BAŞKENT UNIVERSITY
INSTITUTE OF SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER ENGINEERING
MASTER OF SCIENCE IN
COMPUTER ENGINEERING

ESTIMATION OF PERMEABILITY VALUES IN GEOTHERMAL FIELDS WITH MACHINE LEARNING METHODS

BY

ALİ BAŞER

MASTER OF SCIENCE THESIS

ANKARA – 2021

BAŞKENT UNIVERSITY
INSTITUTE OF SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER ENGINEERING
MASTER OF SCIENCE IN
COMPUTER ENGINEERING

ESTIMATION OF PERMEABILITY VALUES IN GEOTHERMAL FIELDS WITH MACHINE LEARNING METHODS

BY

ALİ BAŞER

MASTER OF SCIENCE THESIS

ADVISOR

ASSOC. PROF. MUSTAFA SERT

ANKARA – 2021

BAŞKENT UNIVERSITY

INSTITUTE OF SCIENCE AND ENGINEERING

This study, which was prepared by Ali BAŞER for the program of Computer Engineering, has been approved in partial fulfillment of the requirements for the degree of MASTER OF SCIENCE in Computer Engineering Department by the following committee.

Date of Thesis Defense: 17/08/2021

Thesis Title: Estimation of Permeability Values in Geothermal Fields with Machine Learning Methods

Examining Committee Members **Signature**

Assoc. Prof. Dr. Mehmet Feyzi AKŞAHİN, Gazi University (Chairman)

Assoc. Prof. Dr. Mustafa SERT, Başkent University (Member / Advisor)

Prof. Dr. Hamit ERDEM, Başkent University (Member)

APPROVAL

Prof. Faruk ELALDI

Director, Institute of Science and Engineering

Date: … / … / ……..

BAŞKENT ÜNİVERSİTESİ

FEN BİLİMLER ENSTİTÜSÜ

YÜKSEK LİSANS / DOKTORA TEZ ÇALIŞMASI ORİJİNALLİK RAPORU

Tarih: 27/08/2021

Öğrencinin Adı, Soyadı : Ali BAŞER

Öğrencinin Numarası : 21820003

Anabilim Dalı : Bilgisayar Mühendisliği A.B.D.

Programı : Bilgisayar Mühendisliği Tezli Y.L. Programı

Danışmanın Unvanı/Adı, Soyadı : Doç. Dr. Mustafa SERT

Tez Başlığı : Jeotermal Sahalardaki Geçirgenlik Değerlerinin Makine Öğrenmesi Yöntemleri ile Kestirimi (Estimation of Permeability Values in Geothermal Fields with Machine Learning Methods)

Yukarıda başlığı belirtilen Yüksek Lisans çalışmamın; Giriş, Ana Bölümler ve Sonuç Bölümünden oluşan, toplam 65 sayfalık kısmına ilişkin, 27/08/2021 tarihinde şahsım/tez danışmanım tarafından Turnitin adlı intihal tespit programından aşağıda belirtilen filtrelemeler uygulanarak alınmış olan orijinallik raporuna göre, tezimin benzerlik oranı % 3'dür. Uygulanan filtrelemeler:

1. Kaynakça hariç
2. Alıntılar hariç
3. Beş (5) kelimeden daha az örtüşme içeren metin kısımları hariç

"Başkent Üniversitesi Enstitüleri Tez Çalışması Orijinallik Raporu Alınması ve Kullanılması Usul ve Esaslarını" inceledim ve bu uygulama esaslarında belirtilen azami benzerlik oranlarına tez çalışmamın herhangi bir intihal içermediğini; aksinin tespit edileceği muhtemel durumda doğabilecek her türlü hukuki sorumluluğu kabul ettiğimi ve yukarıda vermiş olduğum bilgilerin doğru olduğunu beyan ederim.

Öğrenci İmzası:

ONAY

27/08/2021

Öğrenci Danışmanı

Doç. Dr. Mustafa SERT

ACKNOWLEDGEMENTS

First of all, I would like to express my gratitude to my advisor, Assoc. Prof. Mustafa SERT for his guidance, support, and valuable advice.

I am grateful to Önder Ramiz Saraçoğlu for supporting me throughout this work.

I want to express sincere gratitude to my mother Sevgi, father Ekrem, and my grandmother Göksel for their patience and warm support during my studies.

Finally, I owe special thanks to my wife Banu for her continuous support. This study would not be possible without her patience, guidance, and encouragement.

ABSTRACT

Ali BAŞER

ESTIMATION OF PERMEABILITY VALUES IN GEOTHERMAL FIELDS WITH MACHINE LEARNING METHODS

Institute Of Science And Engineering

Department of Computer Engineering

2021

Numerical modeling of geothermal fields is a very time-intensive task. Modeling the natural state of a geothermal field, where there is no production or reinjection in the field, is vital in this process. Natural state modeling is generally conducted by employing a trial and error procedure that depends on intuition in determining the rock properties to match the temperature and pressure readings. This study proposes a method for the distribution of permeability estimation in natural state modeling of geothermal fields using machine learning algorithms. In the study, firstly, a synthetic dataset is created by giving several permeability distributions to a numerical simulator called TOUGH2. Temperature and pressure outputs of the numerical simulator are then collected, and a dataset is created. Random Forest, Support Vector Regression, Multilayer Perceptron, Convolutional Neural Networks, and Transfer Learning methods are trained in this study to learn the relation between the pressure and temperature data and the distribution of permeability values in the field. The study results show that the proposed method can estimate the permeability distributions and help the geothermal field modeling process by decreasing the required time and costs.

KEYWORDS: Geothermal Modeling, Natural State Modeling, Convolutional Neural Networks, Simulation, TOUGH2

ÖZET

Ali BAŞER

JEOTERMAL SAHALARDAKİ GEÇİRGENLİK DEĞERLERİNİN MAKİNE ÖĞRENMESİ YÖNTEMLERİ İLE KESTİRİMİ

Başkent Üniversitesi Fen Bilimleri Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalı

2021

Jeotermal rezervuarların sayısal olarak modellenmesi yoğun uzmanlık gerektiren ve oldukça zaman alan bir süreçtir. Sayısal modellemenin en önemli adımlarından biri olan doğal durum modellemesinde, ilgili sahanın üretim ve reenjeksiyon gibi insan kaynaklı faaliyetlerle değiştirilmeden önceki yapısının tespiti için çalışmalar yapılır. Sahaya yönelik bilginin ve uzman deneyiminin önemli olduğu bu aşamada genellikle kayaçların ve sistemin özellikleri tahmin edilmeye çalışılarak ölçülen sıcaklık ve basınç değerleriyle eşleşme sağlanması hedeflenir. Bu çalışmada makine öğrenmesi yardımı ile jeotermal kaynakların doğal durum modellemesinin kilit değişkenlerinden biri olan geçirgenlik değerlerinin belirlenmesi amaçlanmıştır. Öncelikle, TOUGH2 sayısal modelleme yazılımına çeşitli geçirgenlik dağılımları girdi olarak verilmiştir. Sıcaklık ve basınç değerlerinden oluşan çıktılar işlenerek sentetik bir veri kümesi hazırlanmıştır. Bu veri kümesi ile doğal durumdaki belirli bir basınç ve sıcaklık dağılımını sağlayan geçirgenlik değerlerinin tespiti için Rassal Orman, Destek Vektör Regresyonu, Çok Katmanlı Algılayıcı, Evrişimsel Sinir Ağları ve Öğrenme Aktarımı yöntemleri kullanılmıştır. Sonuçlar, önerilen yöntemin geçirgenlik değerlerini tahmin edebildiğini göstermektedir. Söz konusu tahminin makine öğrenmesi yoluyla daha hızlı bir şekilde yapılabilmesi jeotermal sahaların modellenmesine zamansal ve ekonomik katkılar sağlayacaktır.

ANAHTAR KELİMELER: Jeotermal Modelleme, Doğal Durum modellemesi, Evrişimsel Sinir Ağları, Simülasyon, TOUGH2

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

ANN	Artificial neural network
CNN	Convolutional neural network
MLP	Multilayer perceptron
ReLu	Rectified Linear Unit
RF	Random forest
SVM	Support vector machine
SVR	Support vector regression

1. INTRODUCTION

The development of civilization and society has always been dependent on energy. Throughout history, the planet's population and the average person's energy consumption have been growing rapidly. Even though the primary energy sources are still fossil fuels, there is a growing transition from fossil fuels to renewable sources such as wind, solar and geothermal. With the effects of climate change and pollution become more visible with every passing year, the importance of renewable energy is also growing. In order to execute the transition to renewable energies, it is vital to advance the technologies used in these sectors as well. Like in every other industry, digital technologies are also pivotal in developing the energy sector. With the implementation of digital technologies, firms could reduce the workforce, the time, and the cost of projects undertaken. Machine learning, also the primary tool used in this study, has proven to be one of the most valuable advances in digital technologies. Machine learning is also gaining popularity in the energy sector by providing solutions to everyday problems, especially in the oil and gas industry. In drilling, production, and transportation operations, many firms include machine learning applications in their solutions. Following the advances of the oil & gas industry and transforming most of them for their uses, the geothermal industry is also implementing machine learning applications in its inner workings. Using natural language processing to check daily drilling and operations logs, image processing to analyze the drilling cuttings, enhancing the seismic interpretations with advanced algorithms are common applications of machine learning used in the industry, which will also be migrated to the geothermal industry with time. One of the most promising fields of the geothermal industry in which machine learning is being used is reservoir engineering and modeling.

1.1. Problem

During geothermal modeling, one of the main challenges is determining the permeability distributions for natural state conditions. One of the critical points in geothermal modeling is creating the natural state model, which depends on correct permeability distributions and is generally determined by trial and error procedures. The trial and error process, by its nature, takes much time. Thus, increases the cost of the projects.

Other approaches to determining the parameters include inverse calculation methods, which automate the trial and error processes by implementing optimization functions. These methods are generally helpful if optimization starts from an acceptable point in determining the parameters, thus might not provide a starting point for permeability distributions.

Geoscience problems are generally hard even to define because of having blurry boundaries both in space and time. The problem we try to solve also depends on the location, it is overly multi-variate and does not follow linear relationships, so it can not be solved by linear approaches [1].

Numerical reservoir simulation in geothermal reservoirs is a powerful technique for visualizing and better understanding the real-world situation of the reservoir. It also allows the immediate execution of any desired operation and observation of its consequences without doing so in the real world, saving time and money. With numerical reservoir simulation, researchers can determine the optimum production and injection rates of the geothermal fields, predict the problems before they became hard to deal with, and guide the management of the field. Thus, numerical simulation is vital in developing the fields to use this renewable energy source to its fullest potential.

1.2. Aim

This study proposes a machine learning-based permeability distribution determination method for natural state modeling of geothermal fields from temperature and pressure readings. Using five proven machine learning algorithms(RF, SVR, MLP, CNN, Transfer Learning), we present the results using the normalized root mean squared errors and visual representations of the predictions versus expected values.

1.3. Importance of the Study

This study is the first in its field to use the fundamental tools of machine learning in a comprehensive way to determine the permeability values in geothermal fields. With our approach, we have generated a method to mimic nature roughly while creating the base models. By introducing the constant permeability zones to the model as cap rocks or reservoir rocks and the shuffling of the sections to capture the folds and faults, our proposed

method could generate models that would better resemble the natural process. This also results in a more detailed model with more rock sections to assign the permeabilities. Furthermore, experts can still influence the model to increase its performance by creating the base model with their field knowledge using our proposed method.

Being an addition to the inversion methods, our study can also be used with other inversion tools as an initial estimator. Users of PEST or iTOUGH2 can start the optimizations with the permeability distributions created by our method and increase their success.

Our proposed method is an essential addition to the field by introducing new opportunities for cost savings and giving researchers new tools to model the geothermal fields better and use this renewable energy to its full potential.

2. BACKGROUND AND LITERATURE REVIEW

2.1. Modeling of Geothermal Fields

The numerical modeling of a geothermal field happens in several steps. In each step, it demands the dedication and the experience of the engineers. A successful model of a field can take months to create. The first step in geothermal modeling is creating the conceptual model of the field. This conceptual model is created from the information gathered by direct methods such as analysis of drilling core plugs or well logs and also indirect methods such as seismic analysis, magnetotelluric analysis, and well test analysis. Direct measurements, especially the core plug analysis, give critical information about the rock properties. One of the essential features of these rock properties is called permeability. Permeability is one of the main parameters used in numerical modeling. It is a property that measures how well a fluid travels in a porous medium or rock.

Using the conceptual model, identified rock properties are assigned to the numerical model, composed of many cells to mimic the underground structure. A typical geothermal model includes the thermal anomaly, which creates the geothermal field. This thermal anomaly can usually be introduced to the model as a high-temperature water flow area or high heat flux region. After setting the position of the thermal anomaly, other features such as the air pressure, ambient temperature, and underground aquifers might be inserted into the model as boundary conditions. With these steps, the basic structure of the geothermal field is created.

Following the creation of the basic model, the numerical simulation software, which models the three-dimensional flow of fluid and heat, is used to check if the model is valid by running the model until it reaches the natural state conditions. Steady or quasi-steady-state condition is reached when the changes in the model between a set amount of time steps are small enough to be neglected. At this stage, the model could be compared with the data gathered from the geothermal field, such as measured temperature and pressure data (Figure 2.1). If the pressure and temperature values of the actual well measurements are matched with the created model results, we may conclude the status of the geothermal field before any wells opened is closely modeled, and the model could be further improved with historical production matches.

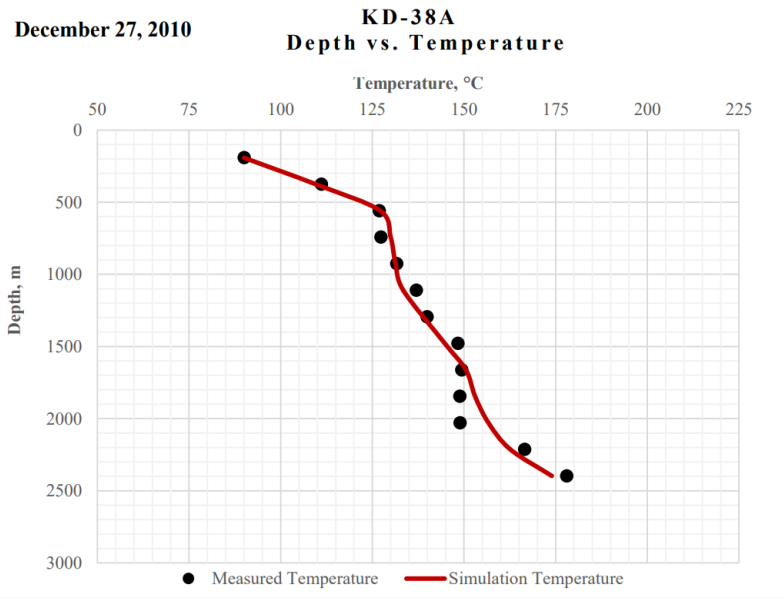

Figure 2.1 - A temperature match from a geothermal numerical model [2]

To summarize, modeling a geothermal field start with the creation of the conceptual and the geological model. Then rock and fluid properties are assigned to the model. After setting up the model's boundary conditions according to the conceptual model, the numerical simulators are finally run to predict the production of the wells and the change of temperature and pressure of the fields. Modeling of the geothermal fields is mainly performed to achieve these future predictions. However, it is hard to determine the correct values to be assigned to the rocks, and inversion of the models is required. In inverse modeling, engineers collect field data and find the rock properties that match the data after running it on simulation software, thus validating the assigned rock properties. While being vital in geothermal modeling, the inversion procedures also take too much time, and because of that, new methods are being developed for inversion procedures.

2.2. Literature Review

During numerical modeling of the geothermal fields, assigning correct rock properties to the model is very important, and at the same time, a challenging task. To validate the assigned rock properties, engineers collect field data first, assign the rock properties to the created model, and try to match the collected data with the results from the numerical simulators. This method is called inverse modeling. Trial and error is the most common inversion method in determining the properties, but automatic inversion tools and statistical

approaches [3], [4] are also used. While being vital in geothermal modeling, the inversion procedures also take too much time, and because of that, new methods are being developed for inversion procedures.

Usage of inversion software attracted some attention with an automatic inversion tool iTOUGH2 [5]. In this first study, researchers calibrated the field's permeabilities during the natural state by setting five different rock types to the field and asking the inversion tool to find the optimum permeability values of these rocks [6]. Following these studies, other researchers also employed automatic tools in their workflow [7], [8]. Later increasing the number of rock types in their models, researchers optimized fields with up to a hundred different permeability values [9]. While working on numerical simulations, the trade-off between the computational costs and the model's precision is always an issue. Therefore, researchers have always tried to balance the number of rock types and the computation time [10],[11]. The inversion software named PEST [12] is another option, which finds increasing interest in the field [13], [14].

Using inversion software helps the researchers, but they also come with their problems. First of all, inversion is still a very computationally heavy task, and determining the properties of increased rock numbers still takes too much time. Also, generally, inversion software is helpful in optimization only if the starting permeability distributions are good enough [15]. In practice, because of common convergence errors, experts usually use their own experiences and rely on train and error procedures.

With the increase of machine learning applications in geoscience, inversion problems also found interest. Researchers tried SVR to find a solution to the problem with some success [16].

2.3. Machine Learning Methods

In this study, permeability distributions of the geothermal systems for natural state modeling are trained from the generated synthetic data composed from the related temperature and pressure distributions. By using our dataset, Random Forest, Support Vector Regression, Multi-Layer Perceptron, Convolutional Neural Networks, and Transfer Learning methods are trained, and their success is compared to each other by the Normalized Root Mean Squared Error of each model.

2.3.1. **Random Forest**

Random Forest (RF) is a versatile supervised learning method that can be used for both regression problems and classification problems. Random Forest belongs to one of the most popular and best-known ensemble methods known as Bagging methods [17]. In ensemble methods, many weak learners are aggregated to create a stronger learner. In the Random Forest case, the weak learners are Decision Trees.

A Decision Tree is a tree-like structure in which a node is a conditional decision point to split the flow into two different branches in numeric cases. A tree greedily looks for the best point to split the whole training data from the root [18]. The Decision Tree algorithm learns the data by calculating the cost function and splitting the data according to this. The procedure tries and tests the splitting points and calculates the errors of the results according to this cost function. This cost function is generally chosen as the sum of squared errors (SSE) in regression problems. In equation 2.1, y represents the expected outcome of the training sample and $\bar{y}$ represents the prediction [19], [18]. S_1 and S_2 represents the two sets that are dvided in a decision node.

$$SSE = \sum_{i \in S_1} (y_i - \bar{y}_1)^2 + \sum_{i \in S_2} (y_i - \bar{y}_2)^2 \tag{2.1}$$

$$\bar{y}_1 = \frac{1}{|s_1|} \sum_{a=1}^{S_1} y_a$$

$$\bar{y}_2 = \frac{1}{|s_2|} \sum_{a=1}^{S_2} y_a$$

The process of splitting continues until all data is split [17]. This may lead to overfitting the training data, in which the model memorizes the training data and can not generalize well.

Random Forests, composed of Decision Trees, get most of their properties from trees themselves but also differ on application. In Random Forest, individual trees are trained from a sample of the dataset. In which the sample is selected from the whole dataset with replacement. Selecting the samples with replacement is called bootstrapping [19]. After bootstrapping, each tree would train and maybe overfit the data they are trained on, but each

tree would be trained on a different data so that the whole model would be free from overfitting problems. The training also does not depend on the calculation of the information gain anymore. Instead, for each split, the Random Forest picks features randomly to be assigned to a node, thus creating a more versatile forest. The combination of the trees, the forest can overcome the problem of overfitting and generalize well by either voting for a decision or averaging the values produced by individual trees [20].

2.3.2. **Support Vector Machines**

Support Vector Machine is a supervised learning method that is versatile and very powerful. Due to its capabilities of performing both classification and regression tasks, it has become a popular model with wide usage in machine learning tasks [17].

The main idea behind SVM is to separate two linearly separable classes with a line or hyperplane that also stays at a maximum distance from each of the classes. It is possible to draw infinitely many lines between two linearly separable classes, but some drawn lines could be very close to the training instances and may not perform well with new data. In order to maximize the effectiveness of the line to separate the classes even with new data, it is essential to place it so that it stays away from the samples of the two classes (Figure 2.2). The distance between the closest data points and the separation line or hyperplane is called the margin.

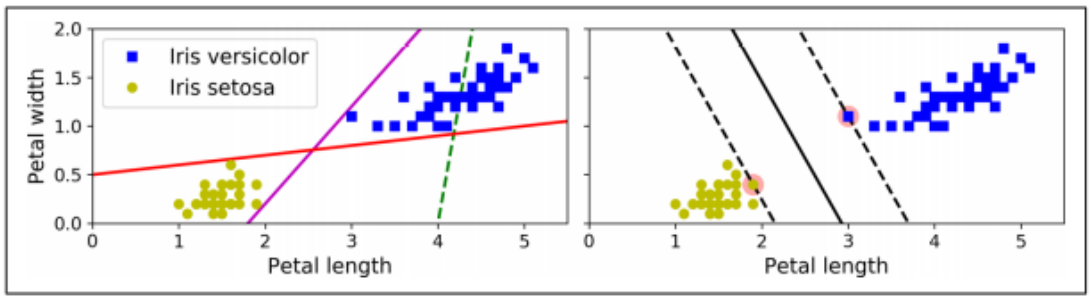

Figure 2.2 - Separation lines and the Decision Boundary *[17]*

While calculating the margin and the position of the line, only the closest points to the line are taken into account, and these points are called support vectors. These support vectors define the separation line or hyperplane, and only they are considered when the optimization is performed. SVM's goal is to maximize the margin by learning from the training data [19].

If we define the input vectors of the dataset as $X = \sum_{i=1}^{n} \vec{x}_i$ where $\vec{x}_i = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_j \end{pmatrix}$. Then equation 2.3.2.1 describes the hyperplane, in which $\vec{w}^V$ is the coefficients vector of the hyperplane.

$$w_1 x_{i1} + \cdots + w_p x_{ip} + b = \vec{w}^V \vec{x}_i + b = d \qquad (2.3.2.1)$$

The field composed of the hyperplane and the margin is called the decision boundary. Maximal Margin Classifiers do not allow any samples from the classes to be inside the decision boundary. Thus, if the classes could not be separated linearly, the model would fail, and this approach cannot solve most real-life problems. By relaxing the margin calculations rules, the decision boundary can include some samples from the classes. This relaxation is called Soft Margin Classification, and it gives SVMs the power to tackle outliers and problems which are not linearly separable. The new goal is then maximizing the margin while keeping the samples that violate the margin at a minimum. In Soft Margin Classification, a new set of coefficients is included, which gives the margin the ability to move in different dimensions, helping the new goal.

SVMs can also handle nonlinear data by employing kernels. Polynomial kernels and Gaussian RBF are two of the most common kernels used in machine learning, which uses the principles of adding polynomial features and similarity functions to help separate the classes linearly.

2.3.2.1. Support vector regression

As previously mentioned, SVMs are very versatile, and they can be designed to suit many needs. SVMs can not only be used to classify linear and nonlinear problems; they can also be used in linear and nonlinear regression problems [17]. By reversing the goal of SVMs, trying to find the maximum margin possible with minimum violations, into the goal of finding a decision boundary that includes as many samples as possible and trying to limit the samples that are left outside, SVMs can be used in regression problems. In this study, a kernelized version of SVR is used (Figure 2.3).

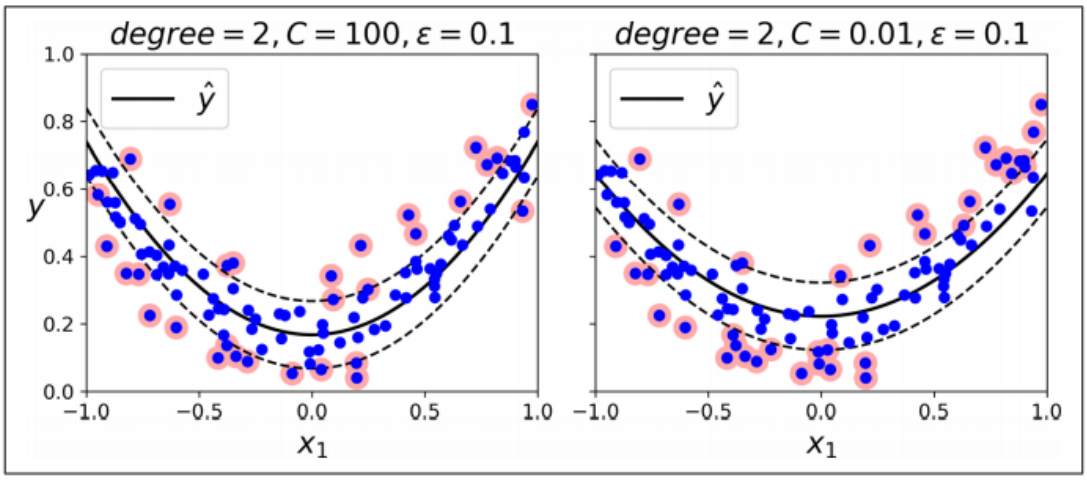

Figure 2.3 - A Nonlinear SVR *[17]*

2.3.3. **Multilayer Perceptrons**

Getting their influence from the nature of biological neurons, Artificial Neural Network (ANN) is one of the essential tools in Machine Learning [17]. An ANN can be written as a nested function. A three-layered ANN is shown in equation 2.3.3.1.

$$f_{NN}(x) = f_3\left(f_2\left(f_1(x)\right)\right) \tag{2.3.3.1}$$

$$f_l(xz) \underset{=}{^{def}} g_l(W_{lz} + b_l)$$

In this equation, l can span from 1 to any number of layers, g_l is an activation function such as relu, or tanh. The parameter W_l is a matrix and b_l is a vector, and they are learned using gradient descent and a cost function such as RMSE.

