions were done by model 3DM5k and 3DM5kRes.

6.8 Training A Model on a Wider Distribution

Until now, the models have been trained using data sets with a similar distribution as the entire data set of labels. What seems to be recurring is that the models lack training when it comes to the highest and lowest labels. Labels above the 95th percentile or below the 5th percentile have a much higher $|RE|$ than the rest of the labels for all models. A solution to this problem could be to train a model using a data set with a different distribution curve. A higher number of high and low labels could help build a model better suited for predicting the whole specter of data samples. The question is whether this happens at the expense of the model's predictive skills for the majority of the labels. The model that will be presented now has been trained using a training data set of 50000 samples. In this set, 10000 samples are above the 90th percentile, 10000 samples are below the 10th percentile, and 30000 samples have a label between the 10th and 90th percentile. A comparison between the distribution of the training set and the distribution for the complete data set is shown in Fig. 6.10.

The model trained using the data set with the orange distribution curve in Fig. 6.10 is named 2DM50kDist. From the *MAE* presented in Table 6.15, it seems like the overall predictive skill of the model is decreased. Concerning the $|RE|$, the average and median have increased. The minimum $|RE|$ has also increased, while the maximum has decreased. The trendline for the predictions of model 2DM50kDist is way better than that for model 2DM50k. It shows signs of a more narrow spread and also a reduced underprediction for high labels and reduced overprediction for low labels. In Table 6.16, statistics about the $|RE|$ and RE for the two models are found. What seems to be the reason for the good m of the

trendline is the rather low median $|RE|$ for **High** labels, which eventually affect the trendline by tilting it upwards. The spread of the predictions is narrowed in, especially for the **High** labels, but also for the **Low**. For the **Rest**, the spread is actually wider than for the 2DM50k model. The results have thus improved for the samples below the 5th percentile and above the 95th percentile, but it may be at the expense of the overall predictive abilities of the model, as the error for the **Rest** has increased. Why this happens is not clear. It could be because the specter of labels is too wide for the model to be able to yield good predictions for the whole specter. It could also be because the model has been exposed to significantly fewer of the **Rest**-category samples than 2DM50k and therefore lack training. It is anyway clear that the model is able to yield good predictions for the **High** labels. The predictions for the **Low** labels have also clearly improved, but the result for the **High** is outstanding.

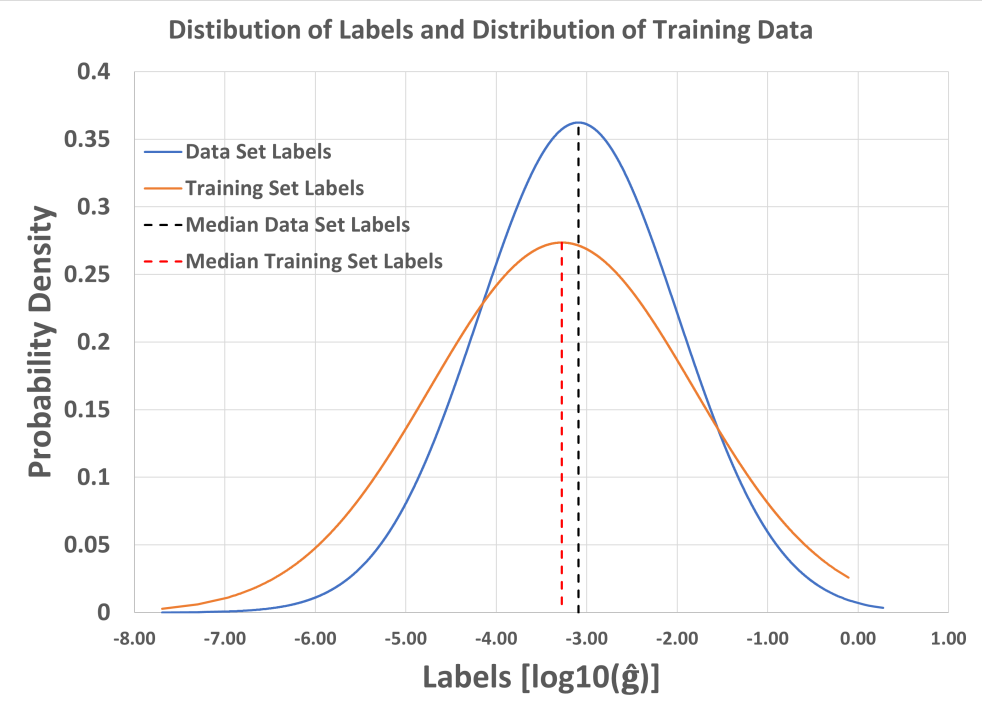


Figure 6.10: A distribution plot showing the label-distribution for the full data set (blue) of the rock sample. The distribution of the training set (orange) clearly has a wider distribution (higher standard deviation).

2D	**Different Training Set Distributions**	
MAE	**2DM50k**	**2DM50kDist**
MAE	0.3800	0.4045
|RE|	**2DM50k**	**2DM50kDist**
Average	13.21 %	13.45 %
Median	10.14 %	10.31 %
Max	1568.65 %	1346.59 %
Min	0.0017 %	0.0038 %
Trendline	**2DM50k**	**2DM50kDist**
R^2	0.7870	0.8177
m	0.8557	0.8951

Table 6.15: 2DM50k refers to the model trained on a data set with the distribution of the blue line in Fig. 6.10. Model 2DM50kDist was trained using a training set with the distribution of the orange line. Both models were trained using 50000 2D images.

