June 2020

Data-Driven Modeling and Prediction for Reservoir Characterization and Simulation Using Seismic and Petrophysical Data Analyses

Xu Zhou
Louisiana State University and Agricultural and Mechanical College

DATA-DRIVEN MODELING AND PREDICTION FOR RESERVOIR CHARACTERIZATION AND SIMULATION USING SEISMIC AND PETROPHYSICAL DATA ANALYSES

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

The Craft & Hawkins Department of Petroleum Engineering

by
Xu Zhou
B.S., China University of Petroleum - Beijing, 2012
M.S., Tulane University, 2015
August 2020

Acknowledgements

I would like to thank my advisor Dr. Mayank Tyagi for his instructions and support in this dissertation. I sincerely appreciate the guidance and help from him in advising me to finish this study. He has been very encouraging, inspiring, and patient throughout my time during my Ph.D. study and research. This dissertation won't be possible without the guidance and instruction from him.

I also thank Dr. Ipsita Gupta, Dr. Juan Lorenzo, and Dr. Krishnaswamy Nandakumar for serving on my dissertation committee and for their suggestions and help on this study. I appreciate the valuable advice and support from all of them.

I would like to thank Dr. Bin Li for the suggestions from him on the statistical modeling part of this research. I also thank Dr. Jyotsna Sharma for the discussion and suggestions on the wavelet convolution neural network chapter of this dissertation.

I thank the advising and guidance from Dr. Arash Dahi Taleghani for the first two and half years of study and research. I appreciate the guidance from him in helping me learn and conduct research in geomechanics and unconventional resources.

I thank the Craft & Hawkins Department of Petroleum Engineering at Louisiana State University for offering me the admission to the doctoral program and providing teaching and research assistantship for my research and study in the department. I thank the guidance from Dr. Andrew Wojtanowicz and Dr. Seung Kam as the Graduate Advisors in the department. Their advising and suggestions helped my research and study during my Ph.D. journey. I also appreciate all the support and discussion from all other professors and staffs in the department.

I thank Dr. Chao Wang for providing assistantship during my Ph.D. study and research. The discussion with him and his group members, particularly Srikanth Bangaru, has provided valuable insights in this research.

I am grateful to all the support and discussion from fellow graduate students in the department. I would like to thank my group members and colleagues, especially Mark Behl, Derek Staal, Qishuai Yin, Sharareh Heidarian, Bin Wang, Jack Blears, Hope Asala, Denis Klimenko, Kian Sheikhrezaei, Livio Santos and many others.

I thank Computational Modeling Group (CMG) for providing CMG software licenses for this dissertation. I also thank Schlumberger for providing Petrel license for this dissertation. I appreciate Society of Petroleum Engineers (SPE), Society of Exploration Geophysicists (SEG), TGS Geophysical Service, Rocky Mountain Oilfield Testing Center (RMOTC) and the U.S. Department of Energy (DOE) for providing data for this research. This dissertation won't be possible without donation of these software licenses and datasets.

Lastly, I would like to express my gratitude to my parents for their support and love for me during all my entire life. I also thank my wife, Wenwen Mei, and my daughter, Grace Zhou, for their caring and support.

Table of Contents

List of Tables

Nomenclature

TP	true positive prediction
TN	true negative prediction
FP	false positive prediction
FN	false negative prediction
K	curvature seismic attribute
r	radius of circle that is tangent to a curve or a surface
$\phi(t)$	instantaneous phase seismic attribute
F(t)	seismic trace
E(t)	trace envelope
s(t)	analytic trace
MAE	mean absolute errors
MSE	mean squared errors
R^2	coefficient of determination
y(i)	output of a convolution layer
x(j)	input of a convolution layer
$\omega(j)$	weights of a convolution layer
α	fluid phase
o	oil phase
w	water phase
k	permeability
ρ	density
p	pressure

S	saturation
M	original matrix m × n prior to SVD
U	a m × m unitary matrix after SVD
S	a diagonal matrix containing singular values
V	a n × n unitary matrix after SVD
σ	singular values

Abstract

This study explores the application of data-driven modeling and prediction in reservoir characterization and simulation using seismic and petrophysical data analyses. Different aspects of the application of data-driven modeling methods are studied, which include rock facies classification, seismic attribute analyses, petrophysical properties prediction, seismic facies segmentation, and reservoir dimension reduction.