A sub-branch of ANNs, Feed Forward Neural Networks, in which calculations can only go in one direction and not cyclic, includes multilayer perceptrons. A multilayer perceptron should contain at least three layers of perceptrons, the input, hidden, and output layers (Figure 2.4). The input layer is where the data enters the structure. The hidden layer resides in between input and output layers, and they can be more than one. The output layer is where the MLP either gives the training results or, using a supervised learning technique called backpropagation, recalculates the weights between connections according to the errors and restarts the process, thus learning. A perceptron calculates the outputs by multiplying the inputs with weights, adding the result together, and passing the results from an activation function. An activation function transforms the results passes thru them depending on the

type of the activation function. Each layer can use different activation functions, and by combining different layers with different weights and biases, MLP can solve nonlinear problems.

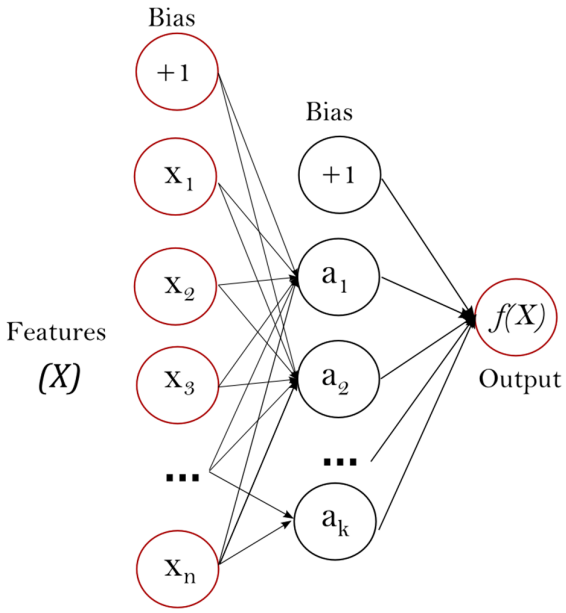

Figure 2.4 - An MLP with one hidden layer *[20]*

2.3.4. **Convolutional Neural Network**

A Convolutional Neural Network (CNN) is a deep learning network that specializes in grid-like data structures, such as images. CNN's are very good at capturing the patterns on images such as lines, dashes, circles, ears, and faces. Because of that, they are widely used in computer vision. Unlike previous image computer vision algorithms, they do not need any preprocessing to be implemented.

The human visual cortex inspires the creation of CNN. Different layers stacked on top of each other analyses and identify the images seen with increasing complexities in the visual cortex. Convolutional neural networks are a type of feed-forward neural networks with convolutional layers [17]. Like in the human visual cortex, each unit in a CNN can process and analyze data in its field of vision. By stacking a few of these convolutional layers on top of each other, bit by bit, a CNN can identify the handwritten digits. By staking more, a convolutional neural network can even help self-driving cars (Figure 2.5).

Typically a convolutional neural network has three layers. These are a convolution layer, a pooling layer, and a dense or fully connected layer.

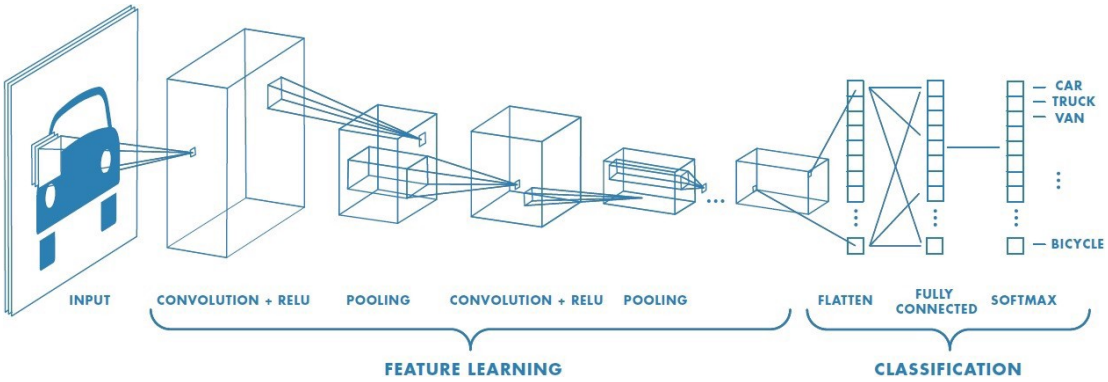

Figure 2.5 - A CNN Architecture *[21]*

A CNN's central unit is the convolutional layer. The convolutional layer contains the kernels, which are filters for different aspects of the image that is being processed. Just like looking at a picture thru a microscope, a kernel can only see and manipulate a small section of the image. By moving the kernel on the image, different aspects of the image can be collected. Later, to determine the most critical aspects of the images, these layers can be pooled into smaller ones by moving another layer on top of others called pooling layers. These pooling layers, either by averaging or filtering the maximum, condenses the features, thus enforcing the essential features to survive. The third type of layer in a CNN is a dense or fully connected layer. The convolutional layers and pooling layers manage to collect the features from the images by only seeing a portion of the image, thus not being in connection fully. However, dense or fully connected layers collect the gathered data and fully connect to the preceding and succeeding layers to learn the relationship between the image features and the output. Using the combination of these three fundamental layers and different activation functions, many different CNN architectures can be created to solve many problems.

2.3.5. **Transfer Learning**

Transfer learning is a powerful supervised learning method, where the previously learned information from a model can improve the generalization performance of another model [22]. In computer vision problems, if there is a similar pre-trained model, it is usually very beneficial for the task to implement some low-level features from the pre-trained model. Since training deep learning, models can take quite a long time, when applicable, using

transfer learning can reduce the time spent by avoiding the random initialization of the weights and biases.

When using a pre-trained model, the first step is removing the original classifier or regressor from the model. Later a new classifier and regressor suitable for the new problem are attached to the model [23]. Finally, the model can be fine tunes by either training the whole model, or freeze some layers and train the rest of the model, or freezing the convolutional base altogether Figure 2.6.

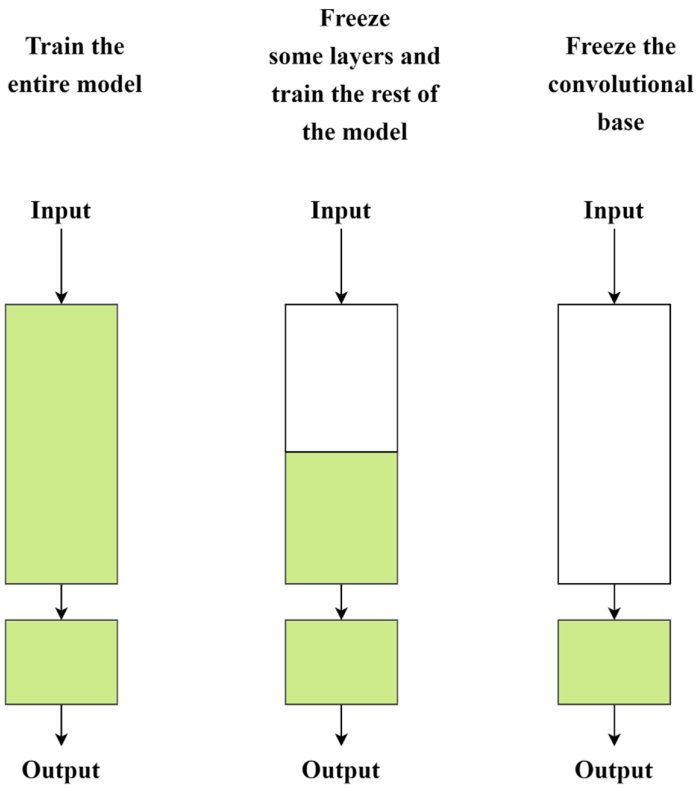

Figure 2.6 - Fine-tuning for transfer learning

The convolutional base performs the feature extraction in CNN, so it is vital to decide how much of it will be transferred from another model. Training the whole model is generally a very time and energy-consuming task. It is used when the current dataset is large enough, and the problem is different from the pre-trained one. Freezing some layers and training the rest of the model is usually preferred when having a large dataset and similar problems to the pre-trained one. In this case, the level of the frozen layers is determined by the similarity of the problem. The last case, freezing the convolutional base, can be beneficial if the dataset of the new problem is small and the problems are similar to each other.

VGG-16 is an architecture invented by the Visual Geometry Group. It has 13 convolutional layers with ReLU activation functions and three fully connected layers. By stacking up the same convolutional layers, the model reaches 138m parameters. The VGG group also designed a deeper version of VGG-16, called VGG-19.

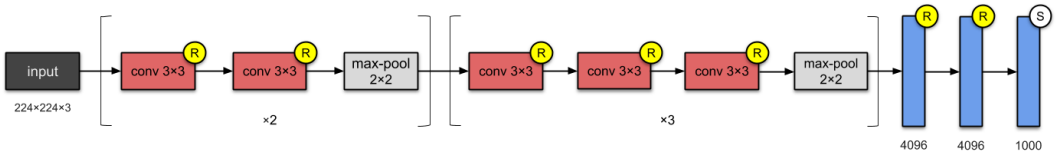

Figure 2.7 - VGG-16 [24]

3. METHOD

In this study, the permeability distributions used in natural state modeling of geothermal fields are determined from the temperature and pressure values, which are relatively easy to measure in the field. For this purpose:

- A geothermal field model, called the base model, comprised of 2,744 cells (14x14x14), is created in Petrasim [25]
- A permeability distribution method that crudely mimics real-life scenarios to generate appropriate geothermal fields is created. This method creates 11 different regions of different permeabilities, whose values are chosen between 0.01-200 md (~1.0E-17 m^2, ~2.0E-13 m^2) permeability values randomly.
- Ten thousand different permeability distributions are generated by the mentioned method and fed into the base model.
- The model is then run on TOUGH2 [26] for each different permeability distribution, and the output of temperature and pressure values are collected.
- The synthetic dataset is created with 10,000 different permeability distributions and the corresponding temperature and pressure distributions.
- This dataset is then split into three as the train(6,000), validate(2,000), and test(2,000) datasets.
- The permeability distributions are trained from temperature and pressure distributions using RF, SVR, MLP, CNN, and Transfer Learning methods from this dataset.

The data creation process can be seen in Figure 3.1.

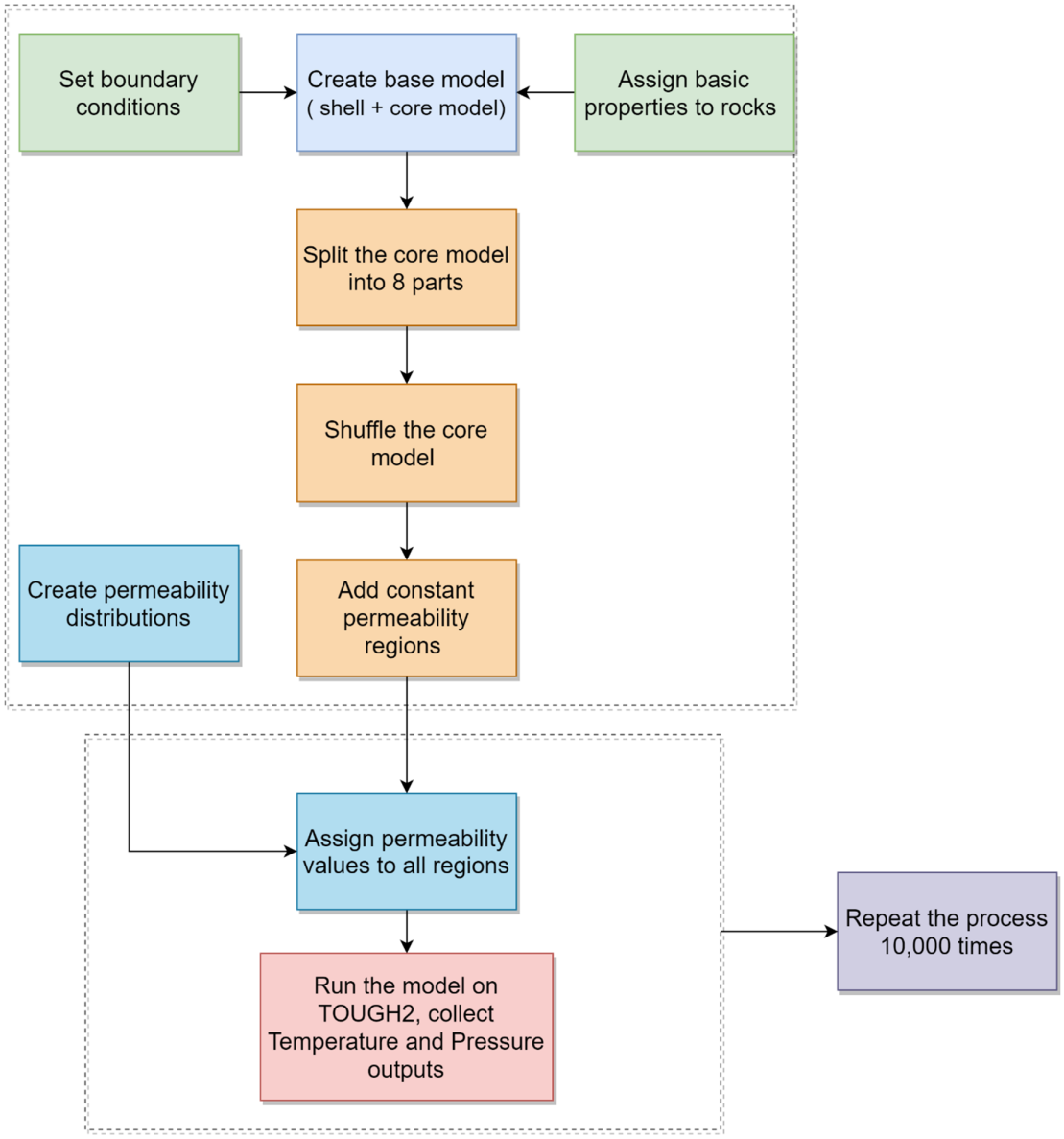

Figure 3.1 – Data preparation process.

3.1. Preparation of The Base Model

In order to create the dataset to find the relationship between the temperature and pressure distribution of the geothermal fields with its permeability distributions, a base model is created. This base model is created in Petrasim, software commonly used as a graphical user interface for creating input files and analyzing the outputs of the TOUGH simulators family. The base model comprises 2744 cells, with 14 cells in x, y, and z. Deciding the number of cells to be included in a model is generally one of the most critical parts of numerical modeling. The computational power at hand dictates the number of cells, and the number of cells should be no more than the minimum with which one can capture

the geothermal field's structure or the unique geometries that should be investigated in the model. Since the base model is to be run 10,000 times and the study aims to find the general distribution of permeability, the number of cells is kept relatively small. Inside the model, two cells away from the outer boundaries, a 10x10x10 region is chosen as the core region. The permeability values in the core area are later manipulated, and the dataset is generated from the results of these manipulations. The remaining outer cells, the shell, are kept as a buffer to limit the boundary effects of the numerical modeling. As seen in Figure 3.2, the outer cells have a bigger volume than the core cells. The reasoning behind this is again to prevent the boundary effects from reaching the core cells.

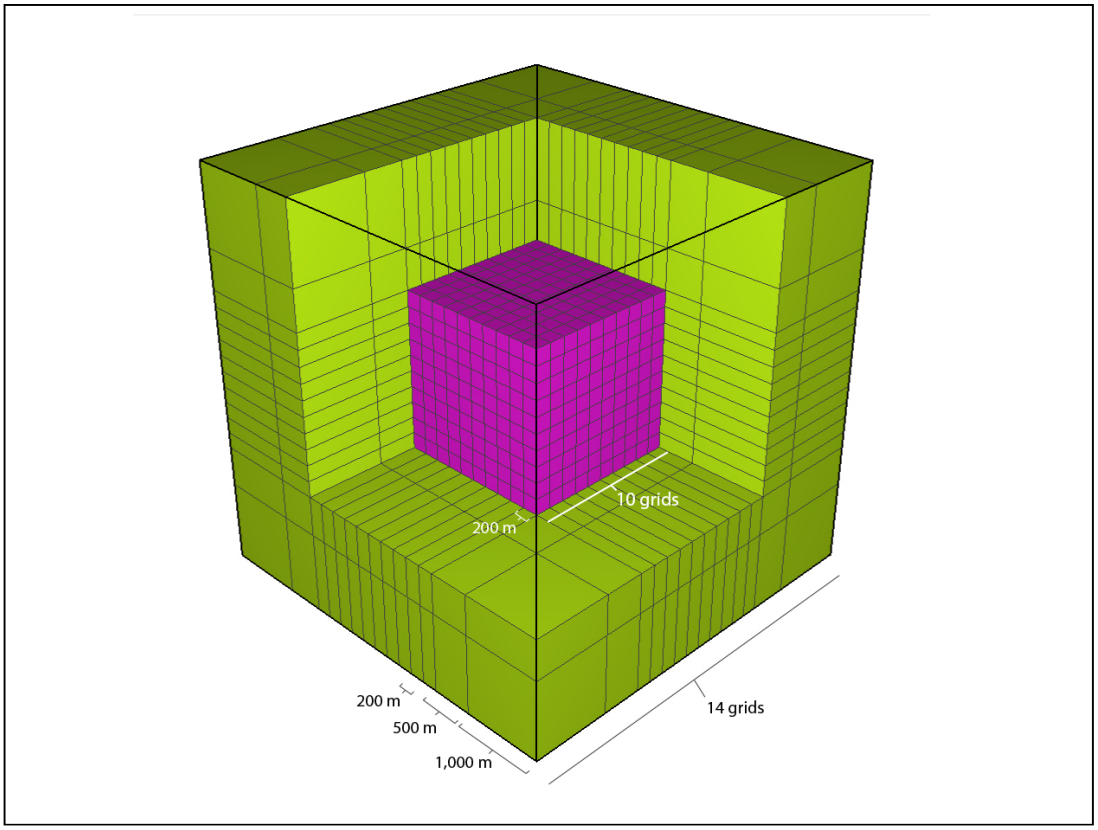

Figure 3.2 - Base model, the shell and core regions

The dimension of the base model is 5,000m in all directions. The detailed grid structure of the model is given in Table 3.1.

Table 3.1 The Grid structure of the base model

Direction	Cell #	Cell Size
X	1	1000
X	1	500
X	10	200
X	1	500
X	1	1000
Y	1	1000
Y	1	500
Y	10	200
Y	1	500
Y	1	1000

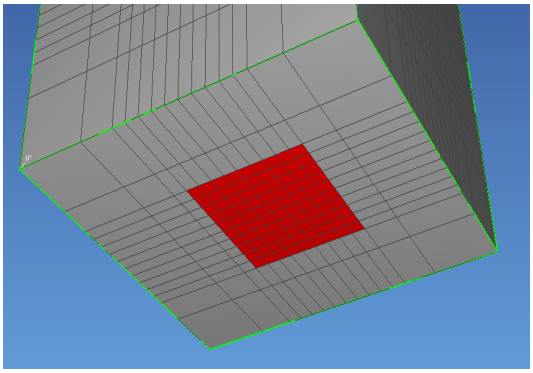

Figure 3.3 - Hot water source of the base model

As boundary conditions, the upper layer of the model is set as an open boundary with a constant temperature of 62.9 °C and a constant pressure of 55.4 bar, which are typical values at 500 m depth in geothermal fields.

The rock densities of the model are set as an expected value of 2600.0 kg/m³ for all cells. Similarly, the wet heat conductivity of the whole model is set as 2.0 W/m·K, and the specific heat is assigned as 1000.0 J/kg·K, which are typical values. Water source is introduced into the model from the bottom layer (Figure 3.3) with the rate of 0.15 kg/s and the enthalpy of 1.4E6 J/kg.

The model's porosity values are 0.01 and 0.05 for inner and outer regions, common in geothermal fields. The permeability values are set as 1 md (~10^{-15} m^2) throughout the model. The permeability values that are used in this study are included as a multiplier to the 1md value previously set. This method lets the experts assign different permeability values to different regions, thus carrying the field experience of the experts to the model and leading the machine learning processes in the correct direction by constructing the base model the way they desire.

Equation of state 1 (EOS1) is used for numerical simulation, and the initial conditions of the model are set as 12.0 bar + 0.0868 bar/m for pressure and 33.0 °C + 0.0598 °C/m for temperature. The simulation is set to be run for a million years to ensure that the model reached its natural state.

3.2. Preparation Of The Dataset

A typical geothermal field usually has a reservoir section and a caprock section. The reservoir contains high-temperature fluid, and generally, it has high permeability due to the rock properties or the fault formations inside the reservoir. The caprock section is generally composed of rocks with very low permeability, and also it might have thermal insulation properties. In order to create the permeability distributions, which crudely resemble naturally occurring geothermal fields, it is essential to set some sections of the model as reservoirs and some sections as cap rocks. Also, keeping the natural processes in mind, the permeability distributions should have some recognizable patterns, some local groupings, and some displacements to mimic the depositions, erosions, and faulting.

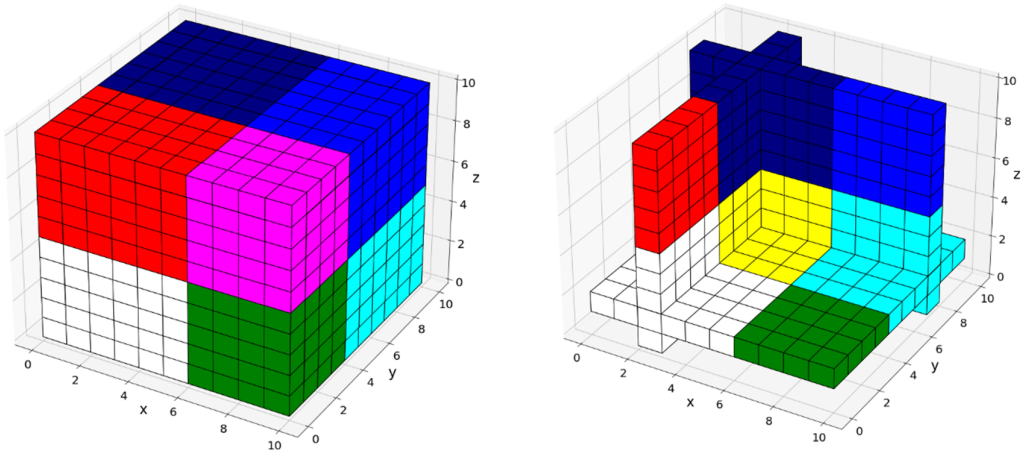

Figure 3.4 - Splitting the core model into eight pieces.

In order to fulfill these requirements and at the same time represent the model with a relatively low resolution, the core model is split into eight regions randomly (Figure 3.4). Before assigning different permeability distributions to these regions, some randomly chosen layers are shuffled by batches of 2 or 3 layers Figure 3.5. Shuffling is included to mimic the natural processes such as erosions and displacements of the rock by faults or folds.

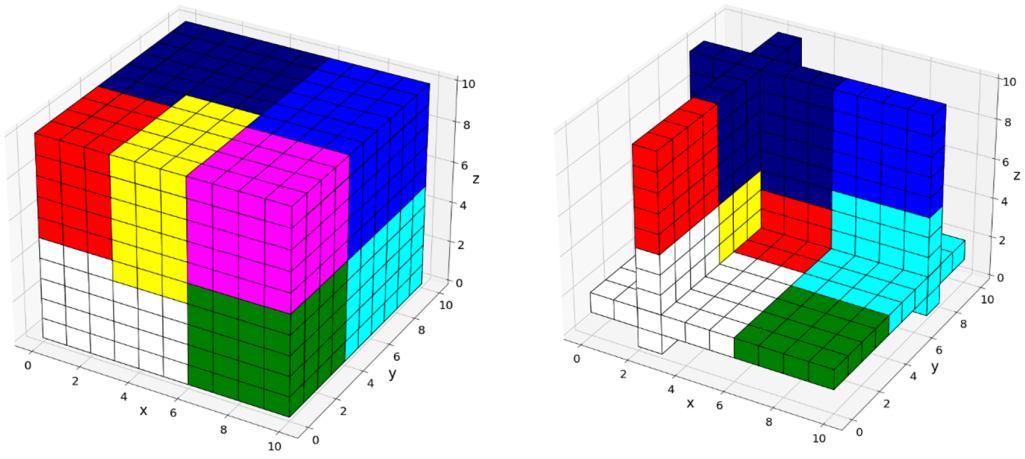

Figure 3.5 - Shuffling of layers in the base model

Lastly, at least one and at most three layers are chosen randomly to become the constant permeability regions (Figure 3.5). These regions are included in the model as high permeability zones to become better reservoirs or possible cap rocks to limit the flow of fluids. If a constant permeability zone is assigned as a caprock, its permeability multiplier is multiplied by 0.01 to ensure the layer has a lower permeability. The constant permeability layers also have a chance to provide a connection between otherwise separate regions depending on the permeability of the layers. After including the constant permeability zones, the number of possible permeability zones in a model becomes 11 in total.