	Comparison of Different Training Set Distributions					
2D	**2DM50k**			**2DM50kDist**		
|RE|	**High**	**Low**	**Rest**	**High**	**Low**	**Rest**
Average	31.39 %	15.47 %	12.08 %	11.74 %	14.16 %	13.51 %
Median	13.71 %	14.25 %	9.71 %	0.97 %	13.45 %	10.95 %
Max	1568.65 %	46.12 %	77.77 %	1346.59 %	44.53 %	77.67 %
Min	0.1913 %	0.3876 %	0.0017 %	0.0038 %	0.1111 %	0.0046 %
RE	**High**	**Low**	**Rest**	**High**	**Low**	**Rest**
Average	25.75 %	-14.82 %	4.33 %	11.33 %	-12.89 %	6.14 %
Median	10.63 %	-13.99 %	3.96 %	0.80 %	-13.24 %	5.57 %
Max	1568.65 %	20.55 %	77.77 %	1346.59 %	15.77 %	77.67 %
Min	-37.03 %	-46.12 %	-62.73 %	-2.83 %	-44.53 %	-66.82 %

Table 6.16: Comparison of the $|RE|$ and RE of the models 2DM50k and 2DM50kDist for three different parts of the test set. As for previous tables, **High** refers to labels above the 95th percentile, **Low** refers to the labels below the 5th percentile, and **Rest** refers to the labels between the 5th and 95th percentile.

2D	**Single Model vs Three Models**	
MAE	**2DM50k**	**2DM5kComb**
MAE	0.3800	0.3497
|RE|	**2DM50k**	**2DM5kComb**
Average	13.21 %	11.33 %
Median	10.14 %	9.20 %
Max	1568.65 %	72.20 %
Min	0.0017 %	0.0002 %
Trendline	**2DM50k**	**2DM5kComb**
R^2	0.787	0.8207
m	0.8557	0.8581

Table 6.17: The result of using three specialized models vs a single model. The three models were trained using 5000 training samples each. The single model was trained using 50000 training samples.

6.9 Combining Three Specialized Models

All the models introduced in the previous sections have one thing in common: they struggle with giving good predictions for the whole specter of labels, especially for the ones referred to as **High** and **Low**. To improve the predictive abilities for high and low labels, a solution could be to train three different models. One model will be trained solely using labels above the 90th percentile. One model will be trained using labels within the 10th percentile, and one will be trained using labels between the 10th and the 90th percentile. All three models will be used to predict the same data set.

The three models were trained using 5000 2D training samples each. After training, the models predicted the part of the test set 6.4 that was within their domain. The combination of the three models will be referred to as 2DM5kComb. In Table 6.17, the *MAE*, |*RE*| and trendline for the predictions of 2DM5kComb are compared to the predictions of model 2DM50k.

First of all, the *MAE* of model 2DM5kComb3 is significantly lower than the *MAE* of model 2DM50k. This tells us that the error for most of its predictions is low. The trendline is also reflecting low errors for the high and low labels, as well as a more narrow spread. All the assets for |*RE*| in the table are better than for model 2DM50k. Taking a look at Table 6.18, it can be observed that the error of the **High** and **Low** labels are significantly improved. The spread is reduced, and the median and average are drastically decreased.

Concerning the |*RE*| in the **Rest**-category, model 2DM50k has a better me-

dian and almost the same average. This is quite surprising since the model in 2DM5kComb used for predicting this category was, as mentioned, trained using 5000 training samples. This indicates that the predictive skill of model 2DM50k for the **Rest**-category is negatively affected by the **High** and **Low** labels during training. Theoretically, since model 2DM50k has been exposed to 45000 **Rest**-labels (from the distribution of the labels), which is over 40000 samples more than 2DM5kComb, it should be able to give better predictions. It could be that since it also has to tune its weights to try and fit the extreme cases as well, it decreases its predictive skills for most of the samples.

Figure 6.11 shows a crossplot of the $|RE|$ for the predictions of the two models. For the labels lower than -4.5, model 2DM5kComb has a significantly lower error, that seems to increase around $log10[\hat{g}] = -5.5$. The error for the two lowest samples is actually way higher for model 2DM5kComb. For labels above -1.5, most predictions have a rather low error for model 2DM5kComb. For model 2DM50k, the error seems to increase as the value of the label increase. The error of the labels between -4.5 and -1.5 seems to have no distinct difference.

One of the limitations of this model is clear: When the label of a sample is unknown, it is also unknown which model to use. One could use all three models to predict the same data set of unknown samples. The problem is that the models do not leave any clue of which of the model's prediction to use. The models are, as mentioned, built using the Keras packages in Python. Keras does not have a function for returning a confidence score with the predictions for regressors. A confidence score would be great for deciding which of the models' predictions to use. Li (2018) writes in *Towards Data Science* that a confidence score is the one thing that the Keras-library is lacking (Li 2018). Though a confidence score would be of great use, in this case, it is not given that the models would be able to give a neutral evaluation of their ability to predict. It is also suboptimal to have three models instead of one.

	Comparing the Results of the Models 2DM50k and 2DM5kComb					
2D	**2DM50k**			**2DM5kComb**		
|RE|	**High**	**Low**	**Rest**	**High**	**Low**	**Rest**
Average	31.39 %	15.47 %	12.08 %	4.74 %	5.06 %	12.04 %
Median	13.71 %	14.25 %	9.71 %	3.16 %	4.11 %	9.96 %
Max	1568.65 %	46.12 %	77.77 %	64.39 %	23.97 %	72.20 %
Min	0.1913 %	0.3876 %	0.0017 %	0.0141 %	0.0132 %	0.0002 %
RE	**High**	**Low**	**Rest**	**High**	**Low**	**Rest**
Average	25.75 %	-14.82 %	4.33 %	0.57 %	-1.52 %	1.81 %
Median	10.63 %	-13.99 %	3.96 %	-0.44 %	-0.44 %	1.28 %
Max	1568.65 %	20.55 %	77.77 %	64.39 %	13.46 %	72.20 %
Min	-37.03 %	-46.12 %	-62.73 %	-16.71 %	-23.97 %	-65.03 %

Table 6.18: Comparison of model 2DM50k and model 2DM5kComb. **High** and **Low** still refers to the 95th and 5th percentile.