The application of using petrophysical well logs to predict rock facies is explored using different data analytics methods including decision tree, random forest, support vector machine and neural network. Different models are trained from a set of well logs and pre-interpreted rock facies data. Among the compared methods, the random forest method has the best performance in classifying rock facies in the dataset.

Seismic attribute values from a 3D seismic survey and petrophysical properties from well logs are collected to explore the relationships between seismic data and well logs. In this study, deep learning neural network models are created to establish the relationships. The results show that a deep learning neural network model with multi-hidden layers is capable to predict porosity values using extracted seismic attribute values. The utilization of a set of seismic attributes improves the model performance in predicting porosity values from seismic data.

This study also presents a novel deep learning approach to automatically identify salt bodies directly from seismic images. A wavelet convolutional neural network (Wavelet CNN) model, which combines wavelet transformation analyses with a traditional convolutional neural network (CNN), is developed and demonstrated to increase the accuracy in predicting salt boundaries from seismic images. The Wavelet CNN model outperforms the conventional image recognition techniques, providing higher accuracy, to identify salt bodies from seismic images.

Besides, this study evaluates the effect of singular value decomposition (SVD) in dimension reduction of permeability fields during reservoir modeling. Reservoir simulation results show that SVD is valid in the parameterization of the permeability field. The reconstructed permeability fields after SVD processing are good approximations of the original permeability values. This study also evaluates the application of SVD on upscaling for reservoir modeling. Different upscaling schemes are applied on the permeability field, and their performance are evaluated using reservoir simulation.

Chapter 1. Introduction

This research covers topics in the application of data-driven modeling in reservoir characterization and simulation. It explores the possibility of creating statistical models that co-relates various data types including seismic data, well logs, and rock facies. The effectiveness of using SVD for dimension reduction of permeability field is also evaluated.

1.1 Background

Development in data acquisition technologies in the oil & gas industry has created significant amount of data in different types, including seismic, well logs, petrophysical properties, rock facies, and seismic facies. It is of great significance to explore the relationships between different datatypes to understand whether it is possible to use one data type to predict another. If this is possible, it reduces the amount of human effort and capital expenditure in collecting various types of data during hydrocarbon exploration and production.

Among the various types of subsurface data, well log data is a commonly used data source that helps geologists and engineers understand subsurface properties. After collecting well log data from the well bore in the subsurface, rock facies are interpreted by geologists based on the well log responses. It typically takes large amount of time and human efforts to interpret the rock facies. Thus, it is of great significance to look for the statistical relationship between well log data and interpreted rock facies to see whether well log data can be used to predict rock facies using data-driven modeling methods.

Seismic data is also among one of the common data types in the industry. A three dimensional (3D) seismic survey provides large amount of data about the subsurface properties. It enables a 3D display of geologic features and promotes more comprehensive analyses on the subsurface data.

3D seismic data interpretation provides a 3D view of the formations and structures in the subsurface and enables deeper understanding about the reservoir geometry and structure.

Moreover, seismic attribute values can be extracted from a 3D seismic survey and can help provide a quantitively understanding about subsurface properties. Seismic attribute analysis has been proved to be a useful geophysical method that can be used to identify subsurface features such as faults and fracture zones (Khair, 2012). These attributes can be automatically calculated by computer prior to human interpretation. This saves lot of time and work. It also helps characterize reservoir properties and identify structures in deep formations where seismic resolution is low. With larger amount of available 3D seismic data, it is worthwhile to explore the relationship between these seismic attributes and petrophysical properties from well logs. Data-driven models can be created to quantify these relationships.

After a seismic survey is conducted, the collected and processed seismic data can be viewed as seismic images. These seismic images can be used by geologist to interpret geologic features in the subsurface, including formations, structures, seismic facies, etc. Salt body is one of the common subsurface features that can be identified on seismic images. It is important to accurately determine the location and size of salt bodies to better understand the subsurface properties by interpretation of seismic images. For a 3D seismic survey conducted on a large study area, this processes typically take a lot of human efforts and hours to manually interpret seismic facies. Thus, it is of great significance to explore the usage of data-driven modeling in automatically predicting these seismic facies directly from seismic images. The predicted results will be a good reference for geologists and engineers to better understand subsurface properties and can reduce human efforts and costs.