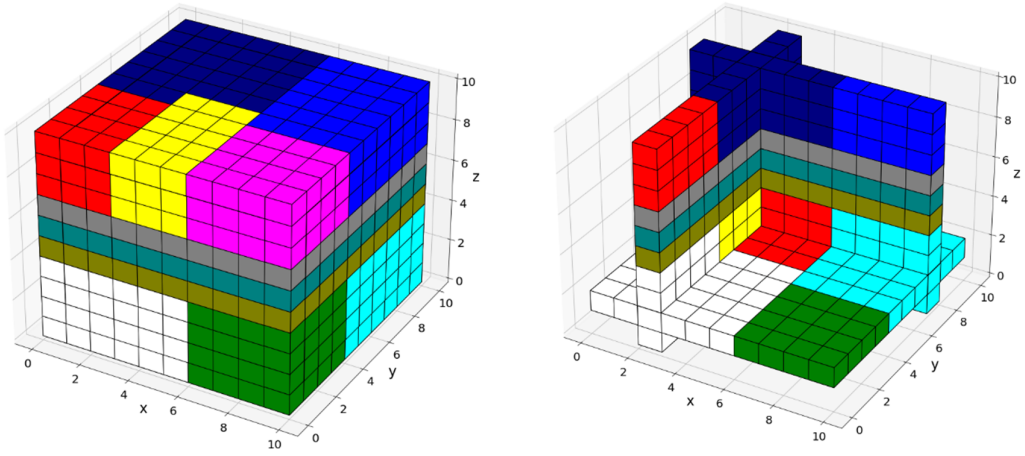

Figure 3.6 - Assigning constant permeability regions to the base model

Next, a table of 11 different permeability values for each run is created by selecting values from the range of 0.01md (~$10^{-17}m^2$) to 200 md (~$2x10^{-13}m^2$) Table 3.2. Figure 3.6

shows that the created permeability values table does not favor any particular value in the range that they are selected. These values are then assigned to 11 different regions of the core model for each of the 10,000 runs (Figure 3.7).

Table 3.2 - Created random permeability table

	0	1	2	3	4	5	6	7	8	9	...
0	197.80	109.91	56.30	15.47	88.90	94.57	9.71	32.67	23.20	125.48	...
1	29.54	189.12	184.84	169.40	124.76	187.20	154.06	73.85	102.30	89.65	...
2	183.89	137.41	159.62	111.60	99.17	6.63	121.23	29.31	87.49	132.80	...
3	101.58	182.82	137.69	124.22	78.20	40.45	170.12	198.51	68.06	56.87	...
4	0.35	141.75	156.82	57.39	190.91	143.37	191.27	65.47	101.81	56.11	...
5	39.80	129.73	30.03	152.71	29.01	136.61	95.13	74.14	189.68	185.16	...
6	33.49	88.79	176.84	105.60	8.57	125.63	37.33	183.27	177.59	80.79	...
7	8.96	67.43	10.59	1.27	113.54	85.73	198.03	128.81	22.61	44.92	...
8	82.52	57.34	119.10	107.00	40.93	146.77	37.72	135.38	22.32	150.52	...
9	20.28	37.01	89.79	101.91	193.63	183.12	118.35	100.64	102.87	122.54	...
10	45.21	141.15	131.45	55.22	40.08	174.60	186.61	35.98	130.00	47.01	...

11 rows × 10000 columns

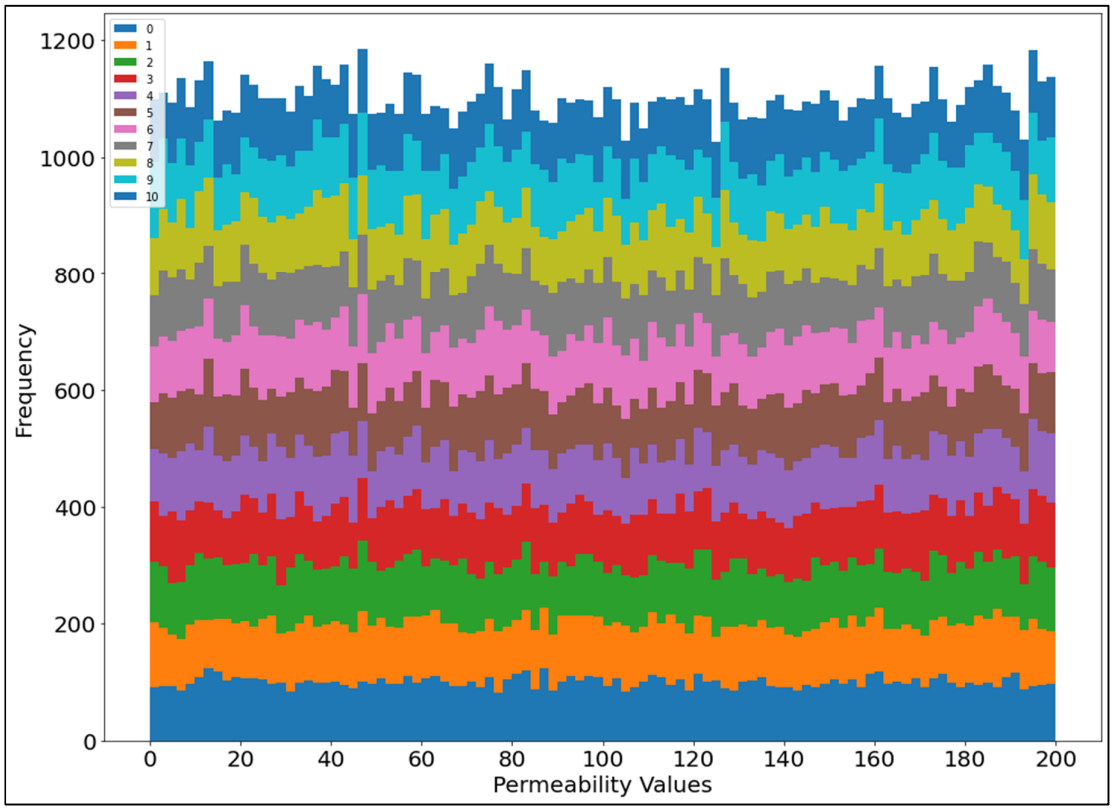

Figure 3.7 - Stacked Histogram of 11 Random Permeability Values for 10,000 runs

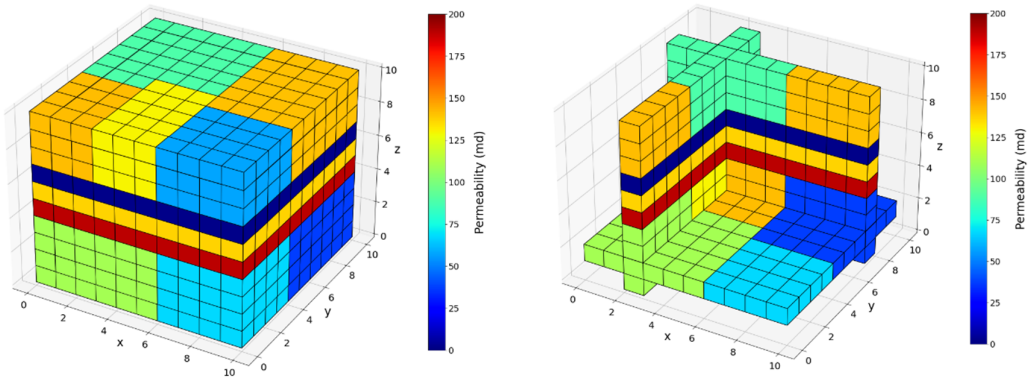

Figure 3.8 - A sample from the base model, showing the assigned permeability values

Created permeability distributions table is later used to alter the required sections of the input file of TOUGH2, generated from the base model discussed previously (

Figure 3.8). During input file creation, first, the cells of the geothermal model related to the 11 regions are determined for each run. Next, for each run, the permeability values of the regions are assigned to the determined cells. When 10,000 input files are created and located in their designated folder, a subprocess written in python is called the TOUGH2 EOS1 for running the simulations. After each run, TOUGH2 provided the output related to the permeability distributions given as an input (Figure 3.9 and Figure 3.10). These output temperature and pressure distributions are then collected into a single file with their corresponding permeability distributions. The created dataset has the dimensions of 10,000x2,000 for features (temperature and pressure values) and 10,000x1,000 for the targets (the permeability values creating the related temperature and pressure values).

The dataset later split into three as the train (0.6), validate (0.2), and test (0.2) datasets. Train dataset contains 6,000 instances, while validate and test datasets contain 2,000 instances each.

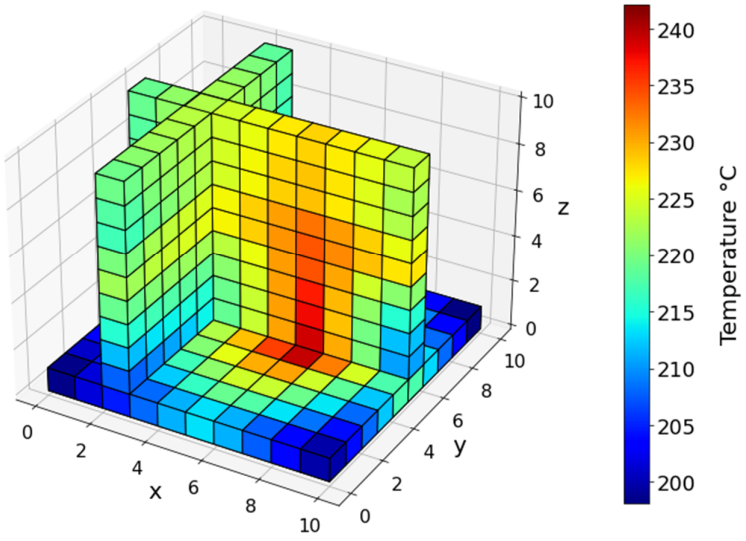

Figure 3.9 - 3d Temperature Data

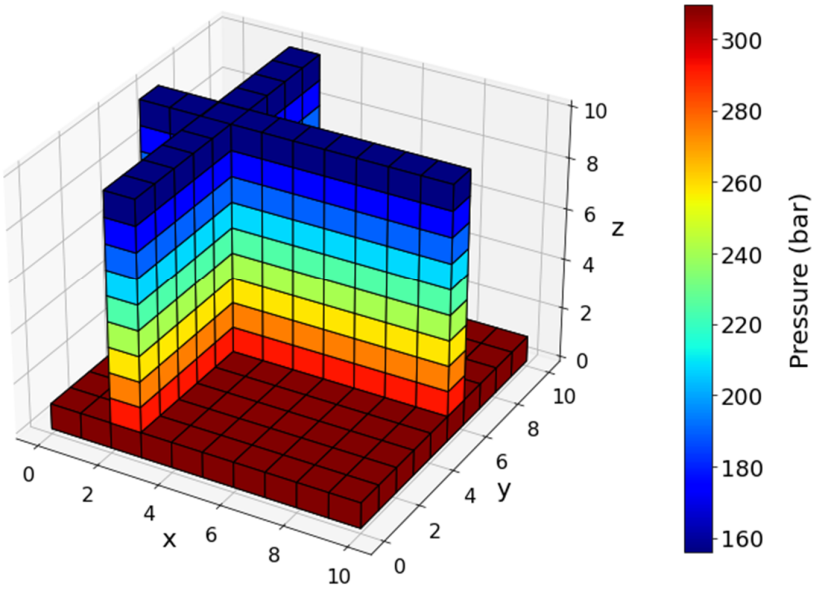

Figure 3.10 - 3d Pressure Data

3.3. Preprocessing and Feature generation

Feature generation is a process to produce new features from the raw dataset, which may be more informative for the models to be built upon or more representative of the data itself after the process. The temperature and pressure data and the permeability distributions are meaningful only if they are in a relationship with their neighbors. In order to represent these neighborhood relationships in one-dimensional space to be used in Random Forest, Support Vector Regression, and Multilayer Perceptron, it is decided to create the gradients of the cells with their first, second, and third neighbors in each layer (2d). The gradients are calculated as the difference between the cells' value and the average of the four cells with the required step size in 4 cardinal directions. (Figure 3.11 and Figure 3.12)

	0	1	2	3	4	5	6
0				3rd			
1				2nd			
2				1st			
3	3rd	2nd	1st		1st	2nd	3rd
4				1st			
5				2nd			
6				3rd			

Figure 3.11 - Neighborhood gradients of one, two and three-order

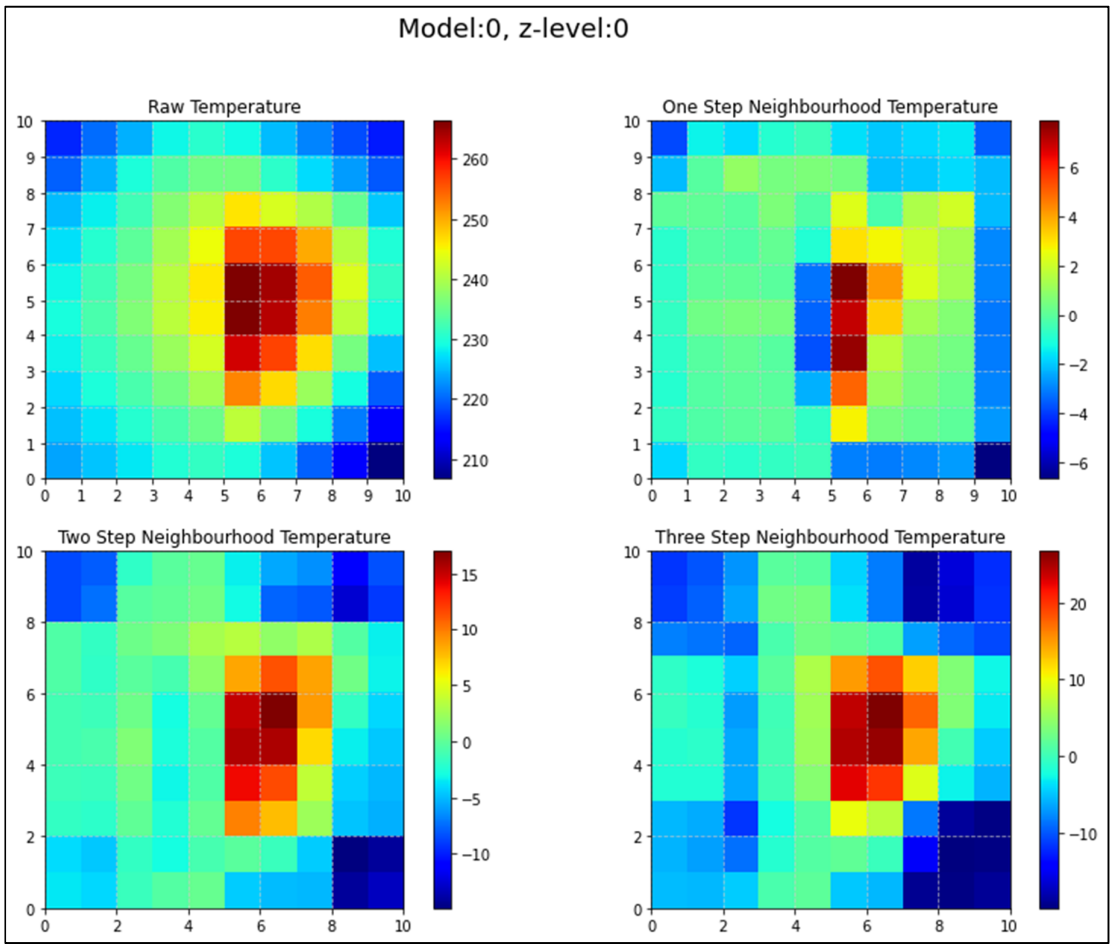

Figure 3.12 - Temperature Neighborhoods

4. EXPERIMENTAL STUDY AND RESULTS

Random Forest, Support Vector Regression, Multilayer Perceptron, Convolutional Neural Networks, and Transfer Learning models are applied to the dataset to test the proposed method of determining the permeability distributions from temperature and pressure distributions. The models are trained for each cell individually in each experiment, and Root Mean Squared Errors are calculated from the predicted permeability values and expected permeability values of 1,000 cells in each simulation run. Later these errors normalized by dividing them with the difference of the maximum values and the minimum values of the observed permeability to find the Normalized Root Mean Squared Errors.

After each model, the predicted values and the model's output are also shown graphically to understand better the models' applicability for both the best and worse scored runs.

4.1. Evaluation Metrics

In order to assess the performance of a machine learning method, we can use several evaluation metrics to measure how well the predictions and expectations are matched [27]. For regression problems, the mean squared error is one of the most commonly used metrics. In equation 4.1, y represents the expected outcome of the training sample and $\bar{y}$ represents the prediction from the training sample.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y}_i)^2 \tag{4.1}$$

If a model's prediction is close to the observed or expected value, the resultant MSE will be small. The root of MSE is called root mean squared error, shown in equation 4.2. RMSE is very easy to interpret because it has the same unit as the predictions [28].

$$RMSE = \frac{1}{\sqrt{n}}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2} \tag{4.2}$$

Since RMSE is also dependent on the scale of the problem, it can be used to compare different methods on the same dataset, but it can be challenging to compare the results between different datasets. Thus, adding normalization to RMSE, normalized root mean squared error (NRMSE) can be calculated. The NRMSE is calculated by dividing the RMSE by the difference of maximum and minimum values of training y values, the expected outcome.

$$NRMSE = \frac{RMSE}{y_{max} - y_{min}} \tag{4.3}$$

4.2. Random Forest Model and Results

In previous sections, different sets of neighborhood gradient datasets were already created. To continue the study, it is essential to assess the best combination of these datasets to be trained in the machine learning models. In order to determine the best combination, a decision matrix is constructed. This matrix includes the original temperature and pressure values as well as one, two, and three neighborhood gradients. Since running an entire dataset for all the cells with all the combinations in the matrix would take too much time. A sample of 23 is selected randomly from 1,000 cells to be trained by different datasets and rank the success. In Table 4.1, the default Random Forest model results for different neighborhood datasets can be seen. The temperature neighborhood of one performed best, which is expected. Since temperature readings differ significantly with changes in the readings' location in geothermal fields, the one-neighborhood gradient of temperature would contain more information about the field than two or three neighborhood ones. As shown in Table 4.1, the best results are obtained from the three-neighborhood pressure gradient dataset for pressure values. This is also expected since changes in pressure are not common in the same layer in geothermal fields. When the distance between the cells increases, the pressure difference would become more significant and carry more information about the field and the potential distribution of the permeabilities. Following this step, all the combinations of neighborhood gradients are also trained in the default Random Forest model. In Table 4.2, the results of these combinations are shown. As expected, the best performing combination is the Temperature Neighborhood-1 and Pressure Neighborhood – 3.

Table 4.1 - Neighborhood selection study results

	Original Temperature	Temperature Neighborhood - 1	Temperature Neighborhood - 2	Temperature Neighborhood - 3
NRMSE of 23 cells	0.17	**0.145**	0.151	0.159
	Original Pressure	Pressure Neighborhood - 1	Pressure Neighborhood - 2	Pressure Neighborhood - 3
NRMSE of 23 cells	0.196	0.214	0.194	**0.186**

Table 4.2 - Neighborhood combination study results

T= Temperature Neighborhood, P= Pressure Neighborhood									
	T1 P1	T1 P2	T1 P3	T2 P1	T2 P2	T2 P3	T3 P1	T3 P2	T3 P3
NRMSE of 23 cells	0.145	0.139	**0.136**	0.150	0.144	0.141	0.159	0.151	0.149

The best performing dataset (T1-P3) is created by concatenating the previous best performers. The final dataset has 6,000 training instances with 1,000 Temperature Neighborhood-3 gradient data and 1,000 Pressure Neighborhood-3 gradient data. This makes the dimensions of the T1-P3 dataset as 6,000x2,000 for training and 2,000x2,000 for both test and validation sets. Corresponding to these, the targets (the permeability values) have the dimensions of 6,000x1,000 for training and 2,000x1,000 for both testing and validation datasets.

In order to find the hyperparameters to be used in Random Forest, a tuning method called random search is used on the validation set. Some of the overfitting problems faced in the future are negated by not tuning the parameters on the training dataset. The random search algorithm randomly assigns the values for the selected parameters to find the best performing combination of those parameters. The parameters included in the search are the maximum number of features to decide when to split the trees, the depth of the trees, the minimum sample number for assigning a leaf node, and the minimum number of samples to split the nodes. By implementing the RandomizedSearchCV algorithm of scikit-learn [20], the max_depth parameter is assigned four different values to be searched,10,60,110, and

None. Max_features are also searched with two possible values, auto, and sqrt. A total of three values are assigned into min_samples_leaf as, 1,2,4. Finally, min_samples are chosen as 2,5,10. After initializing the algorithm, the RandomizedSearchCV combined the parameters and ran the model with different parameters each time. The resulting hyperparameters matched the default values for max_features (auto), min_samples_leaf (1), min_samples_split (2), and the algorithm advised us to use max_depth at 60 instead of the default value of None. After these hyperparameters, the number of estimators is determined by trial and error. The number of estimators is not included in the random search because it generally dominated the results, meaning a higher number of estimators is often better even without considering the other parameters. With trial and error, the number of estimators is determined as 20. Above this value, a minimal performance gain is seen with a high cost of computing time. Since RandomizedSearchCV also performs the cross-validation when calculating the hyperparameters, overfitting might be overcome, or in this case, the assigned parameters would not be specific to the dataset provided but would also be applicable to the training dataset.

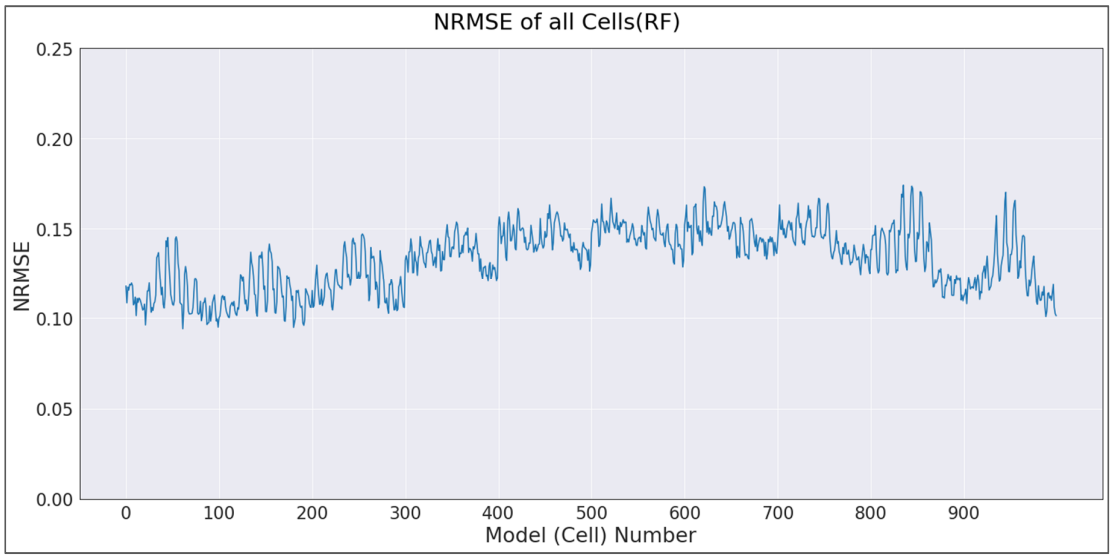

Figure 4.1 - Normalized Root Mean Squared Errors of all Cell Models

After performing the hyperparameter tuning, a Random Forest model for each cell of the simulation model is created. The models are then run in python using scikit-learn's RandomForestRegressor.

In this study, a simulation model consists of 1,000 cells. Moreover, there are 2,000 instances of these simulation models in the test dataset. In the simulation model, each cell is

trained by itself, called the cell model. The collection of these cell models creates the prediction for the simulation model for each instance. So, each of the 2,000 simulation models includes a combination of 1,000 cell models. Cell model 42 has 2,000 different predictions vs. the expected permeability values, each for the different simulation model. Figure 4.2 shows the general success of a cell model, having more points closer to the 45-degree line.