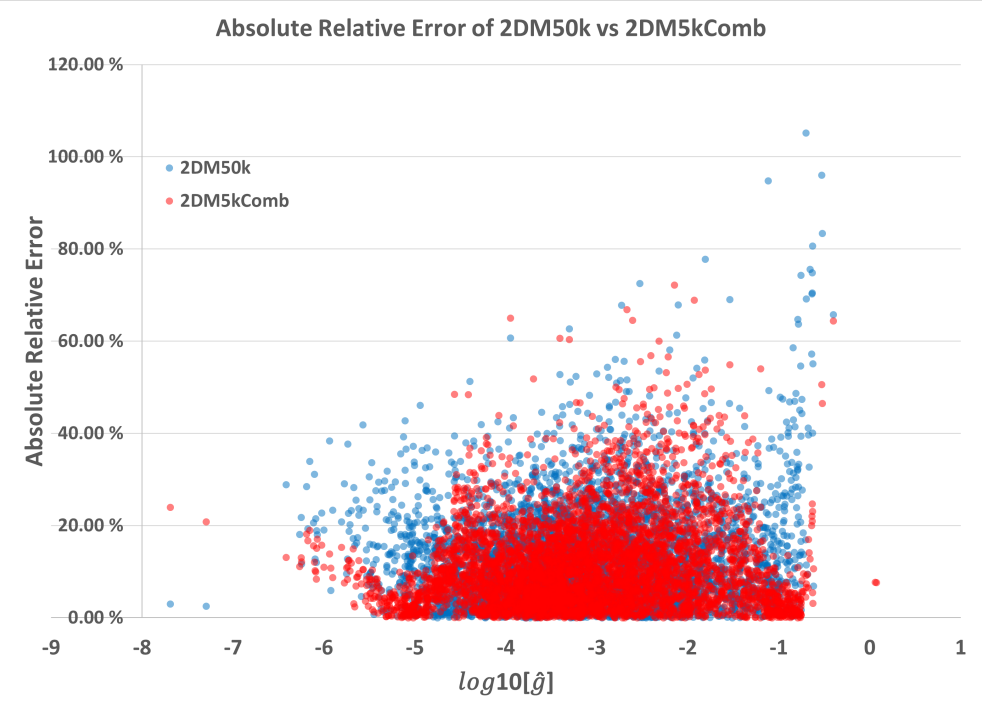


Figure 6.11: Plot showing the $|RE|$ of a prediction to the corresponding label. The blue scatters are the $|RE|$ of model 2DM50k. Two of the predictions for this model is left out of the plot, due to an error of over 1000%. The red scatters are the $|RE|$ for the predictions of model 2DM5kComb. The scatters have a transparency of 50%.

6.10 CNN Model Comparison - Summary

The different CNN models have now been introduced, compared and discussed. Table 6.19 compares results of all the CNN models introduced. Model 2DM5kComb was left out of this section, since that model is not actually possible to use without a confidence score.

In general, it seems to be a fair assumption that increasing the amount of training data gives a better model. Model 3DM25k and 3DM50k is an exception here, but for the other models, this seems to be true. The exact reason why the increased amount of training data impaired the model is hard to say. One explanation could be, as suggested, that the model adapted to the extreme cases to a higher degree, and therefore its predictive skill was reduced for the majority of the samples. For some models, 3D CNN gives a better result than 2D. An exception is the model 3DM50k and 3DM5kRes. Both models was inferior to their 2D counterparts. Unfortunately, only four of the six 3D models finished in time to be included in this thesis. It would have been interesting to see the predictions from model 3DM50kRes, as we now can assume that it would have been an improvement of model 3DM5kRes.

Making a CNN model predict multiple sets of augmented data also seems to enhance the overall precision of its predictions. The only model that does not seem to benefit from this is model 3DM5kRes. The general error for this model has increased after predicting the augmented images. The only improvement after augmentation is the R^2, which is the parameter that was aimed to improve when this introducing augmentation.

6.11 CNN Predictions To Supplement a Network Model

As introduced in Chapter 2, a network model is a common way to estimate flow properties in porous media. During this process, the effective conductance of the links and corresponding node pairs are calculated using the Equations 3.11 and 3.10. In this section, the result of running the code using the calculated hydraulic conductances will be compared to the result of running the code using the predictions as hydraulic conductances.

A code for solving Kirchhoff's equations was used for calculating the permeability of the network models. Flow calculations were done with an inlet pressure, $p_{in} = 2Pa$, and an outlet pressure $p_{out} = 1Pa$. Fluid viscosity was set to $\mu = 1cP$.

6.11.1 Results for the Traditional Network Model

Running the solver using the `link` and the `node .dat` files introduced in Section 4.2 estimated that the network has a permeability of $1834.5mD$.

6.11.2 Results for the CNN Supplemented Network Model

Solving Kirchhoff's equations for a network of this size demanded a lot more computational time than expected. Therefore, only the predictions of model 2DM50k was used. A distribution plot for the predictions and labels of the full data set is shown in Fig. 6.12. The distribution is moved slightly to the left. It is therefore likely that the estimated permeability will be lower for the model using the predictions. As mentioned in Section 5.2, when generating images, some links are ignored. In total, 17027 links were excluded, where 3874 are inlet- or outlet nodes. The conductivity of these links are calculated as usual. The hydraulic conductivity of the remaining 300786 links were predicted and used for estimation of flow properties. Image augmentation was not applied, due to the vast amount of storage capacity needed for storing 8 sets of images. The predictions are log_{10}-values of the spatially-scaled hydraulic conductivities of the links, and was therefore scaled back to their exponential form by applying the scaling process in reverse.

The estimated permeability after supplying the network with 300786 predictions was $1618.9mD$.