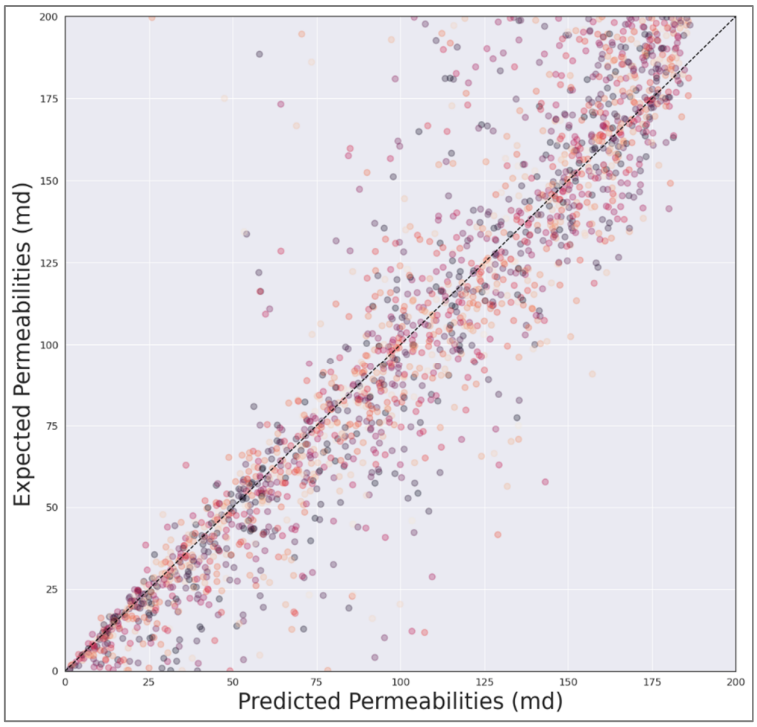

Figure 4.2 - Expected vs. Predicted Permeability values of cell 42

While evaluating the success of the cell models, the normalized root mean squared error (NRMSE) metric is used. In order to determine the general success of the Random Forest algorithm, NRMSE is calculated from the expected and predicted permeability values of all runs as 0.134.

Figure 4.1 shows the success of each cell model. Cells between 400 and 800 are located in the middle of the geothermal model, and their prediction is generally more challenging due to the interference from other cells, both from above and below. Permeability values of remaining cells performed relatively better because of being close to the model's edges, and the permeabilities close to them could be predicted easier due to the undiluted effects of temperature and pressure nearby. The sudden step-like behavior in the graph is due to the layered structure of the geothermal base model. So the predictions of the model change

rapidly in every 100 cells, which composes a 10x10 layer. The NRMSE values of each layer's cell models' predictions have wide margins, leading to more diverse predictions on the layers. This helps the Random Forest cell models to capture sudden changes in permeability, but it also hinders the predictions when the layer has a single expected permeability value.

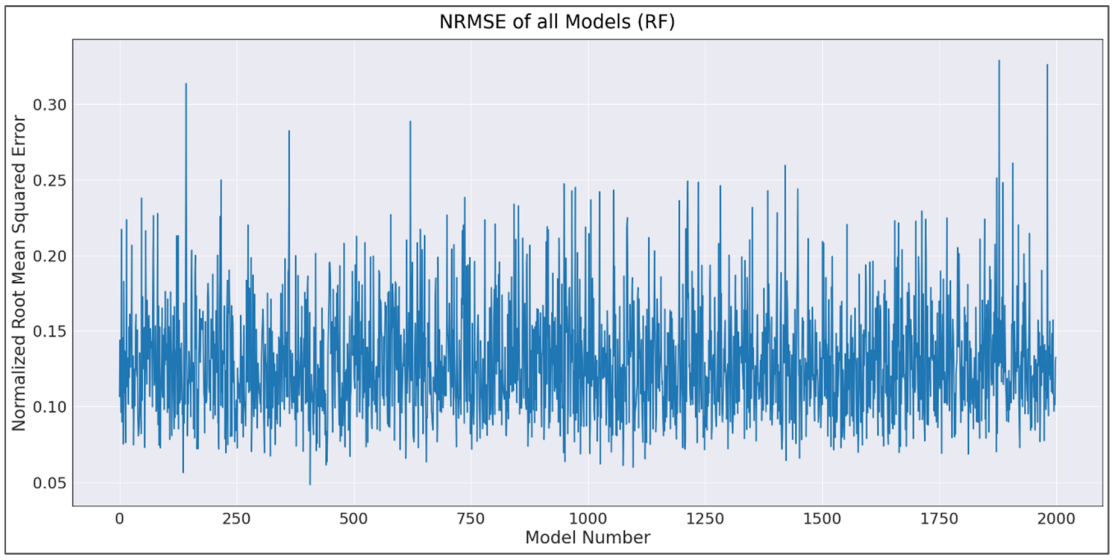

Figure 4.3 - Normalized Root Mean Squared Errors of all Simulation Models

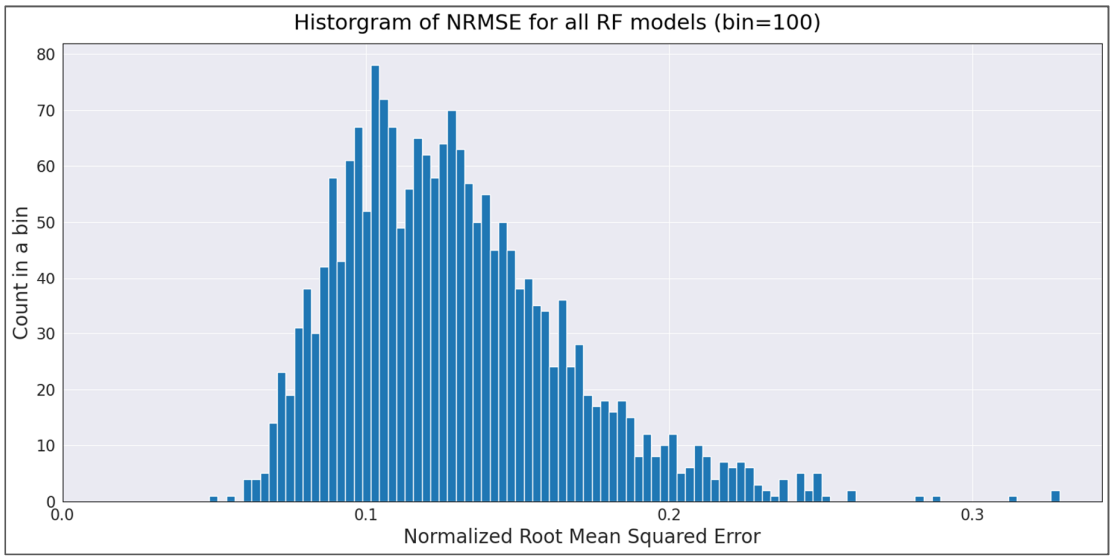

Figure 4.4 - Histogram of NRMSE for all models

Figure 4.3 shows the Root Mean Squared Error of each simulation model (composed of cell models). It is evident that the test data was homogenous enough, and the model's performance did not vary significantly throughout the dataset.

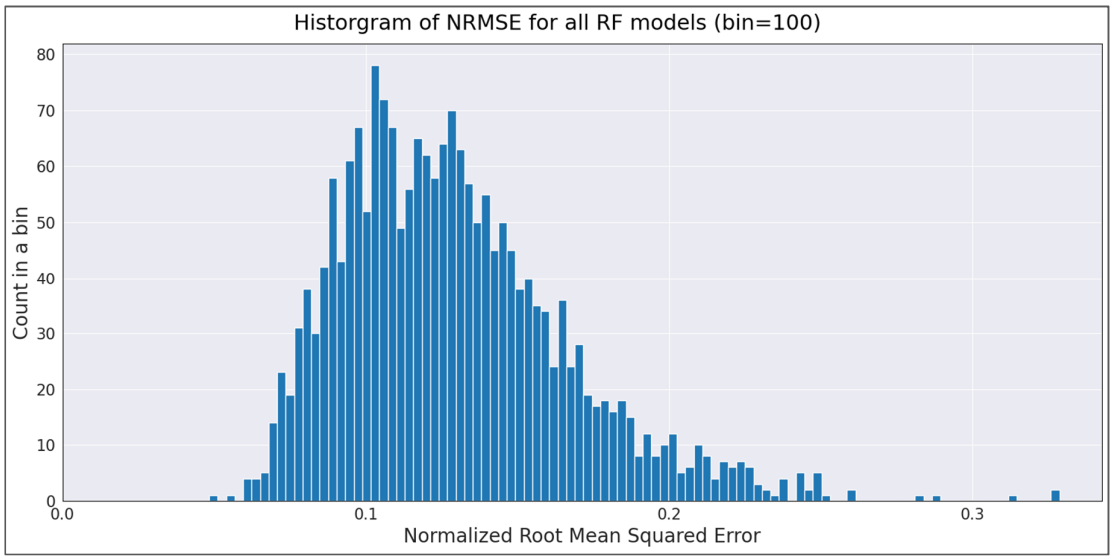

Figure 4.4 shows the histogram (bins=100) of the NRMSE for all simulation models.

When the simulation models are ranked according to their NRMSE values, model 420 is one of the top performers. In

Figure 4.5, the predictions of permeabilities vs. expected values of permeabilities of simulation model 420 are given. A total of nine different bands of permeability values could be seen in the figure, and the model's predictions vary but are acceptable. On the other hand, one of the worst performers, simulation model 142 (Figure 4.6), has predicted ten bands of permeability values. The model underestimates the higher permeability values while overestimating the permeability values with lower values. Figure 4.7 and Figure 4.8 show the expected permeabilities and the predicted permeabilities from model 420. In these figures, drastic variations of the permeability predictions of the cell models can be seen. The RF model can capture the general structure of the field to a degree and would help predict the permeability distributions for natural state modeling.

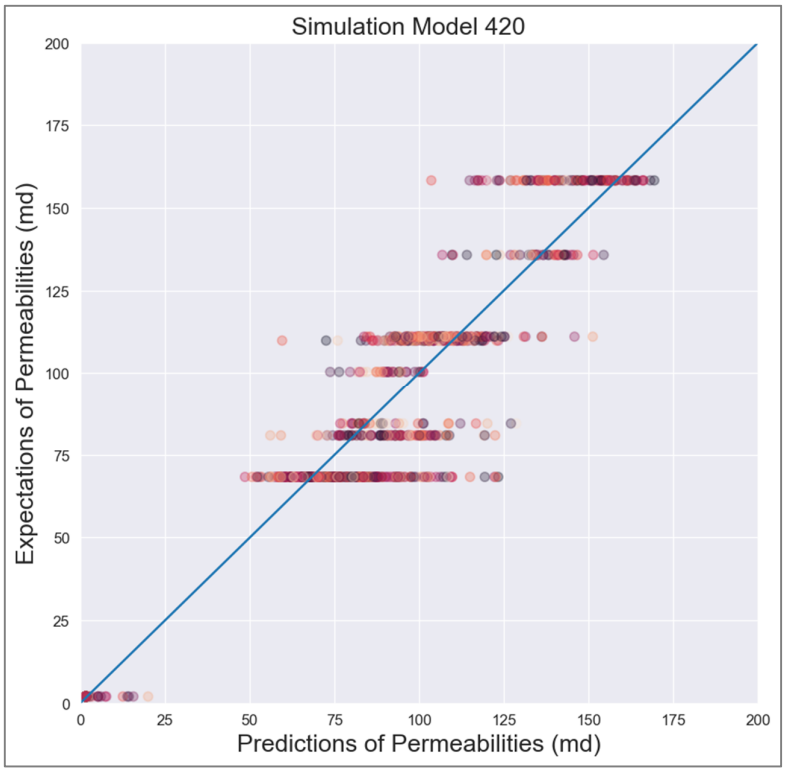

Figure 4.5 - Expected Permeabilities vs. Predicted Permeabilities - Simulation Model 420

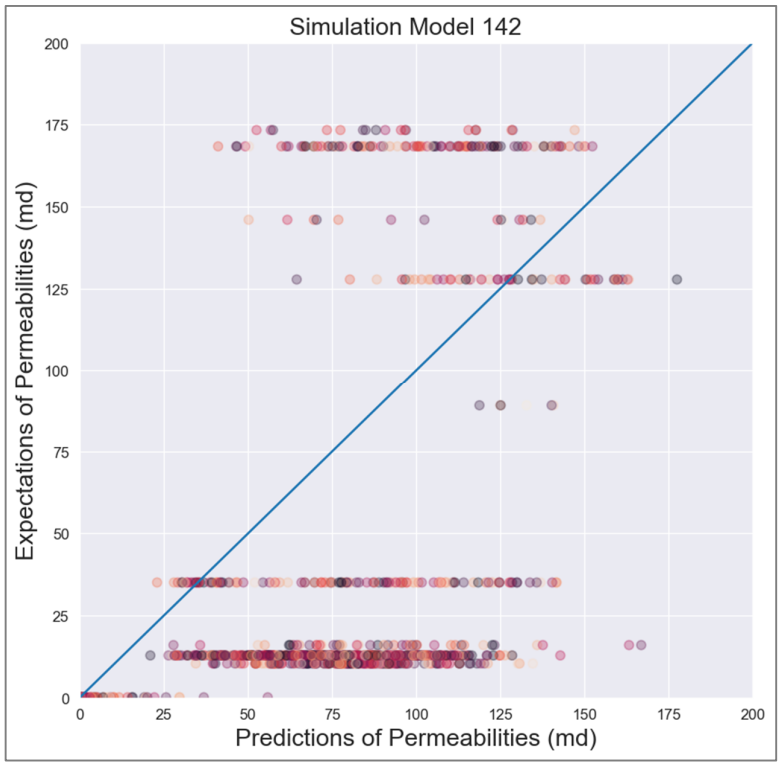

Figure 4.6 - Expected Permeabilities vs. Predicted Permeabilities - Simulation Model 142

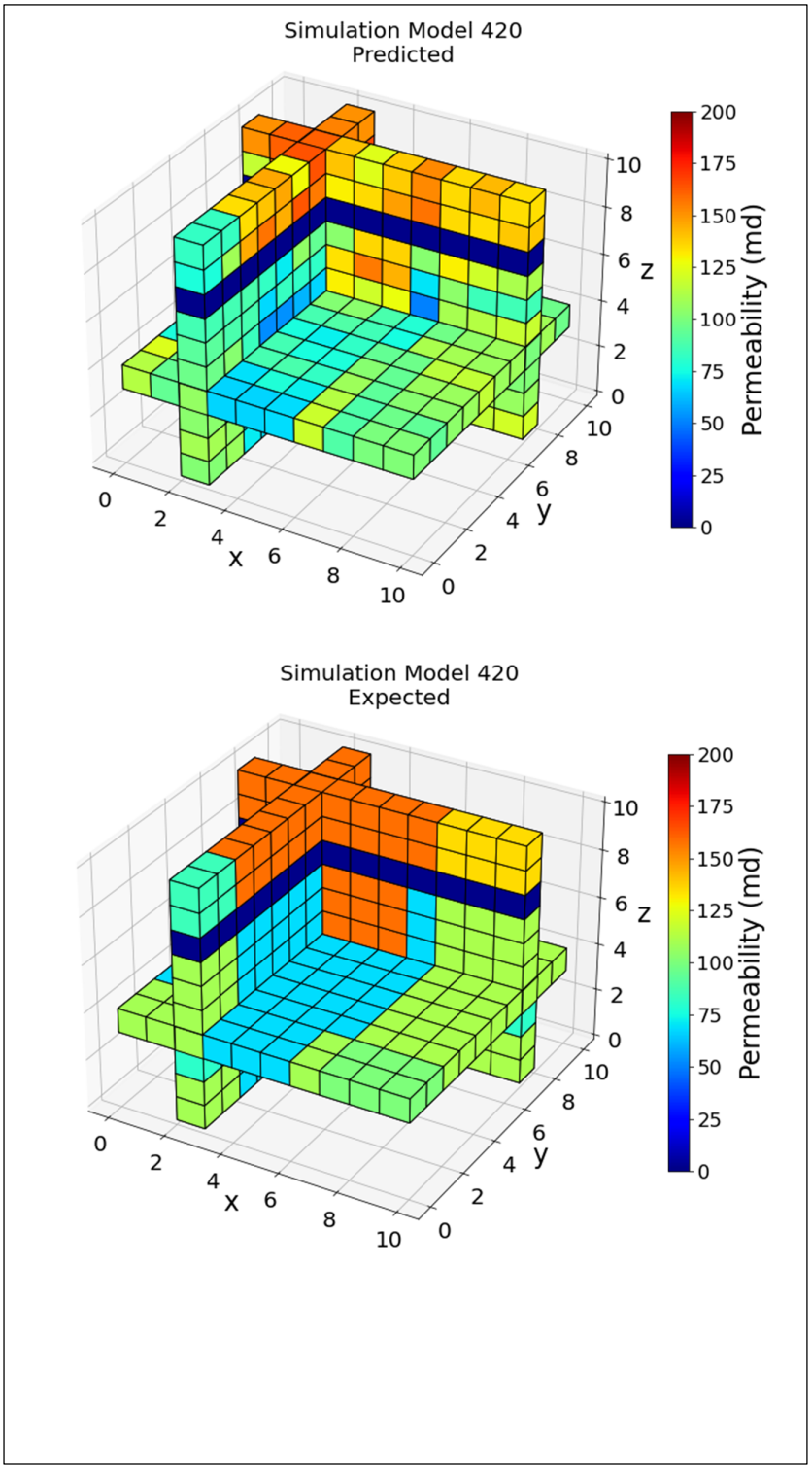

Figure 4.7 - 3D view of Predicted and Expected Permeability values of RF for Simulation Model 420

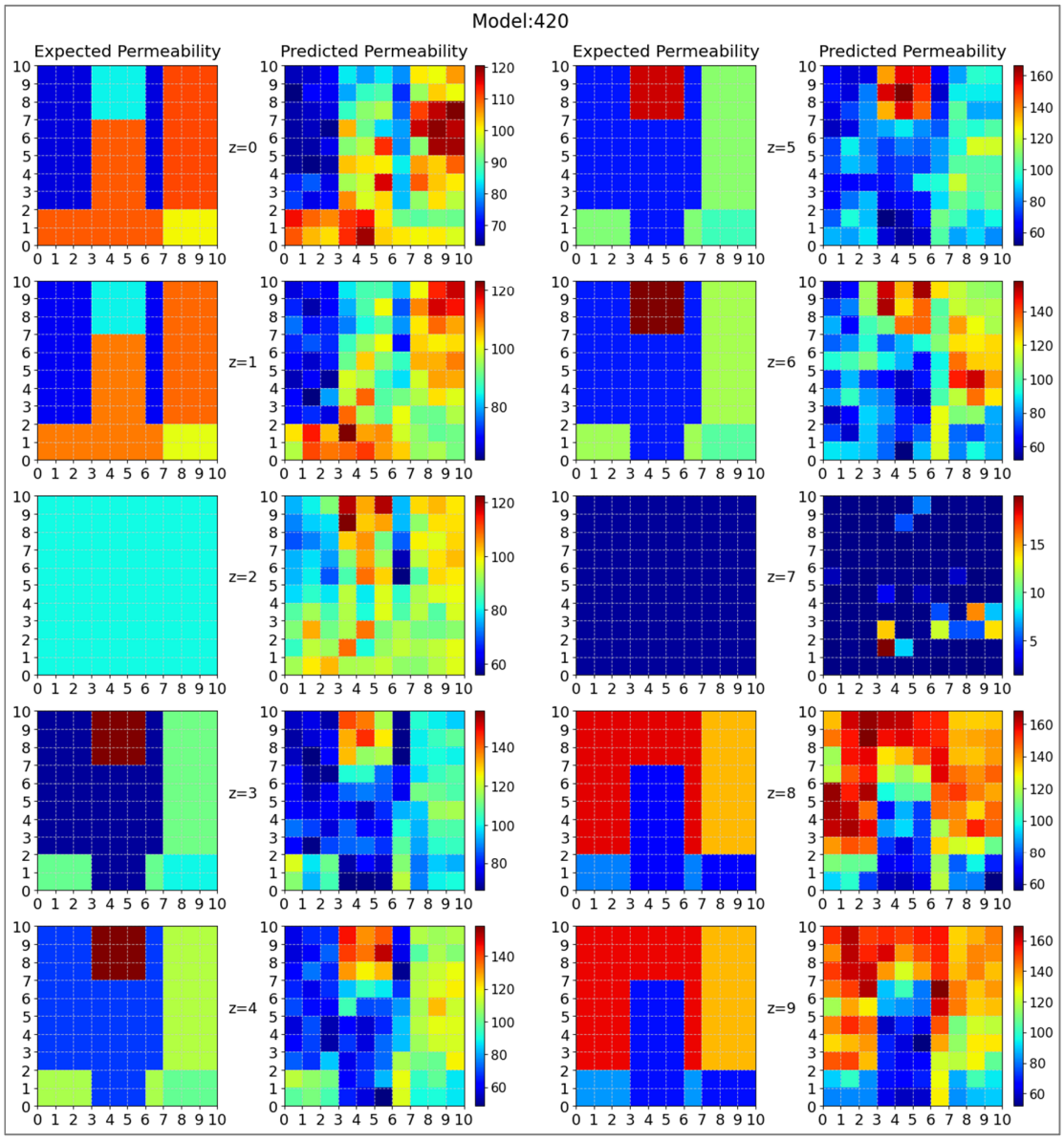

Figure 4.8 - Expected permeabilities and the Predicted permeabilities from model 420

4.3. Support Vector Regression Model and Results

In this part of the study, Support Vector Machines, one of the most popular and versatile tools in machine learning, is used to discuss further the prediction of permeability distributions of geothermal fields for natural state modeling from temperature and pressure distributions.

The dataset's contents are determined in previous chapters, and the best-performing dataset (T1-P3) was created. In order to build the SVR model, firstly, the validation set of the dataset is used in the hyperparameter tuning process. In order to find the best

hyperparameters for the SVR, RandomizedSearchCV, a module in the sckit-learn library, is used. The hyperparameters with best results in SVR is as follows (kernel='poly', C=100, gamma='auto', degree=3, epsilon=0.1, coef0=1).

To include the randomization of data to be used in training, increasing the performance of SVR, and utilizing the multiple processors of the computer the models are trained on, a bagging regressor of SVR is used with ten estimators. This bagging method uses a tuned SVR model and splitting the training data between the estimators, runs the model separately, and aggregates the results into the final decision. BaggingRegressor of skit-learn also uses bootstrapping, in which the samples from training data sent to the estimators are chosen from the population with replacements, thus making the process purely random.

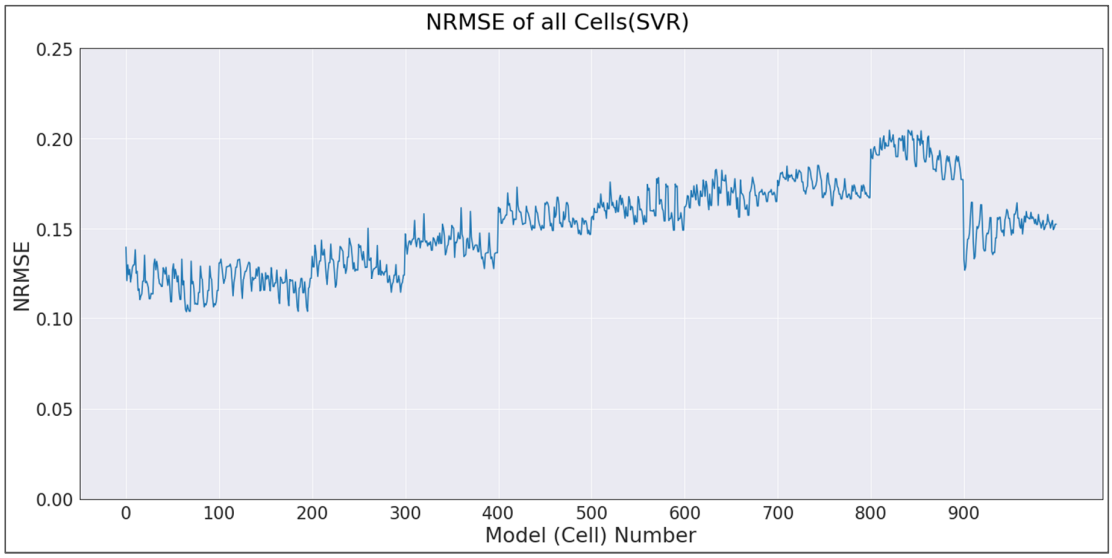

Figure 4.9 - Normalized Root Mean Squared Error of all Cell Models

After collecting the permeability predictions of each cell for each instance, a table is created. In this table, there were negative permeability predictions, which are physically meaningless. These negative values were replaced by zero as a post-processing step before calculating the root mean squared error. Later this RMSE is divided by the difference of the maximum and minimum permeability values to get the Normalized RMSE (NRMSE) metric to assess the method's success. In order to determine the general success of the SVR model, the NRMSE is calculated as 0.153.

In Figure 4.10, different predictions vs. the expected permeability values of 2,000 instances of one of the cell models, model 42, can be seen. This model generally overestimates, as is evident in the figure.