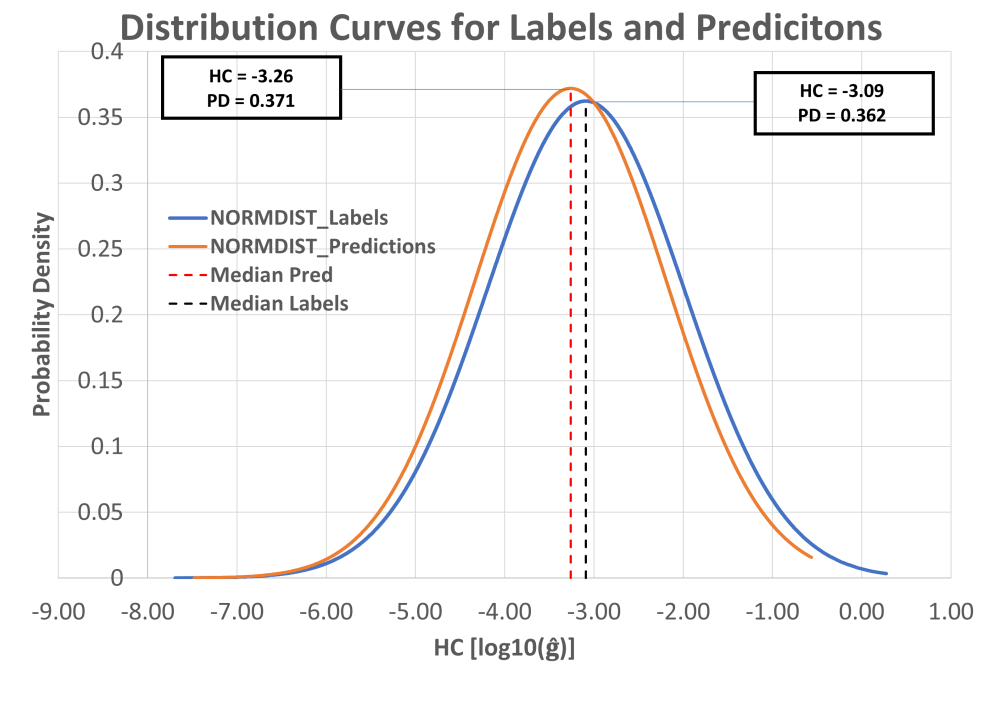


Figure 6.12: The distribution curves of the labels in the data set (blue) and the predictions (orange).

6.11.3 Discussion - Permeability Estimation

The permeability for the traditional network model will be referred to as k_NM and the CNN supplemented network will be referred to as k_CNN.

It is hard to say to which extent the k_CNN is influenced by the links that were ignored when generating images. The conductivity for these links are, as mentioned, the same for both network models. As these links make up a total of less than 5 % of the links in the network. Still, this includes all the input and output links.

It is worth mentioning that the estimations done by the network model is not the ground truth. Both the calculated hydraulic conductances, and the link and node properties, are a result of numerical simplifications. Therefore the estimated permeability k_NM carries some natural insecurities. The estimation of k_CNN also carries a lot of insecurities. Not only is it a result of predictions that has a varying degree of error, but it is also a result of predictions from a CNN model that was trained on numerical estimations of a spatial property. Numerical estimations are never hundred percent accurate, and therefore they can not be treated as the ground truth.

The difference between k_NM and k_CNN is small, and there is no clear way to distinguish which of the estimations that is least reliable. It would be interesting to see what the estimated permeability would be if the network model was supplemented with predictions from the other CNN models.

	2D Models						3D Models			
Model	**2DM5k**	**2DM25k**	**2DM50k**	**2DM5kRes**	**2DM50kRes**	**2DM50kDist**	**3DM5k**	**3DM25k**	**3DM50k**	**3DM5kRes**
MAE	0.4066	0.3984	0.3800	0.3949	0.3779	0.4045	0.3939	0.3577	0.3883	0.4581
R^2	0.7738	0.7902	0.7870	0.7640	0.7867	0.7786	0.7697	0.8001	0.7785	0.7854
m	0.8244	0.8717	0.8557	0.7829	0.8481	0.9029	0.8071	0.8226	0.8594	0.9095
AVG $\lvert$**RE**$\rvert$[%]	14.41	14.01	13.21	14.11	13.02	13.45	13.87	12.26	13.39	16.19
Median $\lvert RE \rvert$**[%]**	11.41	10.73	10.14	10.86	10.12	10.31	10.82	9.46	10.25	13.12
	AFTER AUGMENTATION									
MAE	0.3866	0.3845	0.3490	0.3755	0.3358	0.3762	0.3668	0.3352	0.3482	0.4583
R^2	0.8067	0.8165	0.8207	0.7982	0.8267	0.8177	0.8076	0.8205	0.8212	0.8187
m	0.8033	0.8653	0.8506	0.7688	0.8465	0.8951	0.7823	0.8054	0.8480	0.9006
AVG $\lvert RE \rvert$**[%]**	14.02	13.74	12.34	13.80	11.71	13.07	13.38	11.52	12.14	16.37
Median $\lvert RE \rvert$**[%]**	11.21	10.68	9.42	10.62	8.97	10.22	10.21	8.89	9.25	14.11

Table 6.19: Comparing all the CNN models before and after augmentation.

Chapter 7

Conclusion

The main goal of this thesis was to investigate if a CNN model can give good predictions for the hydraulic conductance of interconnected pores. CNN models have a proven history for problems concerning images. For this purpose, images of the subvolume between each connected node pair were generated and structured so that the CNN model could extract useful information from them. The method used for generating and structuring the images may have been suboptimal for some of the links. The effective hydraulic conductance of the links and their corresponding node pair was calculated and scaled based on the spatial extent of its associated subvolume.

Multiple CNN models were built and compared. The models differed regarding the amount of training data, image resolution, different image structuring (2D or 3D), and training data distribution. The models were compared regarding the mean absolute error, the relative error, and the trendline characteristics. The effect of image augmentation for the purpose of post-processing was also investigated.

It is hard to determine which of the models that gave the best predictions. An overall result is that increasing the amount of training data will possibly also increase the model's predictive abilities. Image augmentation used as described in this thesis gave the predictions a higher precision at the expense of reduced accuracy. There are also some differences between 2D and 3D CNN, as the 2D models seem to benefit from increased image resolution, while it appears to impair the 3D model.

The result of supplying a network model with the predictions from one of the CNN models gave similar permeability as the original network model. This indicates that CNN can be a valuable tool for generating network models from 3D grids.

Chapter 8

Future Work

The permeability estimation indicates that the CNN model can give good predictions overall. However, there is a need for further improvement concerning the individual predictions.

A possible solution could be by, e.g., developing a less general way of generating data set images. A more specialized method, especially for the samples with the lowest and highest labels, could make the task easier for the CNN model. For the samples with the lowest labels, generating images that only include the current pores and throat would probably ease the predicting of such samples. For the samples with the highest labels, generating images of the entire cross sections of the whole pore elements could probably also be an improvement.