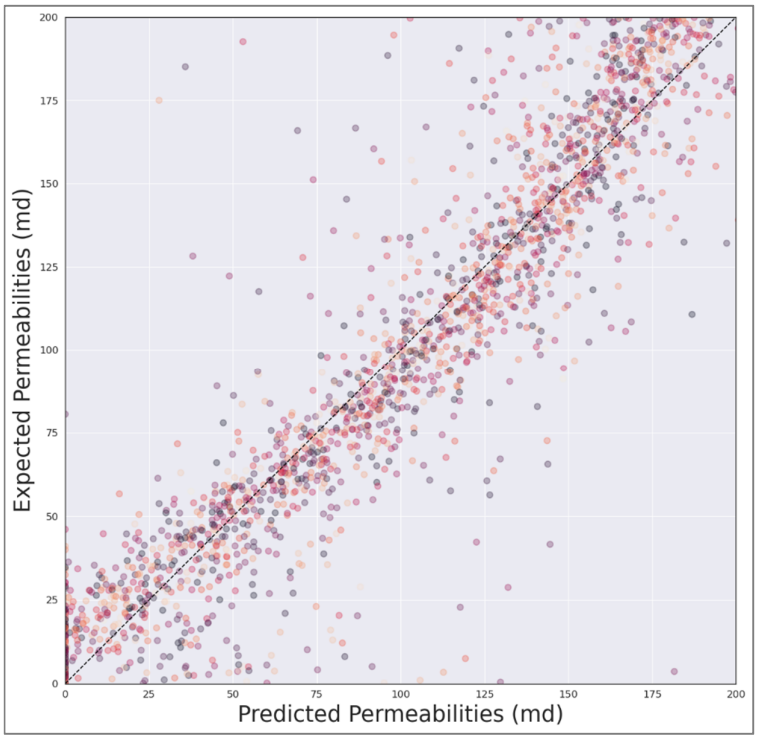

Figure 4.10 - Expected vs. Predicted Permeability values of cell 42

In Figure 4.9, the Normalized Root Mean Squared Errors of all cell models are shown. Similar to the results of Random Forest, predicting the permeabilities of the cells which are deep inside the model is more challenging than the ones that are close to boundaries. As previously mentioned, the reason for the sudden, step-like changes in the figure is from the changes in the depth of the layers, which happens in every 100 (10x10) cells. The figure shows us that the predictions of the first two or three layers are better than the rest. Starting from layer four, the cell models' predictions deteriorated with each new layer until reaching the top layer. The top layer, being under the direct influence of the outer cells of the geothermal model, could be predicted better from the temperature and pressure values. The NRMSE variations between the predictions on each layer are around 0.03, which is lower compared to the Random Forest model. Having lower variations in errors between cells close to each other would help the model to capture the general structure of the layer better and might help the model to produce results more compelling to the human eye, but at the same time would be slow to react to sudden changes such as constant permeability zones which would disturb the general pattern.

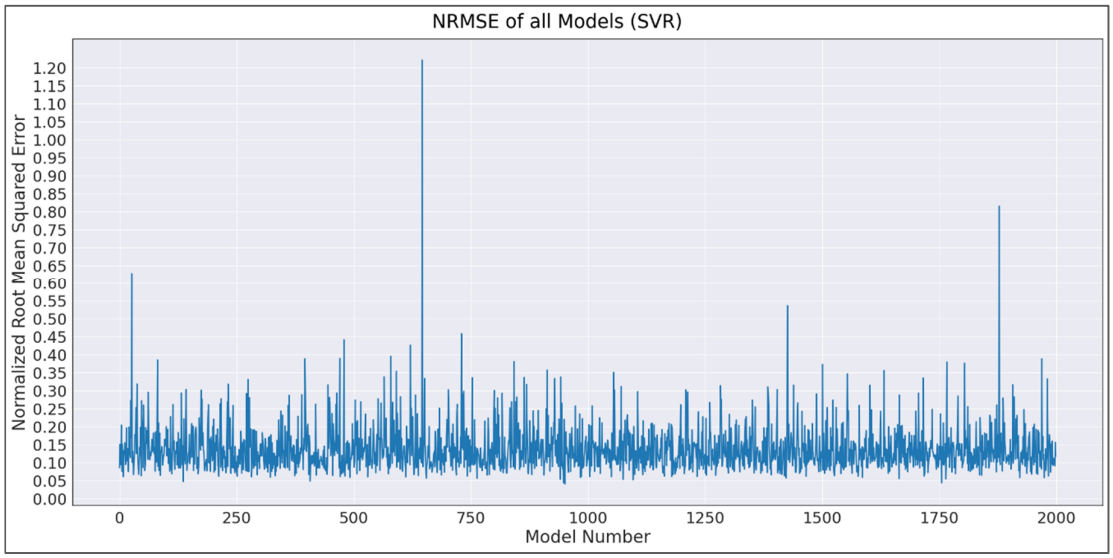

Figure 4.11 - Normalized Root Mean Squared Errors of all Simulation Models

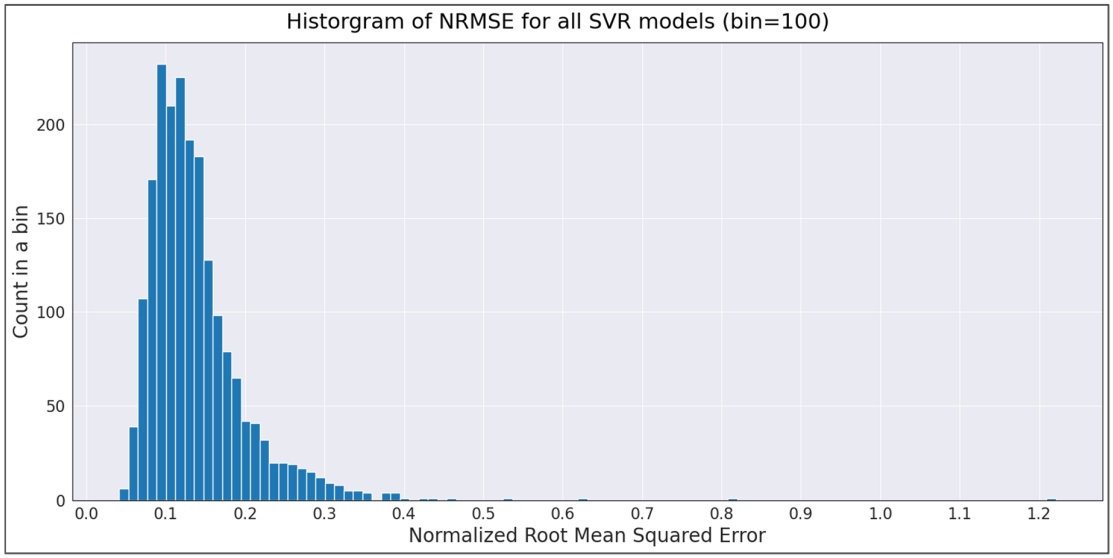

Figure 4.12 - Histogram of NRMSE for all SVR models

As previously mentioned, the combination of 1,000 individual cell models creates a simulation model. Since there are 2,000 instances in the dataset, there are also 2,000 different simulation models. The NRMSE of each simulation model is calculated, and as can be seen in Figure 4.11, the collection has four outliers with NRMSE values higher than 0.5. From the graph, it can also be seen that the model's performance did not vary significantly throughout the dataset.

The histogram of the NRMSE for all simulation models can be seen in Figure 4.12. Model 420 is again in the top 50 on good performers when the simulation models are ranked according to their success.

Figure 4.13 gives information about the success of simulation model 420. The geothermal model 420 has nine distinct permeability zones, and the predictions are at acceptable levels, but the model underestimated two major permeability zones. When it comes to poor performances, with an NRMSE score of 0.30, model 142 is again on the worst 50 performers list. In Figure 4.14, nine bands of permeability predictions, with significant underestimations of higher permeabilities and general overestimation of lower permeabilities.

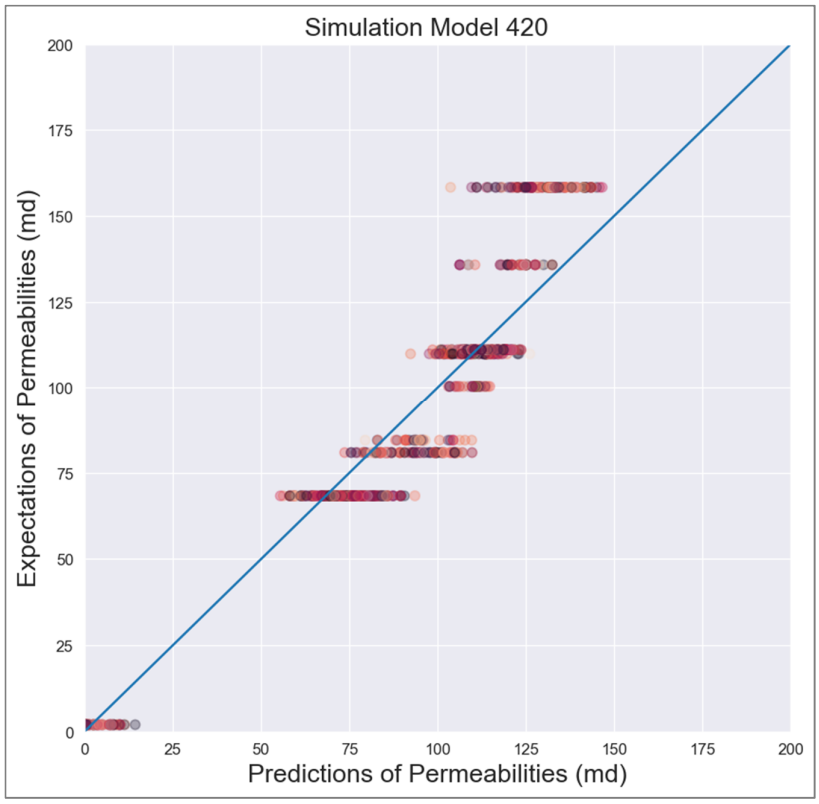

Figure 4.13 - Expected Permeabilities vs. Predicted Permeabilities - SVR Simulation Model 420

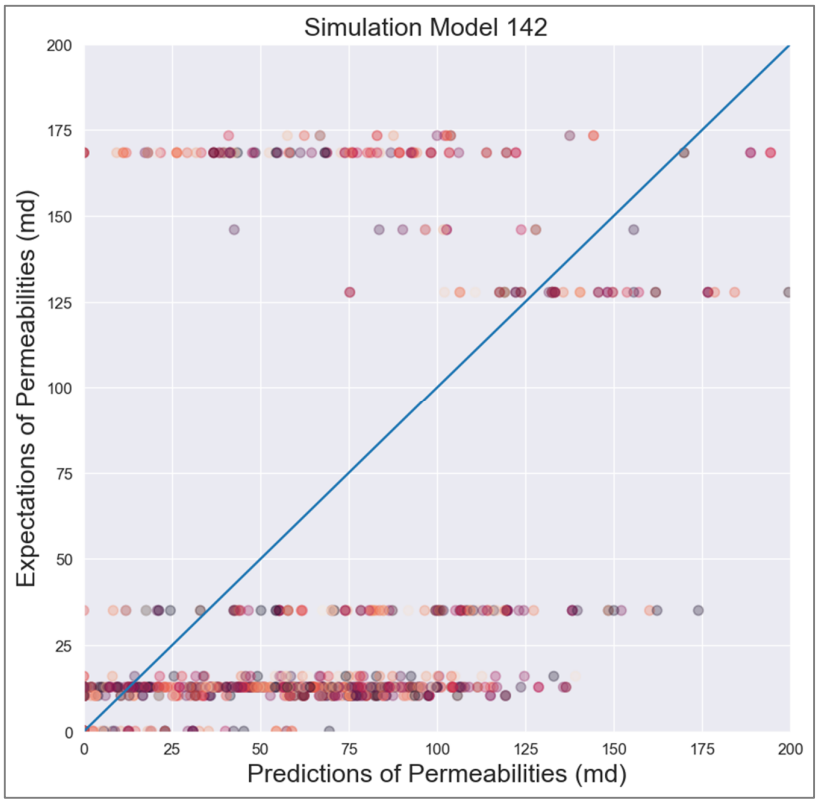

Figure 4.14 - Expected Permeabilities vs. Predicted Permeabilities - SVR Simulation Model 142

Figure 4.15 and Figure 4.16 show the expected permeabilities and the predicted permeabilities from model 420. The SVR model's predictions are not varying drastically between cells that are close to each other, which leads to better predictions on layers. However, as shown in the figure, the model performs poorly on constant permeability layers. Looking at these results, SVR can also help predict the permeability values for natural state modeling from temperature and pressure values.

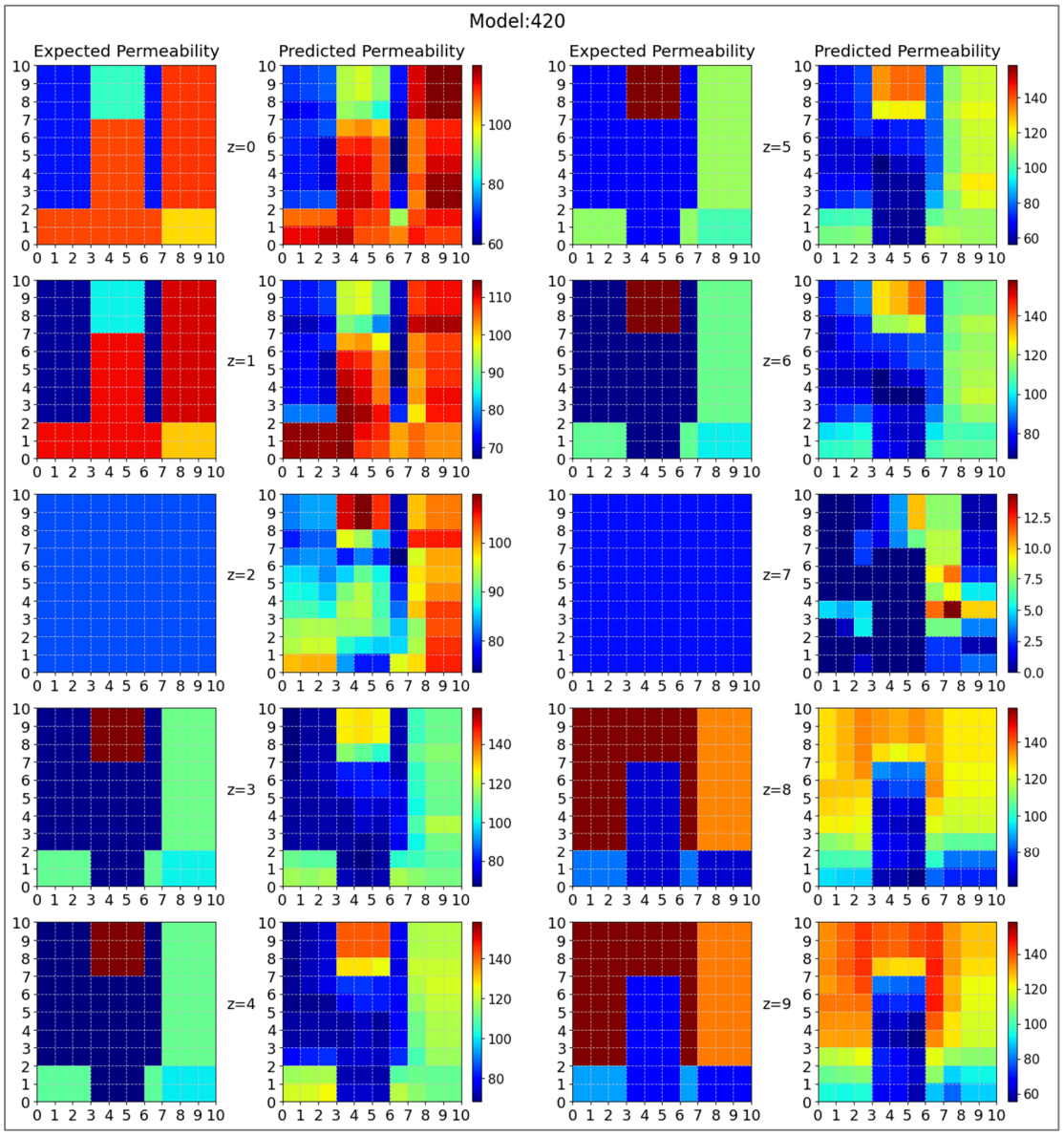

Figure 4.15 - Expected permeabilities and the Predicted permeabilities from SVR model 420

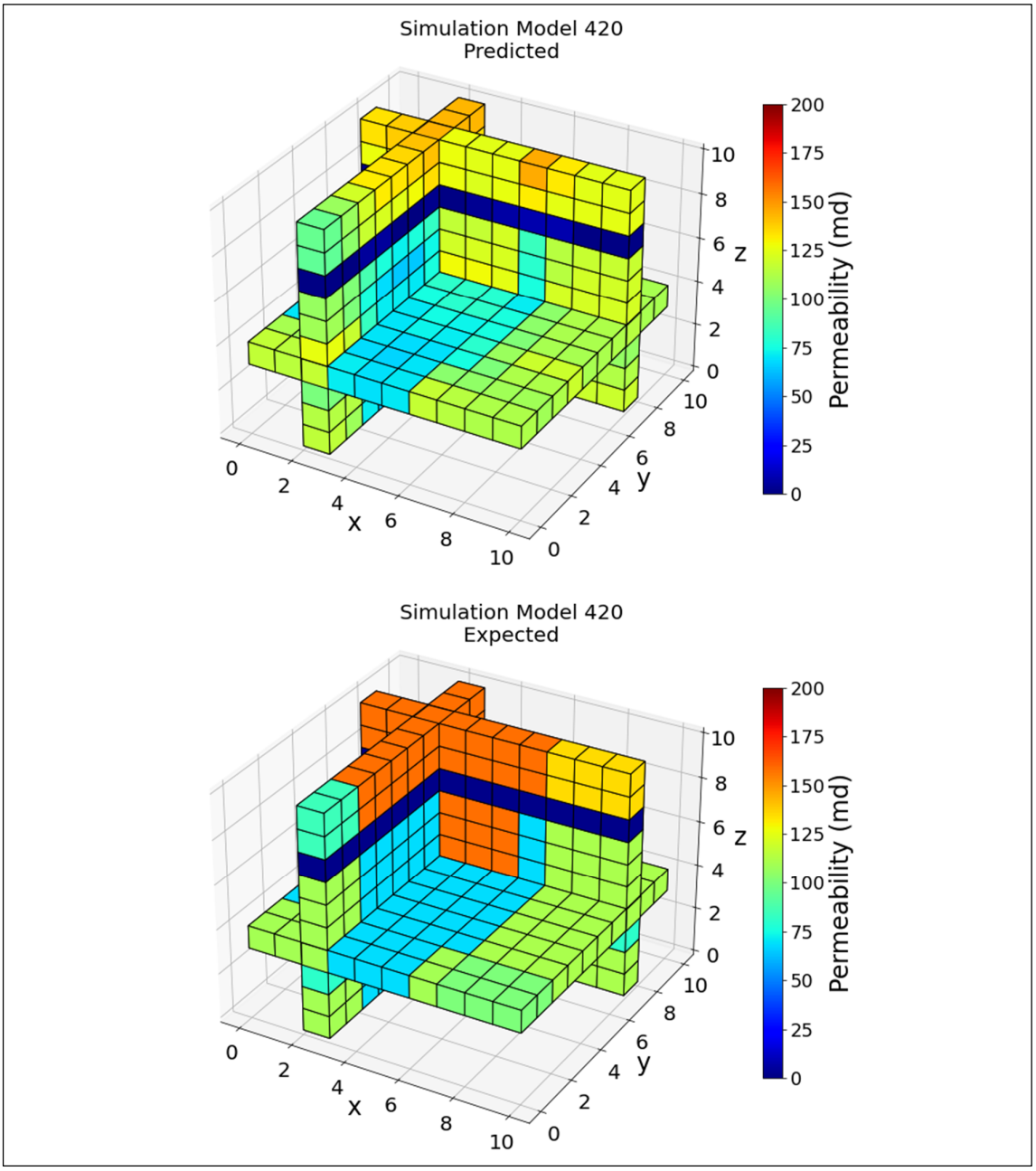

Figure 4.16 - 3D view of Predicted and Expected Permeability values for Simulation Model 420

4.4. Multilayer Perceptron model and Results

MLP is used in this part of the study to predict the permeability distributions of geothermal fields using temperature and pressure distributions. Using our T1-P3 dataset, different network structures are tested on a sample from the validation set to find the best-performing combination. We are using ReLU in hidden layers and linear activation function for the output of our regression problem. The best performing structure is found as 1000-500, as can be seen in Table 4.3.

Table 4.3- MLP structure experiments, the best score given in bold

Nodes	NRMSE of 23 cells
1000 - 500	**0.125**
2000-1000	0.126
500-200	0.126
300-200-100	0.126
500-500-500	0.126
250-100	0.128
300-200-100	0.130
500	0.131
50-20-10-1	0.140

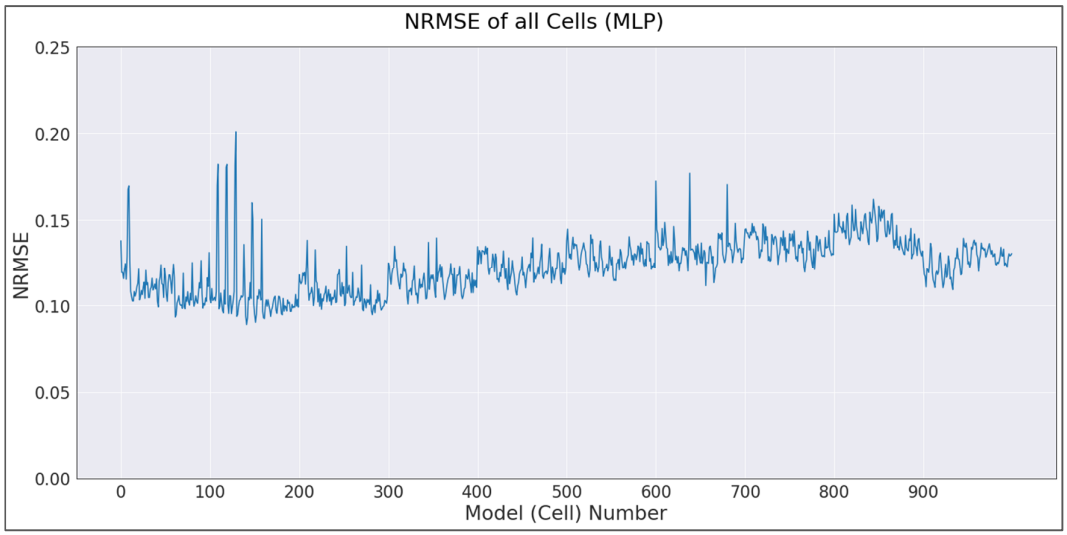

Figure 4.17 - Normalized Root Mean Squared Error of all Cell Models)

To calculate the success of the MLP models, normalized root mean squared error is used. To calculate the normalized root mean squared errors (NRMSE), the RMSE is first

calculated from the expected values and the predicted values of the permeability for all runs, then the calculated RMSE is divided with the difference of the maximum and minimum values of the observed permeability. The NRMSE of the MLP model is found as 0.123.

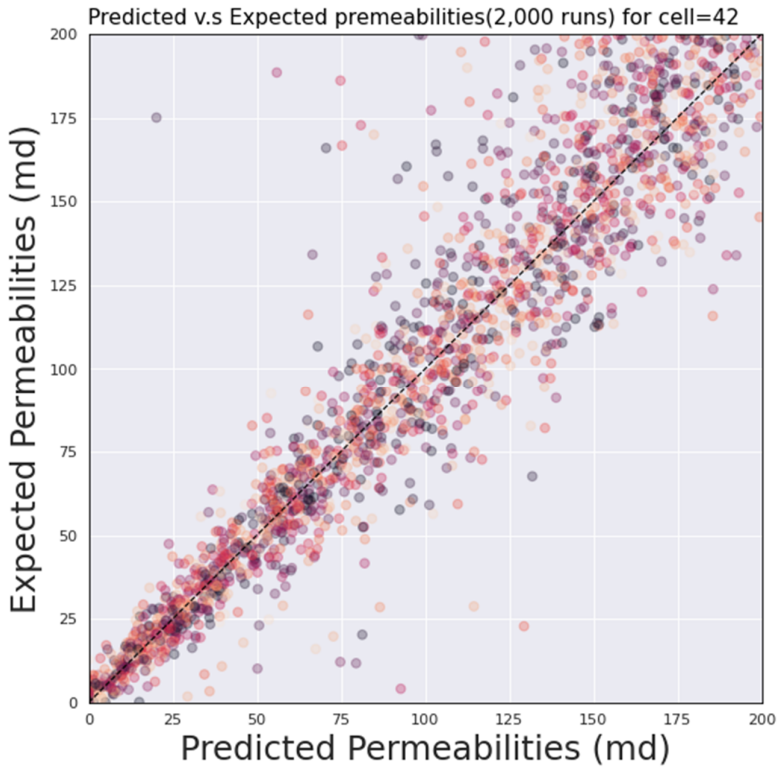

Figure 4.18 - Expected vs. Predicted Permeability values of cell 42

In Figure 4.18, predictions from the 2,000 instances of cell model 42 vs. the expected permeability values can be seen, and model 42 performed better on the prediction of lower permeability values.