The images in this thesis neglected the clay content. Building a model using images that include the clay could be interesting, because the unresolved microporosity of clay indeed affects the hydraulic conductance of a rock. These images can be generated in the same way as the images used in this thesis.

The structuring of the cylindrical images could also be investigated further, as this method allows for additional rotation angles. An increased number of rotations may provide an increased number of augmented data sets, which have proved to beneficial for increasing the precision of the predictions.

This project aimed to investigate the possibility of using CNN models to predict the hydraulic conductance between interconnected pores. The method suggested in this thesis is not limited to this purpose, and could also be used to predict other properties of porous media, e.g., electric conductance.

Bibliography

[1] M. J. Blunt, B. Bijeljic, H. Dong, O. Gharbi, S. Iglauer, P. Mostaghimi, A. Paluszny and C. Pentland, 'Pore-scale imaging and modelling,' en, *Advances in Water Resources*, 35th Year Anniversary Issue, vol. 51, pp. 197–216, Jan. 2013, ISSN: 0309-1708. DOI: 10.1016/j.advwatres.2012.03.003. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0309170812000528 (visited on 08/11/2021).

[2] I. Fatt, 'The Network Model of Porous Media,' en, *Transactions of the AIME*, vol. 207, no. 01, pp. 144–181, Dec. 1956, ISSN: 0081-1696. DOI: 10.2118/574-G. [Online]. Available: https://onepetro.org/TRANS/article/207/01/144/160841/The-Network-Model-of-Porous-Media (visited on 09/11/2021).

[3] Y. Imamverdiyev and L. Sukhostat, *Lithological facies classification using deep convolutional neural network | Elsevier Enhanced Reader*, en, Nov. 2018. DOI: 10.1016/j.petrol.2018.11.023. [Online]. Available: https://reader.elsevier.com/reader/sd/pii/S0920410518310143?token=5D3D9A7BB2B288905969DAF563739FCA90F06F166FED31D2C3E7C7112EB5194947D1089782DA3EAEE66C3BBE52DD95D6&originRegion=eu-west-1&originCreation=20220611153053 (visited on 11/06/2022).

[4] D. Wildenschild and A. P. Sheppard, 'X-ray imaging and analysis techniques for quantifying pore-scale structure and processes in subsurface porous medium systems,' en, *Advances in Water Resources*, 35th Year Anniversary Issue, vol. 51, pp. 217–246, Jan. 2013, ISSN: 0309-1708. DOI: 10.1016/j.advwatres.2012.07.018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0309170812002060 (visited on 08/11/2021).

[5] H. Andrä, N. Combaret, J. Dvorkin, E. Glatt, J. Han, M. Kabel, Y. Keehm, F. Krzikalla, M. Lee, C. Madonna, M. Marsh, T. Mukerji, E. H. Saenger, R. Sain, N. Saxena, S. Ricker, A. Wiegmann and X. Zhan, 'Digital rock physics benchmarks—Part I: Imaging and segmentation,' en, *Computers & Geosciences*, Benchmark problems, datasets and methodologies for the computational geosciences, vol. 50, pp. 25–32, Jan. 2013, ISSN: 0098-3004. DOI: 10.1016/j.cageo.2012.09.005. [Online]. Available: https://www.

sciencedirect.com/science/article/pii/S0098300412003147 (visited on 08/11/2021).

[6] Martin J. Blunt, *Multiphase flow in permeable media: a pore-scale perspective*. Cambridge University Press, 2017.

[7] C. F. Berg, O. Lopez and H. Berland, 'Industrial applications of digital rock technology,' en, *Journal of Petroleum Science and Engineering*, vol. 157, pp. 131–147, Aug. 2017, ISSN: 0920-4105. DOI: 10.1016/j.petrol.2017.06.074. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0920410517305600 (visited on 08/11/2021).

[8] Joseph M Polese, *Electron-Hole Recombination*, en, Jul. 2016. [Online]. Available: https://eng.libretexts.org/Bookshelves/Materials_Science/Supplemental_Modules_(Materials_Science)/Electronic_Properties/Electron-Hole_Recombination (visited on 13/11/2021).

[9] R. M. Sok, T. Varslot, A. Ghous, S. Latham, A. P. Sheppard and M. A. Knackstedt, 'Pore Scale Characterization of Carbonates at Multiple Scales: Integration of Microct, Bsem and Fibsem,' en, p. 12, Sep. 2009.

[10] P. H. Nadeau and A. Hurst, 'Application of backscattered electron microscopy to the quantification of clay mineral microscopy in sandstones,' *Journal of Sedimentary Petrology*, vol. 61, pp. 921–925, Jan. 1991.

[11] V. Cromwell, D. J. Kortum and D. J. Bradley, 'The Use of a Medical Computer Tomography (CT) System To Observe Multiphase Flow in Porous Media,' en, OnePetro, Sep. 1984. DOI: 10.2118/13098-MS. [Online]. Available: https://onepetro.org/SPEATCE/proceedings/84SPE/All-84SPE/SPE-13098-MS/66269 (visited on 12/11/2021).

[12] R. E. Gilliland and M. E. Coles, 'Use of CT Scanning in the Investigation of Damage to Unconsolidated Cores,' en, OnePetro, Feb. 1990. DOI: 10.2118/19408-MS. [Online]. Available: https://onepetro.org/SPEFD/proceedings/90FD/All-90FD/SPE-19408-MS/68389 (visited on 16/11/2021).

[13] M. M. Honarpour, V. Cromwell, D. Hatton and R. Satchwell, 'Reservoir Rock Descriptions Using Computed Tomography (CT),' en, OnePetro, Sep. 1985. DOI: 10.2118/14272-MS. [Online]. Available: https://onepetro.org/SPEATCE/proceedings/85SPE/All-85SPE/SPE-14272-MS/61627 (visited on 16/11/2021).

[14] A. O. Hove, J. K. Ringen and P. A. Read, 'Visualization of laboratory corefloods with the aid of computerized tomography of X-rays,' English, *SPE (Society of Petroleum Engineers) Reserv. Eng.; (United States)*, vol. 2:2, May 1987, Institution: Statoil A/S. DOI: 10.2118/13654-PA. [Online]. Available: https://www.osti.gov/biblio/6503776 (visited on 16/11/2021).

[15] S. L. Wellington and H. J. Vinegar, 'X-Ray Computerized Tomography,' en, *Journal of Petroleum Technology*, p. 14, 1987.

[16] B. P. Flannery, H. W. Deckman, W. G. Roberge and K. L. D'Amico, 'Three-Dimensional X-Ray Microtomography,' *Science*, vol. 237, no. 4821, pp. 1439–1444, Sep. 1987, Publisher: American Association for the Advancement of Science. DOI: 10.1126/science.237.4821.1439. [Online]. Available: https://www.science.org/doi/10.1126/science.237.4821.1439 (visited on 12/11/2021).

[17] S. Berg, H. Ott, S. A. Klapp, A. Schwing, R. Neiteler, N. Brussee, A. Makurat, L. Leu, F. Enzmann, J.-O. Schwarz, M. Kersten, S. Irvine and M. Stampanoni, 'Real-time 3D imaging of Haines jumps in porous media flow,' *Proceedings of the National Academy of Sciences of the United States of America*, vol. 110, no. 10, pp. 3755–3759, Mar. 2013, ISSN: 0027-8424. DOI: 10.1073/pnas.1221373110. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3593852/ (visited on 15/11/2021).

[18] S. Bakke and P.-E. Øren, '3-D Pore-Scale Modelling of Sandstones and Flow Simulations in the Pore Networks,' en, *SPE Journal*, vol. 2, no. 02, pp. 136–149, Jun. 1997, ISSN: 1086-055X, 1930-0220. DOI: 10.2118/35479-PA. [Online]. Available: https://onepetro.org/SJ/article/2/02/136/108049/3-D-Pore-Scale-Modelling-of-Sandstones-and-Flow (visited on 30/08/2021).

[19] R. Hilfer and C. Manwart, 'Permeability and conductivity for reconstruction models of porous media,' *Physical review. E, Statistical, nonlinear, and soft matter physics*, vol. 64, p. 021 304, Sep. 2001. DOI: 10.1103/PhysRevE.64.021304.

[20] Øren, Stig Bakke and O. J. Arntzen, 'Extending Predictive Capabilities to Network Models,' en, *SPE Journal*, p. 13, 1998.

[21] M. Sahimi, *Flow and Transport in Porous Media and Fractured Rock: From Classical Methods to Modern Approaches, 2nd Edition*, English. John Wiley & Sons, Jun. 2011.

[22] P. H. Valvatne and M. J. Blunt, 'Predictive pore-scale modeling of two-phase flow in mixed wet media,' en, *Water Resources Research*, vol. 40, no. 7, 2004, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1029/2003WR002627, ISSN: 1944-7973. DOI: 10.1029/2003WR002627. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1029/2003WR002627 (visited on 11/11/2021).

[23] I. Chatzis and F. Dullien, 'Modelling Pore Structure By 2-D And 3-D Networks With ApplicationTo Sandstones,' en, *Journal of Canadian Petroleum Technology*, vol. 16, no. 01, Jan. 1977, ISSN: 0021-9487. DOI: 10.2118/77-01-09. [Online]. Available: https://onepetro.org/JCPT/article/doi/10.2118/77-01-09/30978/Modelling-Pore-Structure-By-2-D-And-3-D-Networks (visited on 09/11/2021).

[24] H. L. Frisch, J. M. Hammersley and D. J. A. Welsh, 'Monte Carlo Estimates of Percolation Probabilities for Various Lattices,' en, *Physical Review*, vol. 126, no. 3, pp. 949–951, May 1962, ISSN: 0031-899X. DOI: 10.1103/PhysRev.126.949. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRev.126.949 (visited on 19/12/2021).

[25] S. R. Broadbent and J. M. Hammersley, 'Percolation processes: I. Crystals and mazes,' en, *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 53, no. 3, pp. 629–641, Jul. 1957, Publisher: Cambridge University Press, ISSN: 1469-8064, 0305-0041. DOI: 10.1017/S0305004100032680. [Online]. Available: https://www.cambridge.org/core/journals/mathematical-proceedings-of-the-cambridge-philosophical-society/article/percolation-processes/C00CC4943F48228F8AC8031092FE84EC (visited on 10/11/2021).

[26] S. Bryant and M. Blunt, 'Prediction of relative permeability in simple porous media,' en, *Physical Review A*, vol. 46, no. 4, pp. 2004–2011, Aug. 1992, ISSN: 1050-2947, 1094-1622. DOI: 10.1103/PhysRevA.46.2004. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.46.2004 (visited on 11/11/2021).

[27] M. J. Blunt, M. D. Jackson, M. Piri and P. H. Valvatne, 'Detailed physics, predictive capabilities and macroscopic consequences for pore-network models of multiphase flow,' en, *Advances in Water Resources*, vol. 25, no. 8, pp. 1069–1089, Aug. 2002, ISSN: 0309-1708. DOI: 10.1016/S0309-1708(02)00049-0. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0309170802000490 (visited on 08/11/2021).

[28] M. Ding and A. Kantzas, 'Capillary number correlations for gas-liquid systems,' Sep. 2004.

[29] M. A. Gjennestad, M. Vassvik, S. Kjelstrup and A. Hansen, 'Stable and Efficient Time Integration of a Dynamic Pore Network Model for Two-Phase Flow in Porous Media,' en, *Frontiers in Physics*, vol. 6, p. 56, Jun. 2018, ISSN: 2296-424X. DOI: 10.3389/fphy.2018.00056. [Online]. Available: https://www.frontiersin.org/article/10.3389/fphy.2018.00056/full (visited on 11/11/2021).

[30] J. Koplik and T. Lasseter, 'Two-Phase Flow in Random Network Models of Porous Media,' en, *Society of Petroleum Engineers Journal*, vol. 25, no. 01, pp. 89–100, Feb. 1985, ISSN: 0197-7520. DOI: 10.2118/11014-PA. [Online]. Available: https://onepetro.org/spejournal/article/25/01/89/72894/Two-Phase-Flow-in-Random-Network-Models-of-Porous (visited on 11/11/2021).