In Figure 4.17, the Normalized Root Mean Squared Errors of all cell models are shown. As expected, similar to the previous results, predictions near the boundaries are more successful than those located deeper inside the model. The effects of layers can also be seen in every 100th cell because of the 10x10 structure of the model's layers with sharp changes in the values. The MLP model shows some sudden spikes, especially between the cells 100 to 200, but the number of spikes higher than 0.12 is only 10. Even with these outliers, the predictions of the first 300 layers are generally better than the rest of the model. Above cell 900, the expected drop in errors occurs due to being close to the boundaries and free from other layers' effects. Compared to RF and SVR, MLP performed better in general; the

difference of errors between layers is also smaller compared to these two models. The variations of the NMRSE between cells in the same layers are also better than RF, but even the errors, in general, are worse in SVR; it still shows fewer variations between errors and produces results more compelling to the human eye.

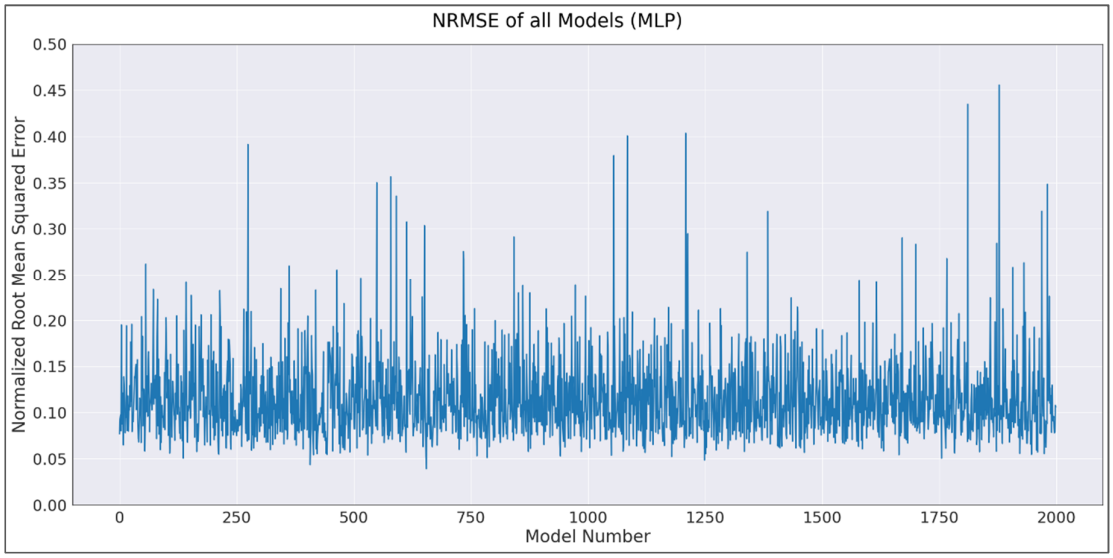

Figure 4.19 - Normalized Root Mean Squared Errors of all Simulation Models

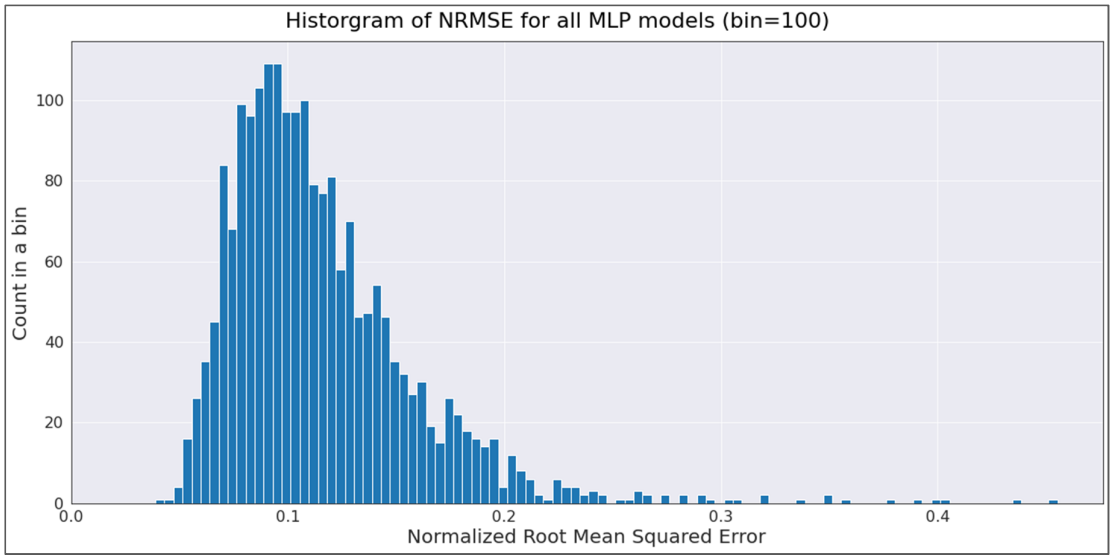

Figure 4.20 - Histogram of NRMSE for all SVR simulation models

Figure 4.19 illustrates the NRMSE of all the MLP simulation models. Each simulation model is a combination of 1,000 individual cell models. Even though there are some outliers

in the graph, they are not enough to distort the general performance of the model throughout the dataset, which is very homogeneous.

The histogram of the NRMSE for all simulation models can be seen in Figure 4.20. When the simulation models are ranked according to their errors, model 420 is 58th on good performers.

Figure 4.21 gives information about the success of simulation model 420. The nine different bands of permeability values can be seen on the graph. This does not mean there were nine different permeability zones on the model, but some regions have permeability distributions close to each other. Even though the model generally underestimates two permeability regions, the model's predictions are still not far from the expected values. Being 42th on the worst performers, model 142 is still one of the poor performers, as illustrated in Figure 4.22. This graph has ten bands of permeability values with very different levels, and most of the expected values are low.

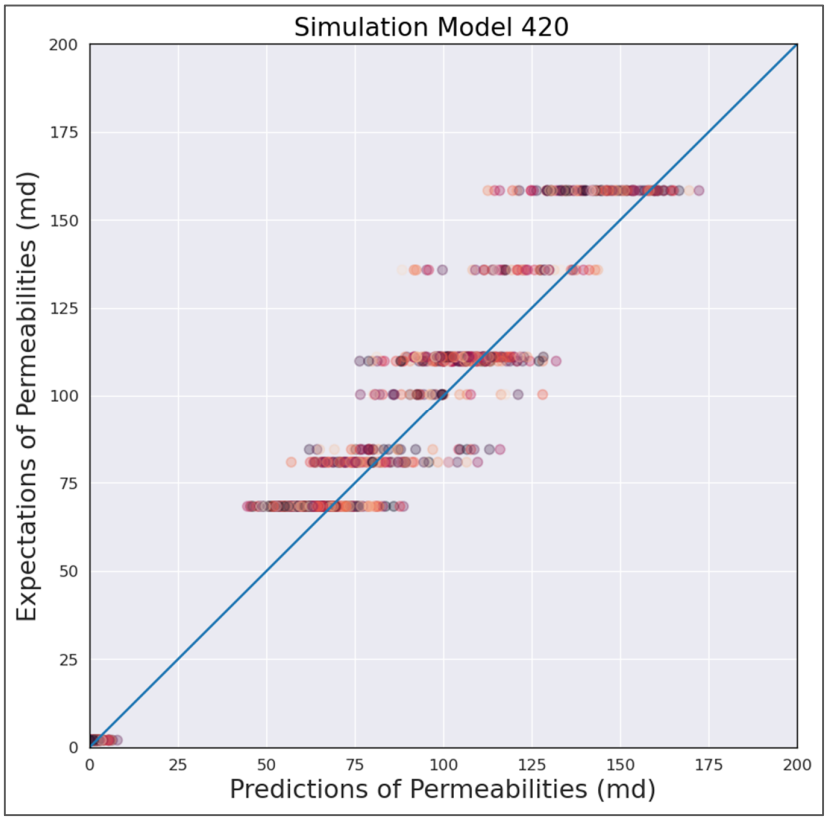

Figure 4.21 - Expected Permeabilities vs. Predicted Permeabilities - MLP Simulation Model 420

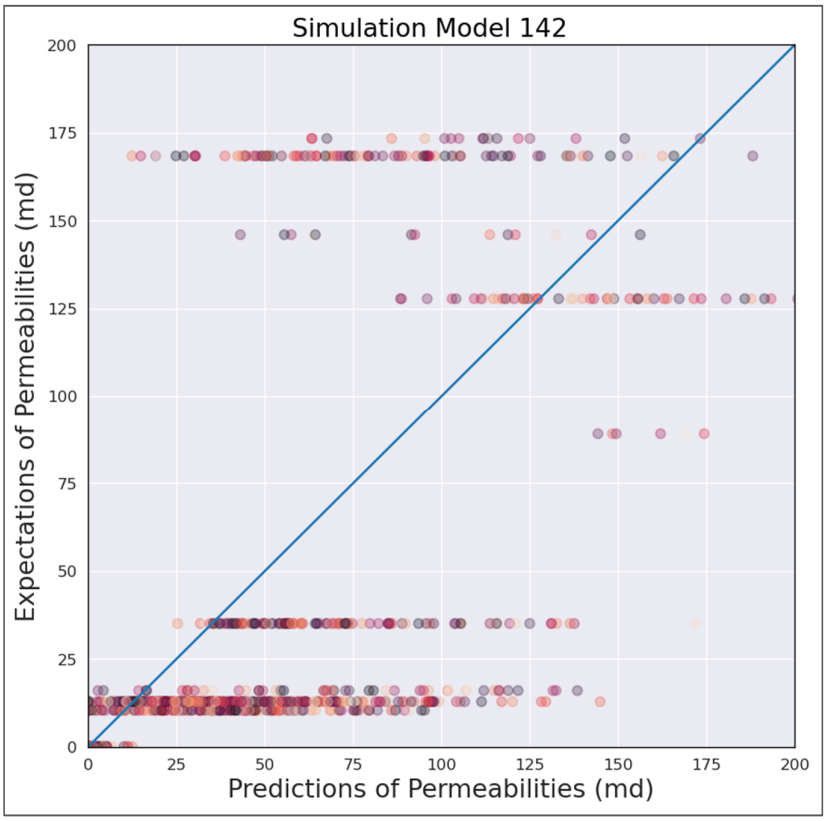

Figure 4.22 - Expected Permeabilities vs. Predicted Permeabilities - MLP Simulation Model 142

Figure 4.23 and Figure 4.24 illustrate the predicted and expected permeability distributions of different layers from model 420. Even though the effects of the upper and lower levels can influence the values on constant permeability zones, the MLP model manages to capture the general structure of the field successfully.

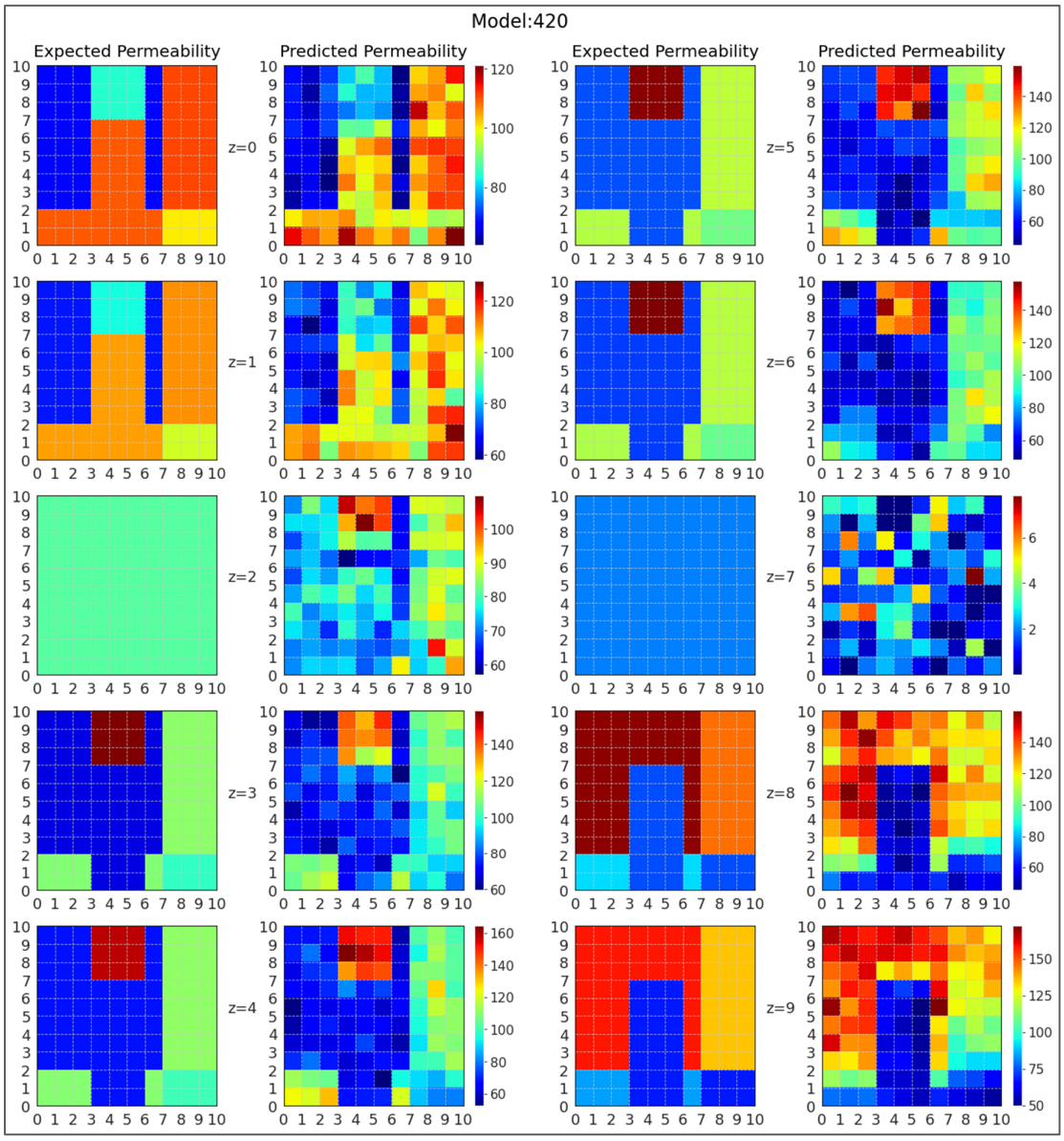

Figure 4.23 - Expected permeabilities and the Predicted permeabilities from MLP model 420

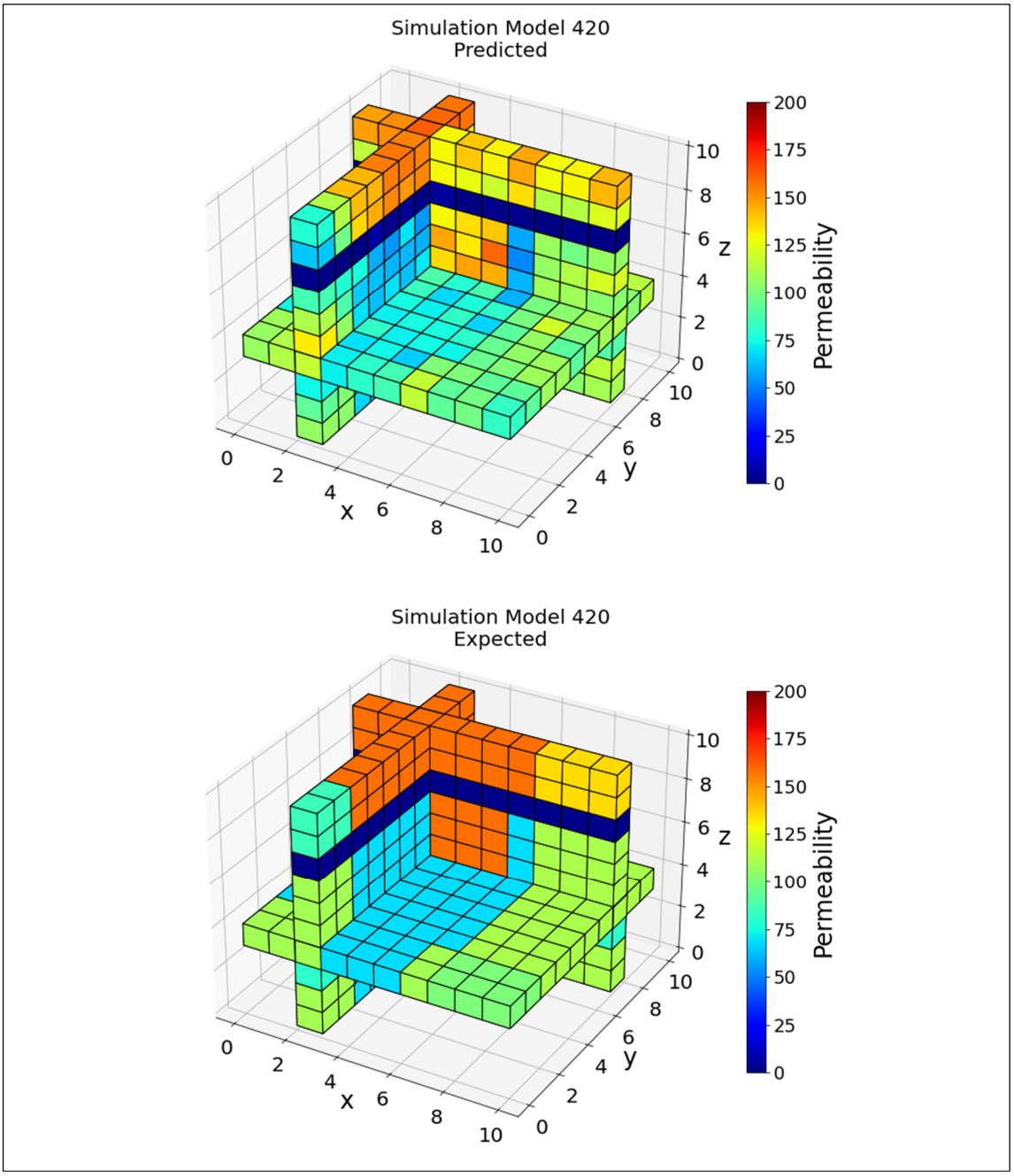

Figure 4.24 - 3D view of Predicted and Expected Permeability values for Simulation Model 420

4.5. CNN model and Results

Convolutional Neural Networks have proven to be very robust and versatile while dealing with many tasks, primarily image classification and processing tasks. Our problem, the determination of permeability values of geothermal fields on natural state modeling, can also be formulated as an image recognition and processing problem.

In order to prepare the data for CNN, we need to create an image of the field's temperature and pressure distributions, and for each cell, train the permeability values corresponding to these images. Because of its previously discussed success in earlier sections, the One Neighborhood Temperature (T1) dataset is used first to assess the viability of CNN in this problem. T1 training dataset has 6,000 instances with 1,000 cells and their corresponding permeability distributions. Since the geothermal model comprises ten layers with 10x10 cells (grids) on each layer, the dataset is reshaped from 6,000x1000 to 6,000x(10x10x10) to resemble the model better. Following this reshaping, the T1 training data then rearranged to stack each instance's z, y, and x layers vertically, thus creating an array with dimensions of 100x30 for each instance of 6,000, as illustrated in Figure *4.27*. The final T1 training dataset for CNN has the dimensions of 6,000x3000. This process was also repeated for the validation and test datasets of T1.

After preparing the datasets for CNN, different architectures were tried to be used for the problem in TensorFlow, a python library for deep learning [29]. The architecture shown in Figure 4.26 is determined by modifying the fundamental LeNet-5 structure by testing different combinations of convolution layers with different activation functions, pooling layers, and fully connected layers with different activation functions, as illustrated in

Figure 4.25. The final structure of the model has three 2d convolutional layers with the size of 32/32/64 and the kernel size of (3,3) and activation functions of ReLU. After these convolutional layers, a max-pooling layer of 2x2 is attached. The output of the max-pooling layer is then flattened to be fed into the fully connected dense layers of 300/200/100, ReLU. The model then finalizes as an output layer with the linear activation function. The dataset is also trained with AlexNet and Vgg-16 architecture, but the model created by the trial and error method performed better.

Because of its widespread usage and success, the model's optimizer is set as Adam. The hyperparameters are also determined by trial and error, and only the learning rate is changed to 0.9 from the default value of 0.001. With the addition of the Keras function [30]

ReduceLROnPlateau, a relatively high learning rate can be used. It decreases the learning rate of the model by 0.2 if for five consecutive epochs the validation loss does not bet better, until the learning rate is back to the default value of 0.001. An early stopping function is also used to control the model by setting the minimum decrease in validation loss needed as 0.001 after ten consecutive epochs.