[31] J. N. Kok, Egbert J. W. Boers, Walter A. Kosters and Peter van der Putten, 'Artificial Intelligence : Definition, Trends, Techniques and Cases,' en, *ARTIFICIAL INTELLIGENCE*, p. 5, 2009.

[32] J. R. Koza, F. H. Bennett, D. Andre and M. A. Keane, 'Automated Design of Both the Topology and Sizing of Analog Electrical Circuits Using Genetic Programming,' en, in *Artificial Intelligence in Design '96*, J. S. Gero and F. Sudweeks, Eds., Dordrecht: Springer Netherlands, 1996, pp. 151–170, ISBN: 978-94-009-0279-4. DOI: 10.1007/978-94-009-0279-4_9. [Online]. Available: https://doi.org/10.1007/978-94-009-0279-4_9 (visited on 17/11/2021).

[33] T. Mitchell, *Machine learning*, ser. McGraw-Hill international editions. McGraw-Hill, 1997, tex.lccn: 97007692, ISBN: 978-0-07-115467-3. [Online]. Available: https://books.google.no/books?id=EoYBngEACAAJ.

[34] A. S. Lampropoulos and G. A. Tsihrintzis, *Machine Learning Paradigms*, en, ser. Intelligent Systems Reference Library. Cham: Springer International Publishing, 2015, vol. 92, ISBN: 978-3-319-19134-8 978-3-319-19135-5. DOI: 10.1007/978-3-319-19135-5. [Online]. Available: http://link.springer.com/10.1007/978-3-319-19135-5 (visited on 12/05/2022).

[35] S. B. Kotsiantis, I. D. Zaharakis and P. E. Pintelas, 'Machine learning: A review of classification and combining techniques,' en, *Artificial Intelligence Review*, vol. 26, no. 3, pp. 159–190, Nov. 2006, ISSN: 0269-2821, 1573-7462. DOI: 10.1007/s10462-007-9052-3. [Online]. Available: http://link.springer.com/10.1007/s10462-007-9052-3 (visited on 20/12/2021).

[36] Daniel Westrich, Justin Lessler and Michele Jonsson Funk, *Propensity score estimation: Neural networks, support vector machines, decision trees (CART), and meta-classifiers as alternatives to logistic regression | Elsevier Enhanced Reader*, en, Nov. 2009. DOI: 10.1016/j.jclinepi.2009.11.020. [Online]. Available: https://reader.elsevier.com/reader/sd/pii/S0895435610001022?token=AE9296B905A10E24C3B718C57B88B7A25193E8460BB790F66397943D890C85B9777F6846D88CE256F9B4790FA7C93AF3&originRegion=eu-west-1&originCreation=20211219210353.

[37] R. Sathya and Annamma Abraham, 'Comparison of Supervised and Unsupervised Learning Algorithms for Pattern Classification,' En, 2013. [Online]. Available: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.278.5274&rep=rep1&type=pdf#page=41.

[38] H. U. Dike, Y. Zhou, K. K. Deveerasetty and Q. Wu, 'Unsupervised Learning Based On Artificial Neural Network: A Review,' in *2018 IEEE International Conference on Cyborg and Bionic Systems (CBS)*, Oct. 2018, pp. 322–327. DOI: 10.1109/CBS.2018.8612259.

[39] D. Zhang, X. Han and C. Deng, 'Review on the research and practice of deep learning and reinforcement learning in smart grids,' *CSEE Journal of Power and Energy Systems*, vol. 4, no. 3, pp. 362–370, Sep. 2018, Conference Name: CSEE Journal of Power and Energy Systems, ISSN: 2096-0042. DOI: 10.17775/CSEEJPES.2018.00520.

[40] Jia Wu, Xiu-Yun Chen, Hao Zhang, Li-Dong Xiong, Hang Lei and Si-Hao Deng, *Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimizationb*, en, ISSN: 1674-862X, Mar. 2019. DOI: 10.11989/JEST.1674-862X.80904120. [Online]. Available: https://reader.elsevier.com/reader/sd/pii/S1674862X19300047?token=16B6E7348516694D4A351DD9FC46FD4FC8F17A430FD5F8566625B283065B786E13142949349F9905C626BD1108200509&originRegion=eu-west-1&originCreation=20211220002811.

[41] Jianxin Wu, *Introduction to Convolutional Neural Networks*, English, May 2017. [Online]. Available: https://cs.nju.edu.cn/wujx/paper/CNN.pdf.

[42] S.-C. Wang, *Interdisciplinary Computing in Java Programming*, en. Boston, MA: Springer US, 2003, ISBN: 978-1-4613-5046-0 978-1-4615-0377-4. DOI: 10.1007/978-1-4615-0377-4. [Online]. Available: http://link.springer.com/10.1007/978-1-4615-0377-4 (visited on 19/12/2021).

[43] K. Shiruru, 'AN INTRODUCTION TO ARTIFICIAL NEURAL NETWORK,' *International Journal of Advance Research and Innovative Ideas in Education*, vol. 1, pp. 27–30, Sep. 2016.

[44] K. Gurney, *Introduction to Neural Networks*, en, ISBN: 9780203451519 Place: Oxford OCLC: 892785047, 1997.

[45] V. Nair and G. E. Hinton, 'Rectified Linear Units Improve Restricted Boltzmann Machines,' en, p. 8, 2010.

[46] P. Savarese and M. Maire, 'Learning Implicitly Recurrent CNNs Through Parameter Sharing,' *arXiv:1902.09701 [cs, stat]*, Mar. 2019, arXiv: 1902.09701. [Online]. Available: http://arxiv.org/abs/1902.09701 (visited on 06/12/2021).

[47] J. L. Ba, J. R. Kiros and G. E. Hinton, 'Layer Normalization,' *arXiv:1607.06450 [cs, stat]*, Jul. 2016, arXiv: 1607.06450. [Online]. Available: http://arxiv.org/abs/1607.06450 (visited on 21/12/2021).

[48] Yan Lecun, *Optimal Brain Damage*, English, 1990. [Online]. Available: http://yann.lecun.com/exdb/publis/pdf/lecun-90b.pdf.

[49] P. Molchanov, A. Mallya, S. Tyree, I. Frosio and J. Kautz, 'Importance Estimation for Neural Network Pruning,' en, in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, USA: IEEE, Jun. 2019, pp. 11 256–11 264, ISBN: 978-1-72813-293-8. DOI: 10.1109/CVPR.2019.01152. [Online]. Available: https://ieeexplore.ieee.org/document/8953464/ (visited on 09/12/2021).

[50] L. Perez and J. Wang, 'The Effectiveness of Data Augmentation in Image Classification using Deep Learning,' *arXiv:1712.04621 [cs]*, Dec. 2017, arXiv: 1712.04621. [Online]. Available: http://arxiv.org/abs/1712.04621 (visited on 20/12/2021).

[51] Baiyang Wang and Diego Klabjan, 'Regularization for Unsupervised Deep Neural Nets,' 2017.

[52] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, 'Dropout: A Simple Way to Prevent Neural Networks from Overfitting,' en, p. 30, Jun. 2014.

[53] H. Zhang, L. Zhang and Y. Jiang, 'Overfitting and Underfitting Analysis for Deep Learning Based End-to-end Communication Systems,' in *2019 11th International Conference on Wireless Communications and Signal Processing (WCSP)*, ISSN: 2472-7628, Oct. 2019, pp. 1–6. DOI: 10.1109/WCSP.2019.8927876.

[54] G. Chassagnon, M. Vakalopolou, N. Paragios and M.-P. Revel, 'Deep learning: Definition and perspectives for thoracic imaging,' en, *European Radiology*, vol. 30, no. 4, pp. 2021–2030, Apr. 2020, ISSN: 1432-1084. DOI: 10.1007/s00330-019-06564-3. [Online]. Available: https://doi.org/10.1007/s00330-019-06564-3 (visited on 16/05/2022).

[55] C. Zhang, P. Patras and H. Haddadi, 'Deep Learning in Mobile and Wireless Networking: A Survey,' *IEEE Communications Surveys Tutorials*, vol. 21, no. 3, pp. 2224–2287, 2019, Conference Name: IEEE Communications Surveys Tutorials, ISSN: 1553-877X. DOI: 10.1109/COMST.2019.2904897.

[56] K. Chawshin, C. Berg, D. Varagnolo and O. Lopez, 'Lithology classifcation of whole core CT scans using convolutional neural networks,' *SN Applied Sciences*, vol. 3, Jun. 2021. DOI: 10.1007/s42452-021-04656-8.

[57] M. Elmorsy, W. El-Dakhakhni and B. Zhao, 'Generalizable Permeability Prediction of Digital Porous Media via a Novel Multi-Scale 3D Convolutional Neural Network,' en, *Water Resources Research*, vol. 58, no. 3, e2021WR031454, 2022, _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1029/2021WR031454, ISSN: 1944-7973. DOI: 10.1029/2021WR031454. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1029/2021WR031454 (visited on 10/05/2022).

[58] K. O'Shea and R. Nash, 'An Introduction to Convolutional Neural Networks,' *arXiv:1511.08458 [cs]*, Dec. 2015, arXiv: 1511.08458. [Online]. Available: http://arxiv.org/abs/1511.08458 (visited on 03/12/2021).

[59] S. Sahoo, *Deciding optimal filter size for CNNs*, en, Nov. 2018. [Online]. Available: https://towardsdatascience.com/deciding-optimal-filter-size-for-cnns-d6f7b56f9363 (visited on 05/12/2021).

[60] Qingrong Xiong, T. G. Baychev and Andrey P. Jivkov, *Review of pore network modelling of porous media: Experimental characterisations, network constructions and applications to reactive transport*, en, Jul. 2016. DOI: 10.1016/j.jconhyd.2016.07.002. [Online]. Available: https://reader.elsevier.com/reader/sd/pii/S016977221630122X?token=80C7C453C8B29E859BFBD24CCA1627D065CDD085FCB57EB9B235B517537365908D9A1E512D06911F

7CC69C0B65BD0F04&originRegion=eu-west-1&originCreation=20211208143637 (visited on 08/12/2021).

[61] Geoffery Mason and Norman R. Morrow, *Capillary Behavior of a Perfectly Wetting Liquid in Irregular Triangular Tubes*, en, ISSN: 0021-9797, Mar. 1990. DOI: 10.1016/0021-9797(91)90321-X. [Online]. Available: https://reader.elsevier.com/reader/sd/pii/002197979190321X?token=9CE375ECCB635D9C0ED69D0AF64862FA10E0905270A7A7CAF47F7FD9DC07B2B5FE3E952E6C1FC65C3C0C1303CCB2D4F6&originRegion=eu-west-1&originCreation=20211208154645 (visited on 08/12/2021).

[62] T. Ramstad, C. F. Berg and K. Thompson, 'Pore-Scale Simulations of Single- and Two-Phase Flow in Porous Media: Approaches and Applications,' en, *Transport in Porous Media*, vol. 130, no. 1, pp. 77–104, Oct. 2019, ISSN: 0169-3913, 1573-1634. DOI: 10.1007/s11242-019-01289-9. [Online]. Available: http://link.springer.com/10.1007/s11242-019-01289-9 (visited on 30/08/2021).

[63] K. Chawshin, C. Berg, D. Varagnolo and O. Lopez, 'Automated porosity estimation using CT-scans of extracted core data,' *Computational Geosciences*, vol. 26, Jun. 2022. DOI: 10.1007/s10596-022-10143-9.

[64] L. Li, *What Keras Models Are Missing*, en, Oct. 2018. [Online]. Available: https://towardsdatascience.com/what-keras-models-are-missing-89b47cc5a4fa (visited on 10/06/2022).

Tormod Kårstad

Predicting Inter-Pore Hydraulic Conductivity Using Convolutional Neural Network

NTNU
Norwegian University of
Science and Technology