C= Conv2d, mp=Max Pooling , F=Flatten, D = Dense , Lr= Learning rate, b=Batch, dp=Drop Out																	NRMSE
C 32 (3,3)	C 32 (3,3)	C 64 (3,3)				mp (2,2)	F	D 300	D 200	D 100				lr 0.9	b 60	**0.1010**
C 32 (3,3)	C 32 (3,3)	C 64 (3,3)				mp (2,2)	F	D 1500	D 1500	D 1500				lr 0.9	b 60	0.1011
C 32 (3,3)	C 32 (3,3)	C 64 (3,3)				mp (2,2)	F	D 300	D 200	D 100	x2			lr 0.9	b 64	0.1020
C 32 (3,3)	C 32 (3,3)	C 64 (3,3)				mp (2,2)	F	D 300	D 200	D 100	x2			lr 0.9	b 16	0.1022
C 32 (3,3)	C 32 (3,3)	C 64 (3,3)				mp (2,2)	F	D 300	D 200	D 100	x2			lr 0.9	b 32	0.1025
C 32 (3,3)	C 32 (3,3)	C 64 (3,3)				mp (2,2)	F	D 400	D 300	D 300	x2			lr 0.9	b 64	0.1031
C 32 (3,3)	C 32 (3,3)	C 64 (3,3)				mp (2,2)	F	D 300	D 200	D 100				lr 0.9	b 50	0.1032
C 32 (3,3)	C 32 (3,3)	C 64 (3,3)				mp (2,2)	F	D 400	D 200	D 200	D 100	x2		lr 0.9	b 64	0.1033
C 32 (3,3)	C 32 (3,3)	C 64 (3,3)				mp (2,2)	F	D 500	D 500	D 500				lr 0.9	b 64	0.1039
C 32 (3,3)	C 32 (3,3)	C 64 (3,3)				mp (2,2)	F	D 300	D 600	D 900				lr 0.9	b 60	0.1040
C 32 (3,3)	C 32 (3,3)	C 64 (3,3)				mp (2,2)	F	D 300	D 200	D 100	x2			lr 0.9	b 128	0.1042
C 32 (3,3)	C 32 (3,3)	C 64 (3,3)				mp (2,2)	F	D 900	D 600	D 900				lr 0.9	b 60	0.1045
C 32 (3,3)	C 32 (3,3)	C 64 (3,3)				mp (2,2)	F	D 300	D 200					lr 0.9	b 60	0.1045
C 32 (3,3)	C 32 (3,3)	C 64 (3,3)				mp (2,2)	F	D 300	D 200	D 900				lr 0.9	b 60	0.1048
C 32 (3,3)	C 32 (3,3)	C 64 (3,3)				mp (2,2)	F	D 500						lr 0.9	b 64	0.1049
C 32 (3,3)	C 64 (3,3)					mp (2,2)	F	D 300	D 200	D 100	x2			lr 0.9	b 64	0.1050
C 32 (3,3)	C 32 (3,3)	C 64 (3,3)				mp (2,2)	F	D 800	D 600	D 400				lr 0.9	b 60	0.1050
C 32 (3,3)	C 32 (3,3)	C 64 (3,3)				mp (2,2)	F	D 400	D 300	D 200				lr 0.9	b 60	0.1051
C 32 (3,3)	C 32 (3,3)	C 64 (3,3)				mp (2,2)	F	D 300	D 200	D 100	x2			lr 900	b 64	0.1055
C 16 (3,3)	C 16 (3,3)	C 64 (3,3)				mp (2,2)	F	D 300	D 200	D 100	x2			lr 0.9	b 64	0.1058
C 32 (3,3)	C 32 (3,3)	C 64 (3,3)				mp (2,2)	F	D 300	D 200	D 100				lr 0.9	b 64	0.1063
C 32 (3,3)	C 32 (3,3)	C 64 (3,3)				mp (2,2)	F	D 300	D 200	D 100			dp(0,1)	lr 0.9	b 60	0.1063
C 32 (3,3)	C 32 (3,3)	C 64 (3,3)				mp (2,2)	F	D 300	D 200	D 100			dp(0,2)	lr 0.9	b 60	0.1065
C 32 (3,3)	C 32 (3,3)	C 64 (3,3)				mp (2,2)	F	D 300	D 200	D 100				lr 0.9	b 64	0.1069
C 32 (3,3)	C 32 (3,3)	C 64 (3,3)				mp (2,2)	F	D 300	D 200	D 100	x2			lr 0.9	b 16	0.1069
C 16 (3,3)	C 32 (3,3)	C 64 (3,3)				mp (2,2)	F	D 300	D 200	D 100	x2			lr 0.9	b 64	0.1074
C 32 (3,3)	C 32 (4,4)	C 64 (3,3)				mp (2,2)	F	D 300	D 200	D 100	x2			lr 0.9	b 64	0.1075
C 32 (3,3)	C 32 (3,3)	C 64 (3,3)				mp (2,2)	F	D 100	D 100	D 100				lr 0.9	b 64	0.1078
C 32 (3,3)	C 32 (3,3)	C 64 (3,3)				mp (2,2)	F	D 300	D 200	D 100	x3			lr 0.9	b 64	0.1085
C 32 (2,2)	C 32 (3,3)	C 64 (4,4)				mp (2,2)	F	D 300	D 200	D 100	x2			lr 0.9	b 64	0.1086
C 32 (3,3)	C 32 (3,3)	C 64 (3,3)				mp (2,2)	F	D 200	D 200	D 200				lr 0.9	b 60	0.1086
C 32 (3,3)	C 32 (3,3)	C 64 (3,3)				mp (2,2)	F	D 300	D 200	D 100				lr 0.9	b 100	0.1095
C 32 (3,3)	C 32 (3,3)	C 64 (3,3)				mp (2,2)	F	D 100	D 100	D 100	x2			lr 0.9	b 64	0.1098
C 16 (3,3)	C 16 (3,3)	C 32 (3,3)				mp (2,2)	F	D 300	D 200	D 100	x2			lr 0.9	b 64	0.1101
C 32 (3,3)	C 32 (3,3)	C 64 (3,3)				mp (2,2)	F	D 1500						lr 0.9	b 60	0.1108
C 32 (3,3)	C 64 (3,3)					mp (3,3)	F	D 300	D 200	D 100	x2			lr 0.9	b 64	0.1127
C 32 (3,3)	C 32 (3,3)	C 64 (3,3)				mp (2,2)	F	D 300	D 200	D 100	x2			lr 0.9	b 8	0.1138
C 32 (3,3)	C 32 (3,3)	C 64 (3,3)	C 64 (3,3)	C 128 (3,3)		mp (2,2)	F	D 400	D 300	D 200	D 100	x2		lr 0.9	b 64	0.1152
C 64 (3,3)						mp (1,1)	F	D 100	D 100	D 100				lr 0.9	b 64	0.1162
C 64 (3,3)						mp (1,1)	F	D 300	D 200	D 100				lr 0.9	b 64	0.1170
C 16 (3,3)	C 16 (3,3)	C 32 (3,3)	C 32 (3,3)	C 64 (3,3)		mp (2,2)	F	D 300	D 200	D 100	x2			lr 0.9	b 64	0.1176
C 32 (3,3)	C 32 (3,3)	C 32 (3,3)	C 32 (3,3)	C 32 (3,3)		mp (2,2)	F	D 300	D 200	D 100	x2			lr 0.9	b 64	0.1183
C 32 (3,3)	C 32 (3,3)	C 64 (3,3)	mp (2,2)	C 64 (3,3)	C 64 (3,3)	mp (2,2)	F	D 300	D 200	D 100	x2			lr 0.9	b 64	0.1209
C 32 (6,6)	C 32 (6,6)	C 64 (3,3)				mp (2,2)	F	D 300	D 200	D 100	x2			lr 0.9	b 64	0.1218
C 32 (3,3)	C 64 (3,3)	mp (2,2)	C 32 (3,3)	C 32 (3,3)		mp (2,2)	F	D 300	D 200	D 100	x2			lr 0.9	b 64	0.1233
C 64 (3,3)						mp (1,1)	F	D 100						lr 0.9	b 64	0.1242
C 32 (3,3)	C 32 (3,3)	C 64 (3,3)	C 64 (3,3)	C 128 (3,3)	C 128 (3,3)	mp (2,2)	F	D 300	D 200	D 100	x2			lr 0.9	b 64	0.1268
C 32 (3,3)	mp (2,2)	C 32 (3,3)	mp (2,2)	C 64 (3,3)		mp (2,2)	F	D 100	D 100	D 100				lr 0.9	b 64	0.1332
C 32 (3,3)	mp (2,2)	C 32 (3,3)	mp (2,2)	C 64 (3,3)		mp (2,2)	F	D 100	D 100	D 100			dp (0,5)	lr 0.9	b 64	0.1387

Figure 4.25 - Experiments for CNN structure, the best score is given in bold on top

```
Model: "sequential_30"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_90 (Conv2D)           (None, 98, 58, 32)        320
_________________________________________________________________
conv2d_91 (Conv2D)           (None, 96, 56, 32)        9248
_________________________________________________________________
conv2d_92 (Conv2D)           (None, 94, 54, 64)        18496
_________________________________________________________________
max_pooling2d_30 (MaxPooling (None, 47, 27, 64)        0
_________________________________________________________________
flatten_30 (Flatten)         (None, 81216)             0
_________________________________________________________________
module_wrapper_104 (ModuleWr (None, 300)               24365100
_________________________________________________________________
module_wrapper_105 (ModuleWr (None, 200)               60200
_________________________________________________________________
module_wrapper_106 (ModuleWr (None, 100)               20100
_________________________________________________________________
module_wrapper_107 (ModuleWr (None, 1)                 101
=================================================================
Total params: 24,473,565
Trainable params: 24,473,565
Non-trainable params: 0
```

Figure 4.26 - CNN architecture

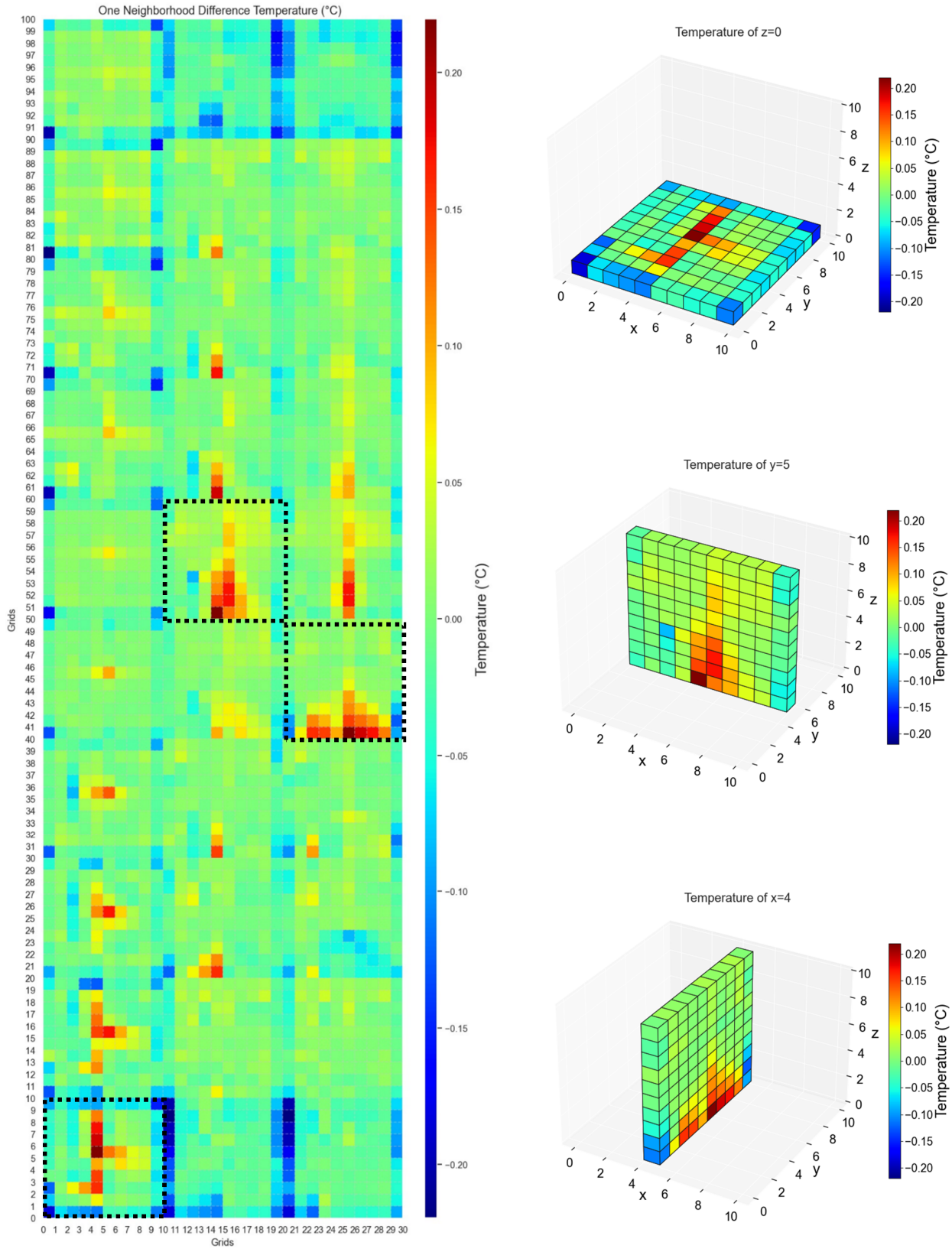

Figure 4.27 - CNN training data sample from T1 dataset (left). 3d visualizations of the sample layers (right)

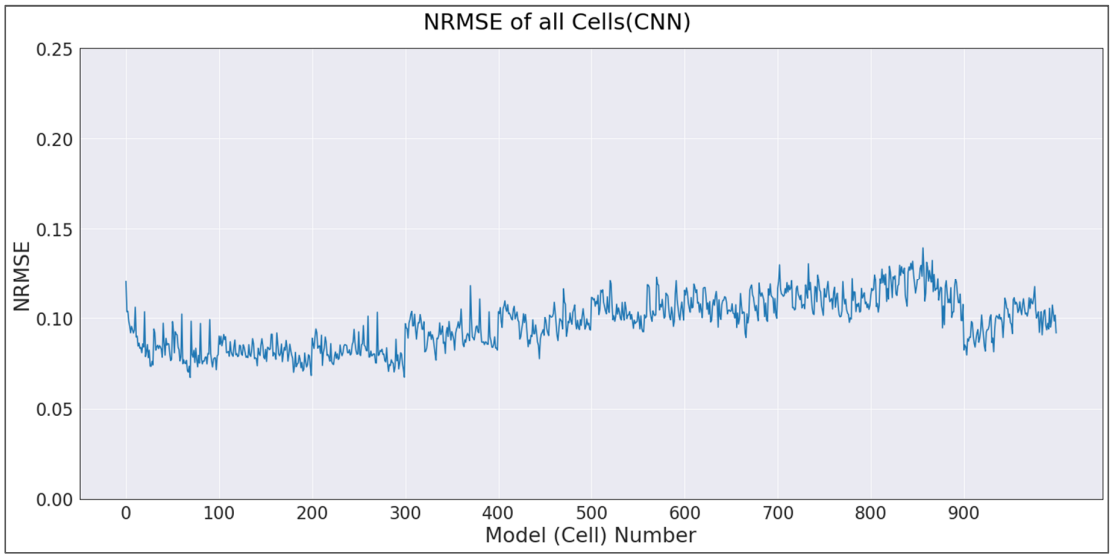

Figure 4.28 - Normalized Root Mean Squared Error of all Cell Models

Normalized root mean squared error is used to assess the success of the CNN models. As mentioned previously, the errors are first calculated from the expected values and the predicted values of the permeability for all runs, then the calculated RMSE is divided with the difference of the maximum and minimum values of the observed permeability. The NRMSE of the CNN model is found as 0.098.

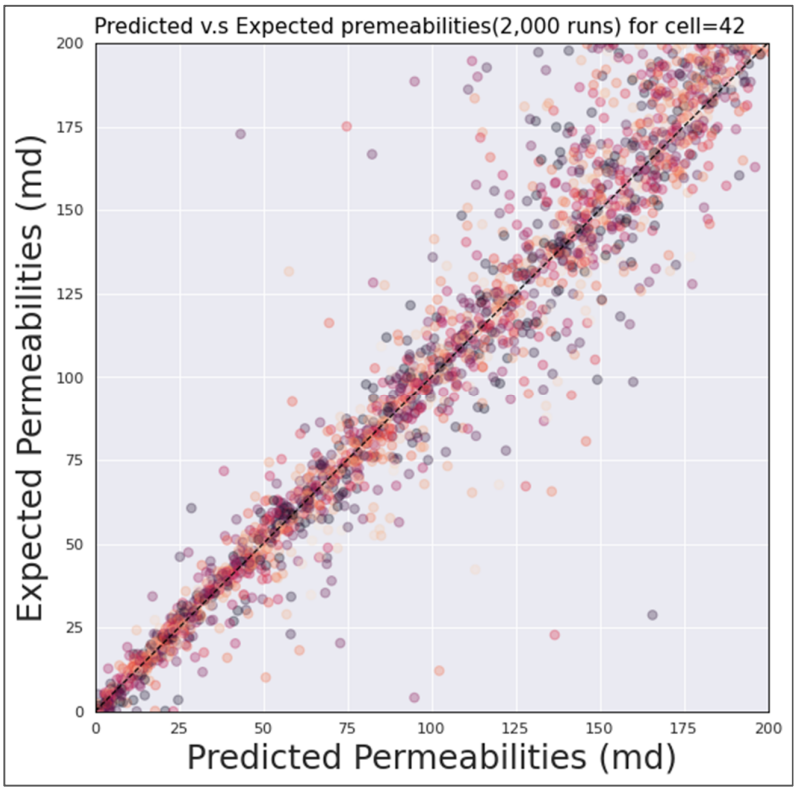

Figure 4.29 - Expected vs. Predicted Permeability values of cell 42

In Figure 4.29, predictions from the 2,000 instances of cell model 42 vs. the expected permeability values can be seen. The graph shows a good agreement between the predicted and expected values.

The difference of errors between the cells that are close to the boundaries and those located deeper into the model are much smaller than RF, SVR of MLP, as illustrated in Figure 4.28. As mentioned before, the sudden increases in the graph are due to the model's layered (10x10) structure. The CNN model's overall performance does not differ significantly from cell to cell, and in general, the predictions are very accurate. The variations of the NMRSE between cells in the same layers are also minor compared to other models.

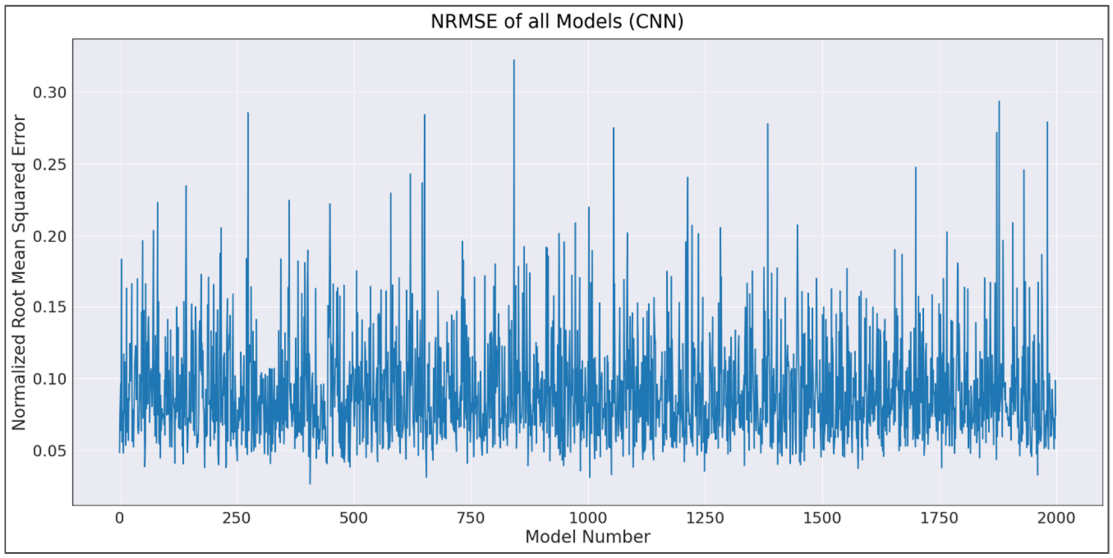

Figure 4.30 - Normalized Root Mean Squared Errors of all Simulation Models

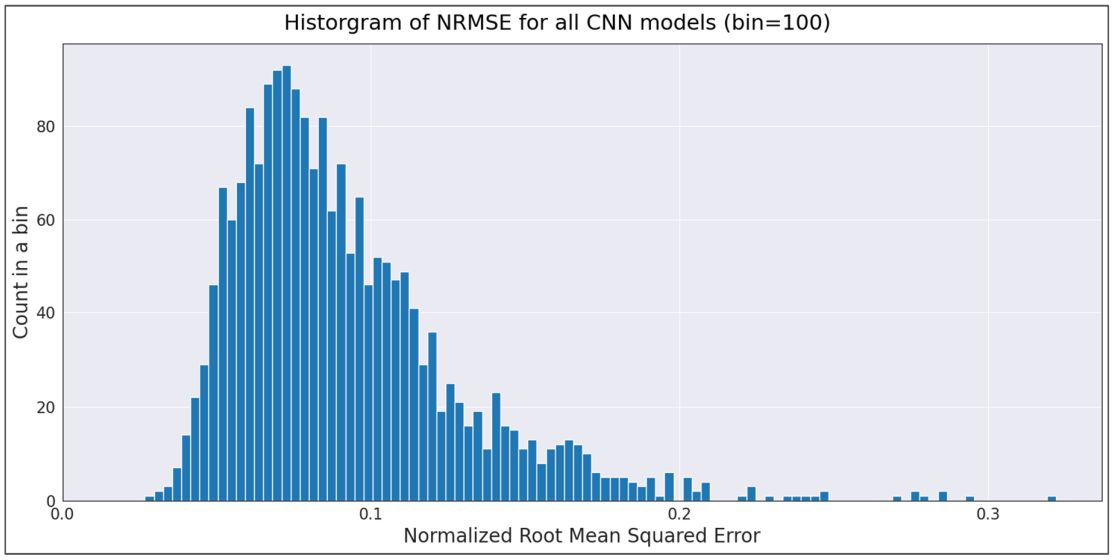

Figure 4.31 - Histogram of NRMSE for all CNN simulation models

The NRMSE of all the CNN simulation models is illustrated in Figure 4.30. Overall performance of the simulation models, which combine 1,000 cell models for each run, is homogeneous throughout the dataset. The histogram of the NRMSE for all simulation models can be seen in Figure 4.31. Model 420 is still in the top 50 among good performers when ranking the simulation models according to their performances.

Figure 4.32 gives information about the success of simulation model 420. Ten different bands of permeability values can be seen on the graph, and the model predictions are close

to the expected Comparing the expected vs. predicted permeability values of the same cell for 2,000 runs, CNN shows its reliability by predicting close to the expected values, thus staying close to thvalues. Being 14th among the worst performers, model 142 is illustrated in Figure 4.33. In this graph, there are ten bands of permeability values with expected permeability values from different ranges; still CNN model performed acceptably, especially for the low permeability predictions, which can be seen on the bottom left corner of the figure, and model 142 could still capture the general structure of the field.

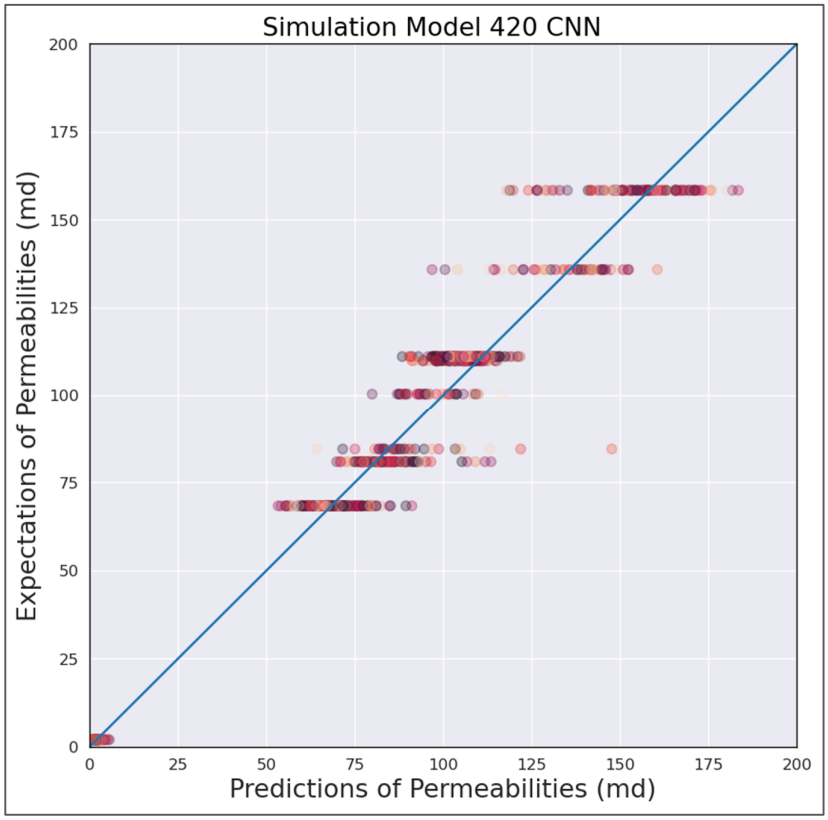

Figure 4.32 - Expected Permeabilities vs. Predicted Permeabilities - CNN Simulation Model 420

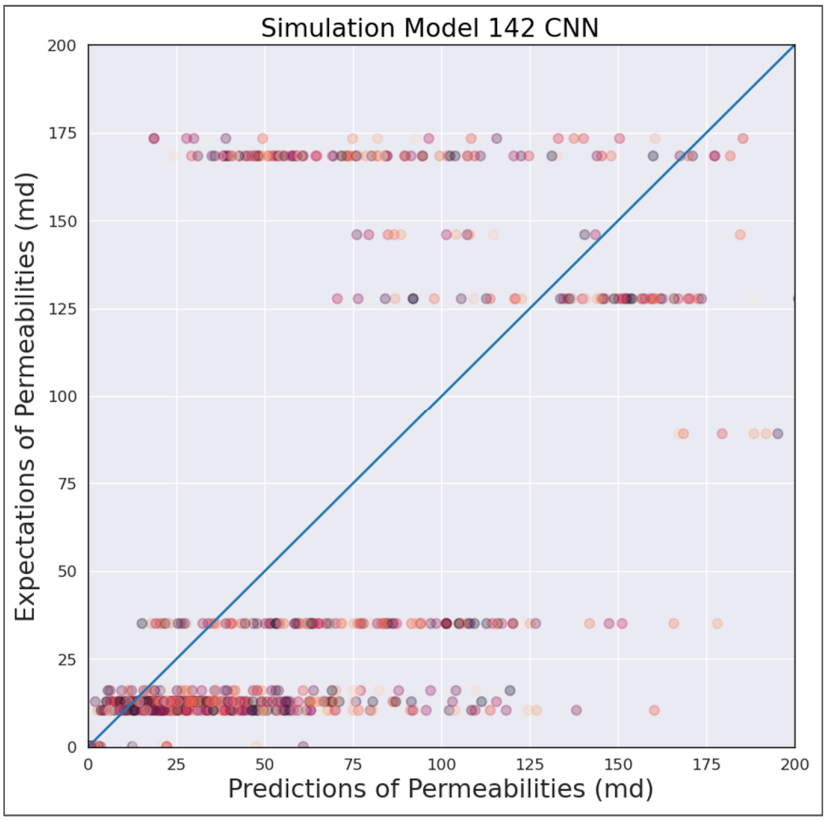

Figure 4.33 - Expected Permeabilities vs. Predicted Permeabilities - MLP Simulation Model 142

Figure 4.34 and Figure 4.35 illustrate the predicted and expected permeability distributions of different layers from model 420. In these figures, the CNN model captures the general structure of the field's permeability distribution. Even on the constant permeability zones, CNN performed better and can be used as a powerful tool when determining the permeability distributions of geothermal fields on natural state modeling.

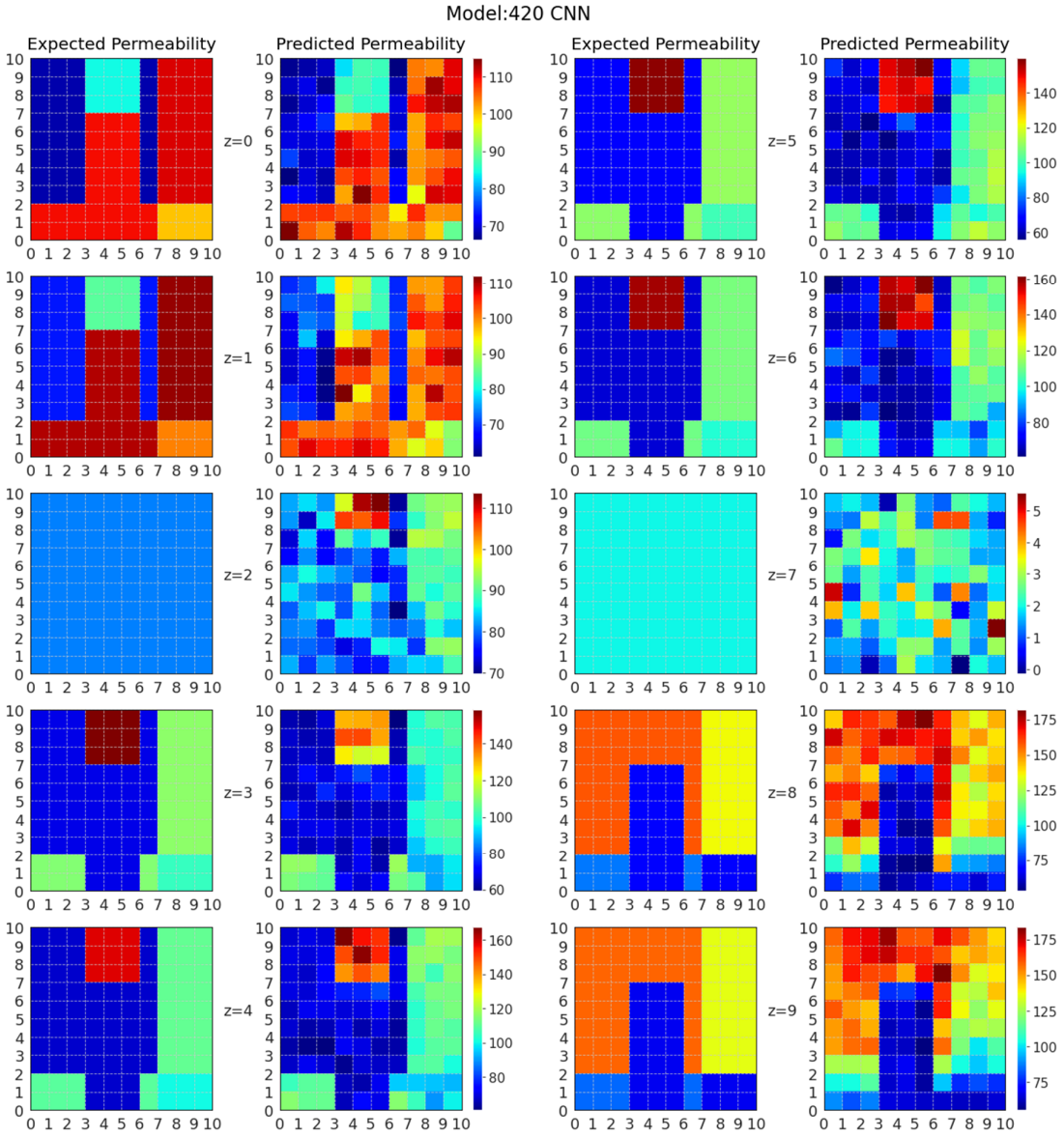

Figure 4.34 - Expected permeabilities and the Predicted permeabilities from MLP model 420

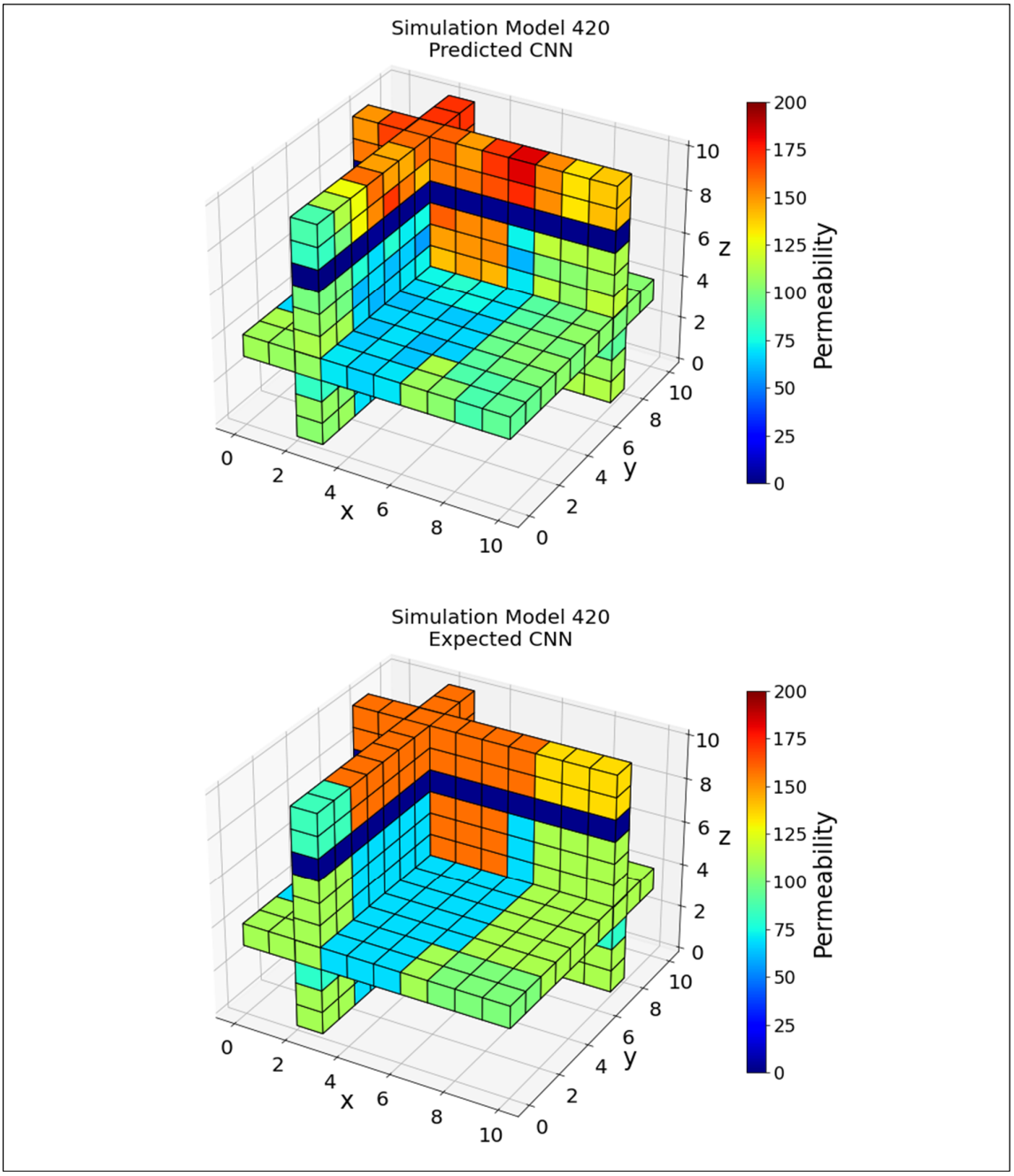

Figure 4.35 - 3D view of Predicted and Expected Permeability values for Simulation Model 420

4.6. Transfer Learning and Results

Transfer learning assumes that a model already trained on a similar problem could be used in another similar problem. In this part of the study, pre-trained models are used to test their viability in permeability determination in geothermal modeling. Typically transfer learning is used on image classification tasks, and since we already have our CNN models, it is only natural to wonder if transfer learning helps them or not.

The deep neural network API, Keras, already has built-in applications to call pre-trained models. Using this API, firstly, VGG-16 is selected for testing [31]. VGG-16 is a model which is trained on the ImageNet dataset with over 14 million images.

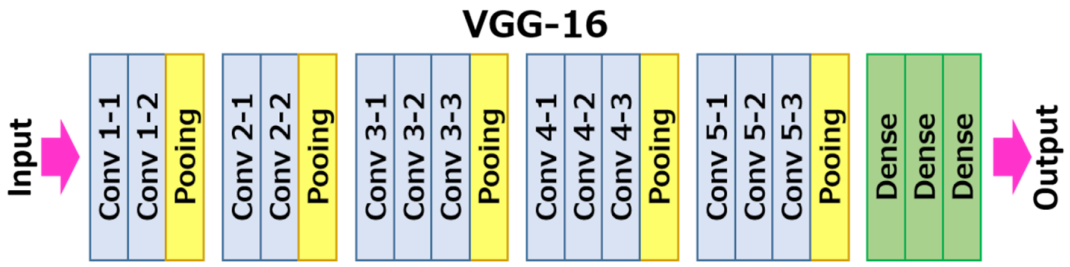

Figure 4.36 - VGG-16 architecture [32]

VGG-16 expects color images of dimensions 244x244 as data. An instance of our data's dimensions are 100x30 and have only one color channel. Also, VGG-16's minimum accepted image dimensions are 32x32x3. Since our dimensions are not enough for the minimum criteria of the VGG-16, we have doubled the size of the images, thus changing the dimensions of an instance to 200x60. After increasing the size, cell 555 is chosen as a cell for transfer learning to be tested. Before testing transfer learning, cell 555 is trained on both 100x30 and 200x30 data to see if changing the size of the images affects the training. Since the size difference did not cause any significant change in the model's success, newly created data is fed into VGG-16 for training.

In the first experiment, the VGG-16 is loaded without the input layer, and the classification output layer is replaced with a fully connected dense layer with a linear activation function. Next, all convolution layers are frozen so that the weights would stay the same throughout the training period. These steps are repeated in also in other experiments. After that, the fully connected part (300/200/100, ReLU) of our CNN structure is attached to the model. This resulted in 15,716,989 total parameters, 1,002,301 trainable

parameters and 14,714,688 Non-trainable parameters. The NRMSE of the training of cell 555 is 0.281, while the CNN model of the cell gave 0.086.

In the second experiment, the VGG-16 model's first 11 layers are frozen. With additional (300/200/100, ReLU) layers, this resulted in 15,716,989 total parameters, 13,981,501 trainable parameters and 1,735,488 Non-trainable parameters. NRMSE of the cell 555 from this model is calculated as 0.112 compared to the CNN result of 0.086.

In the third experiment, the VGG-16 model's whole connection is trained with an additional (300/200/100, ReLU). This resulted in 15,716,989 total parameters, 15,716,989 trainable parameters and 0 Non-trainable parameters. NRMSE of the cell 555 from this model is calculated as 0.114 compared to the CNN result of 0.086.

Similar experiments were conducted with VGG-19, DenseNet, ResNet50, and MobileNet, but the best result is still VGG-16 with partially frozen layers, which is worse than our CNN model. The cause of this low performance might be the difference in training mediums. VGG-16 and other pre-trained models are trained on photographs of real-life objects and might be very different from our use case. The other reason might be that the pre-trained models' expected dimensions were squares, and our input images might not be using the full potential of the pre-trained models.

4.7. Results And Discussions

Normalized Root Mean Squared Errors of the machine learning models used in this study can be seen in Table 4.4.

Table 4.4 - NRMSE of models, the best score is given in bold

Model	NRMSE
RF	0.134
SVR	0.153
MLP	0.123
CNN	**0.098**

In Figure 4.37, the NRMSE of all cells of all models can be seen. CNN achieves the best results in general, with relatively stable success throughout the model. MLP performs in a similar trend to CNN, but its general performance is worse than CNN. RF performs

similar to MLP with limited success in deeper model sections and good performance near boundaries. SVR performs similar to the RF at lower cells close to the heat source, but getting closer to upper cells, SVR becomes a poor model.

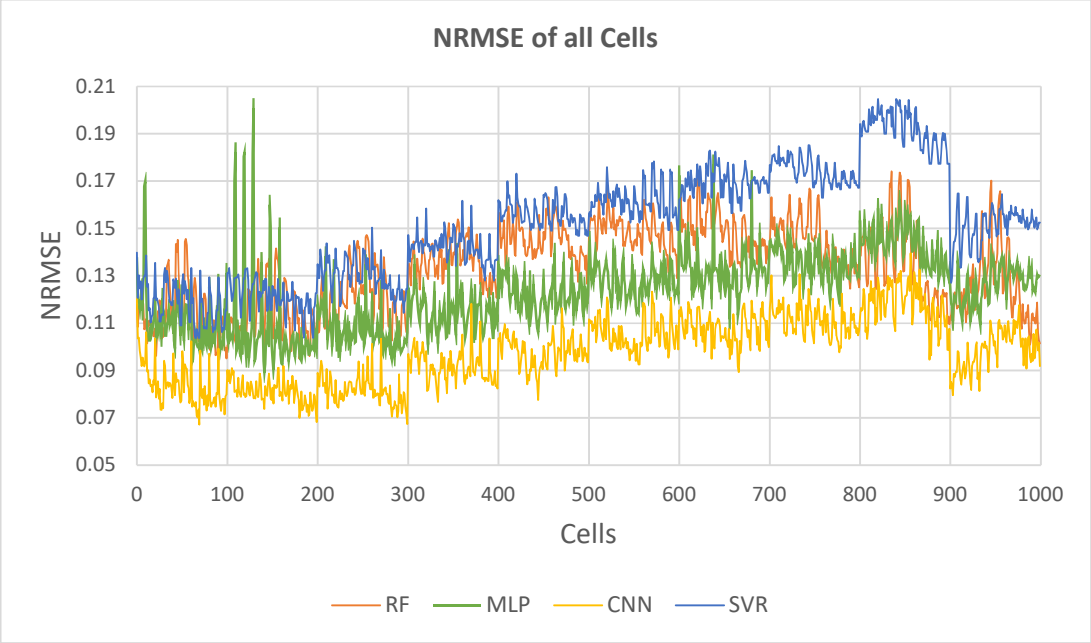

Figure 4.37 - NRMSE of all cells. All models

Comparing the expected vs. predicted permeability values of the same cell for 2,000 runs, CNN shows its reliability by predicting close to the expected values, thus staying close to the 45-degree line (Figure 4.38).

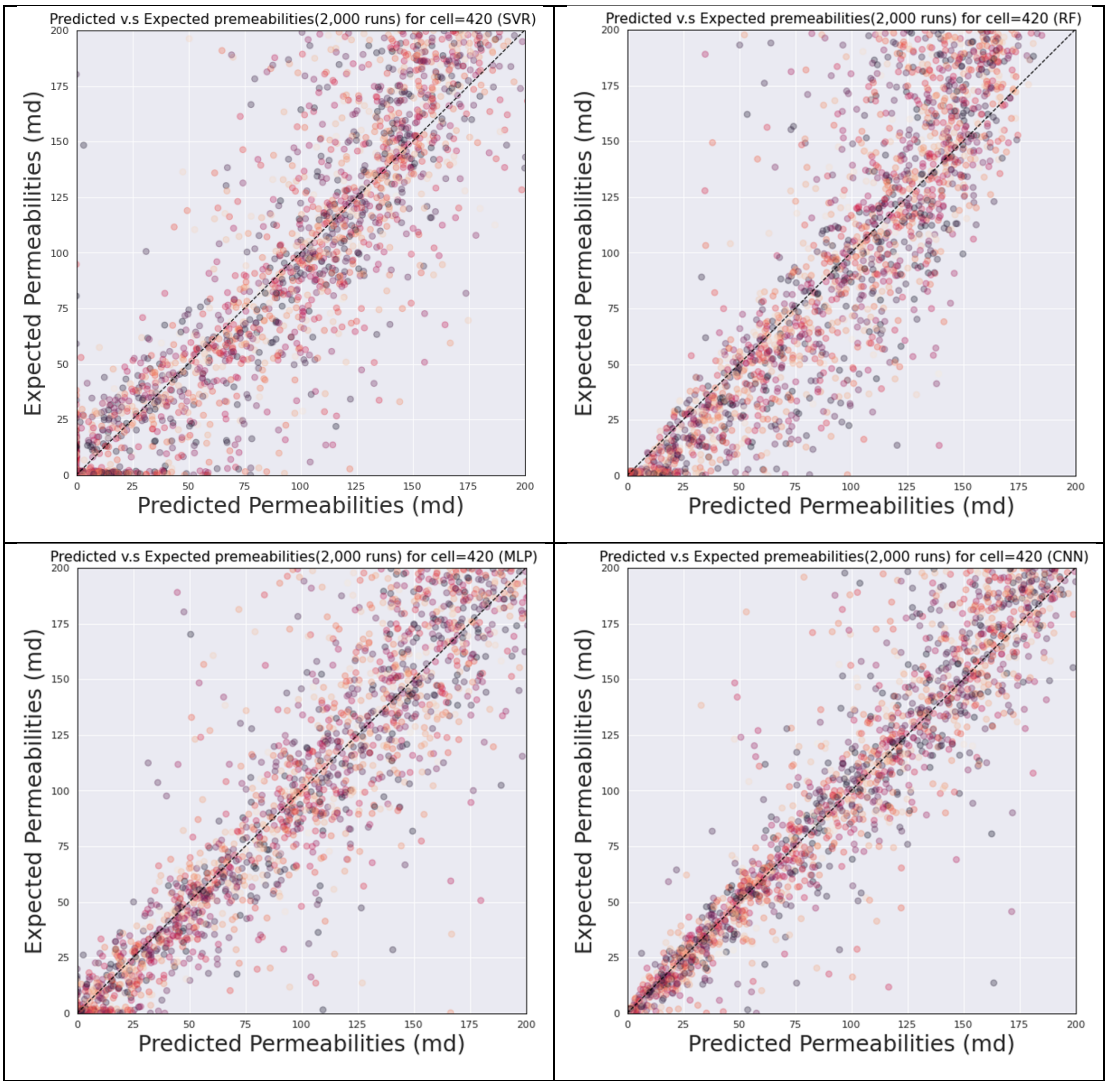

Figure 4.38 - Expected vs. Predicted permeability values of cell 420 for all models, (1md =~1.0E-15 m^2)

After inspecting the study results, the best machine learning algorithm of the proposed method is CNN. In determining the permeability values from temperature and pressure data for geothermal fields, CNN performs satisfactorily and can be used as a valuable tool that saves money and time.

5. CONCLUSION

In this study, we have conducted several experiments using five different machine learning algorithms, RF, SVR, MLP, CNN, and Transfer Learning to find a method to determine the permeability distributions of geothermal fields from pressure and temperature data in natural state modeling. We present the results using the normalized root mean squared errors and visual representations of the predictions versus expected values.

Being the first comprehensive investigation of fundamental machine learning methods to determine the permeability values in geothermal fields makes this study an important addition to the literature. Using our approach, we have created a method to roughly mimic nature while generating the base models. Our proposed method creates more detailed models with cap rocks, reservoirs, and different permeability regions resembling depositions and folds. Furthermore, using our method, experts can still influence the model with their field knowledge.

The method proposed in this study can also be used as an initial estimator for other inversion tools. Users of PEST or iTOUGH2 can start the optimizations with the permeability distributions created by our method and increase their success.

Our proposed method is an essential addition to the field by introducing new opportunities for cost savings and giving researchers new tools to model the geothermal fields better and use this renewable energy to its full potential.

For future work, the ensemble of different machine learning methods will be investigated. Combining the results of the studied methods as well as boosting methods and teaching a hybrid model is also another topic we plan to study. We will also investigate the transfer learning more deeply by implementing different pre-trained models with different input dimensions, including our CNN model, and improving our results.

REFERENCES

[1] A. Karpatne, I. Ebert-Uphoff, S. Ravela, H. A. Babaie, and V. Kumar, "Machine Learning for the Geosciences: Challenges and Opportunities," *IEEE Trans. Knowl. Data Eng.*, vol. 31, no. 8, pp. 1544–1554, Aug. 2019.

[2] S. Küçük, "Simulation of Geothermal Reservoirs With High Amount of Carbon Dioxide," METU, 2018.

[3] T. Cui, C. Fox, and M. J. O'sullivan, "Bayesian calibration of a large-scale geothermal reservoir model by a new adaptive delayed acceptance Metropolis Hastings algorithm," vol. 47, p. 10521, 2011.

[4] E. K. Bjarkason, J. P. O'Sullivan, A. Yeh, and M. J. O'Sullivan, "Inverse modeling of the natural state of geothermal reservoirs using adjoint and direct methods," *Geothermics*, vol. 78, pp. 85–100, Mar. 2019.

[5] K. Finsterle, S., Pruess, "iTOUGH2: Solving TOUGH inverse problems - iTOUGH," in *TOUGH2 Workshop*, 1995, pp. 287–292.

[6] S.P. White, "INVERSE MODELLING OF THE KAWERAU GEOTHERMAL RESERVOIR," in *New Zealand Geothermal Workshop*, 1995, pp. 211–216.

[7] S. Finsterle, K. Pruess, D. P. Bullivant, and M. J. O'Sullivan, "Application of inverse modeling to geothermal reservoir simulation - iTOUGH," in *Twenty-Second Workshop on Geothermal Reservoir Engineering Stanford University, Stanford, California, January*, 1997, pp. 309–316.

[8] D. P. Bullivant and O'Sullivan, M.J., "Inverse modelling of the Wairakei geothermal field | Signed in," in *TOUGH Workshop*, 1998, pp. 391–402.

[9] H. Moon, J. Clearwater, P. Franz, I. Wallis, and L. Azwar, "Sensitivity Analysis, Parameter Estimation and Uncertainty Propagation in a Numerical Model of the Ngatamariki Geothermal Field, New Zealand," *PROCEEDINGS, Thirty-Ninth Work. Geotherm. Reserv. Eng.*

[10] S. Jalilinasrabady, R. Itoi, H. Gotoh, and T. Tanaka, "DEVELOPMENT OF THE OPTIMUM NUMERICAL RESERVOIR MODEL OF THE TAKIGAMI GEOTHERMAL FIELD, OITA, JAPAN," *PROCEEDINGS, Thirty-Sixth Work. Geotherm. Reserv. Eng.*

[11] G. Gunnarsson, A. Arnaldsson, and A. L. Oddsdóttir, "Model Simulations of the Hengill Area, Southwestern Iceland," *Transp. Porous Media 2010 901*, vol. 90, no. 1, pp. 3–22, Aug. 2010.

[12] J. Doherty, "PEST Model-Independent Parameter Estimation User Manual Part I: PEST, SENSAN and Global Optimisers," Watermark Numerical Computing, Brisbane, 2016.

[13] Michael O'Sullivan and Racquel Colina, "Calibration of a geothermal model using PEST," in *35th New Zealand Geothermal Workshop*, 2013.

[14] B. M. Feather, R. C. M. Malate, and S. Knight Merz, "NUMERICAL MODELING OF THE MITA GEOTHERMAL FIELD, CERRO BLANCO, GUATEMALA," *PROCEEDINGS, Thirty-Eighth Work. Geotherm. Reserv. Eng.*

[15] M. J. O'Sullivan and J. P. O'Sullivan, "Reservoir modeling and simulation for geothermal resource characterization and evaluation," *Geotherm. Power Gener. Dev. Innov.*, pp. 165–199, Jan. 2016.

[16] A. Suzuki *et al.*, "Machine Learning for Input Parameter Estimation in Geothermal Reservoir Modeling," in *Proceedings World Geothermal Congress 2020+1*, 2021.

[17] Aurélien Géron, *Hands-on machine learning with Scikit-Learn, Keras and TensorFlow: concepts, tools, and techniques to build intelligent systems*, 2th edition. O'Reilly Media, Inc., 2019.

[18] E. Alpaydin, *Introduction to Machine Learning (Adaptive Computation and Machine Learning series)*. The MIT Press, 2009.

[19] Jason Brownlee, *Master Machine Learning Algorithms:* Machine Learning Mastery, 2019.

[20] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *J. Mach. Learn. Res.*, vol. 12, no. 85, pp. 2825–2830, 2011.

[21] S. Saha, *A Comprehensive Guide to Convolutional Neural Networks — Towards Data Science*. 2018.

[22] I. Goodfellow, Y. Bengio, A. Courville, and F. Bach, *Deep Learning (Adaptive Computation and Machine Learning Series) : Goodfellow, Ian, Bengio, Yoshua,*

Courville, Aaron, Bach, Francis: Amazon.com.tr: Kitap. MIT Press, 2017.

[23] P. Marcelino, "Transfer learning from pre-trained models | by Pedro Marcelino | Towards Data Science." [Online]. Available: https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751. [Accessed: 27-Aug-2021].

[24] R. Karim, "Illustrated: 10 CNN Architectures | by Raimi Karim | Towards Data Science." [Online]. Available: https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d. [Accessed: 27-Aug-2021].

[25] *Petrasim User Manual*. Thunderhead Engineering, 2018.

[26] K. Pruess, C. M. Oldenburg, and G. J. Moridis, "TOUGH2 User's Guide Version 2," Nov. 1999.

[27] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning*, 14th ed., vol. 103. Springer-Verlag New York, 2013.

[28] M. Kuhn and K. Johnson, *Applied predictive modeling*. Springer, 2013.

[29] M. Abadi *et al.*, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems."

[30] F. Chollet and and others, "Keras," *GitHub*. GitHub, 2015.

[31] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, Sep. 2014.

[32] "VGG16 - Convolutional Network for Classification and Detection." [Online]. Available: https://neurohive.io/en/popular-networks/vgg16/. [Accessed: 27-Aug-2021].