During reservoir modeling, dimension reduction is important to reduce the size of reservoir models in order to save computational time for reservoir simulation. With reduced dimension, more realizations can be simulated for models with different parameter values to better characterize uncertainty for reservoir simulation.

Upscaling is an important step for reservoir modeling. Its goal is to substitute a heterogeneous model that consists of high-resolution fine grid cells with a lower resolution reduced-dimensional homogeneous model using averaging schemes. The benefit of upscaling in reservoir simulation is that it efficiently saves simulation time, and effectively preserves key features of data for flow simulations. The critical issue in upscaling is the selection of an appropriate scheme to effectively represent properties for the fine grid cells.

Singular Vector Decomposition (SVD) method is a matrix decomposition method. It has been used for different applications including image processing and facial recognition. It is worthwhile to evaluate the application of SVD in dimension reduction of the permeability field for reservoir modeling.

1.2 Motivation and Objective

With the development of computational power and data acquisition techniques, data-driven modeling has been used in many different fields to create automatic workflows that can make predictions to assist decision making processes. Utilization of these data-driven models has significantly increased the efficiency in performing various tasks in different applications (Porumb et al., 2020; Bangaru et al., 2020). One big area of its application is computer vision, which is to enable computers to be able to interpret the visual world using data-driven models trained from large amount of existing data.

One specific example for the application of data-driven modeling methods that is very common in our daily life is image recognition. In the example shown in figure 1.1, the input is a pixel image of a camera in a self-driving car. The program, or the trained model, can automatically identify the major features in the image, which include vehicles, streets, sidewalks, pedestrians, etc. Predictions are made at all pixels in the original image. These predicted results help the program make decisions to drive the vehicle with less human control. The model is not yet perfect, but there has been significant amount of research and projects being conducted in this area to improve the effectiveness and efficiency of the model.

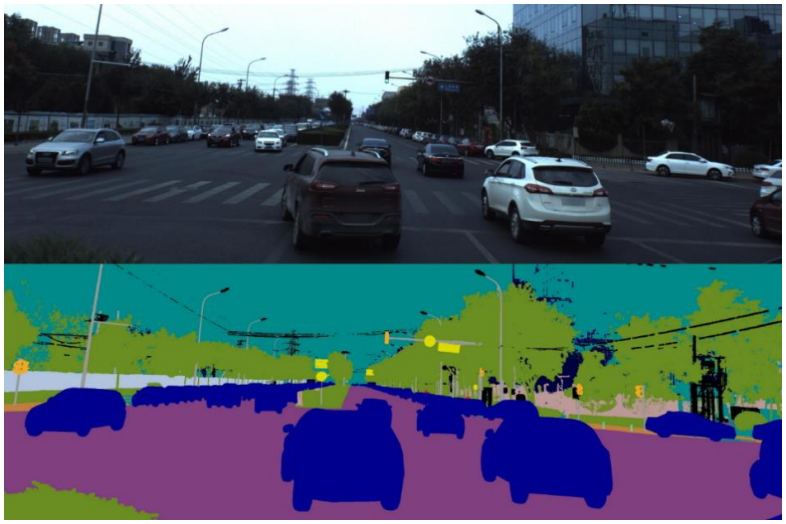

Figure 1.1. An example of using data-driving modeling for developing self-driving vehicles (CVPR workshop on autonomous driving, 2018).

Another example is in health care. Figure 1.2 shows an example that uses the signal of a single heartbeat to detect hear failure. It is an example of a classification problem. The input is signal from individual heartbeats. The output is the classification result predicting the input signal to be a normal heartbeat or a congested heart failure.

After training from heartbeat signals and pre-determined labels, the model is capable to automatically predict heartbeat signals from new patients to be a normal heartbeat or a congested

heart failure. The model has shown a nearly 100% percent accuracy, which provides useful information to assist human interpretation processes.

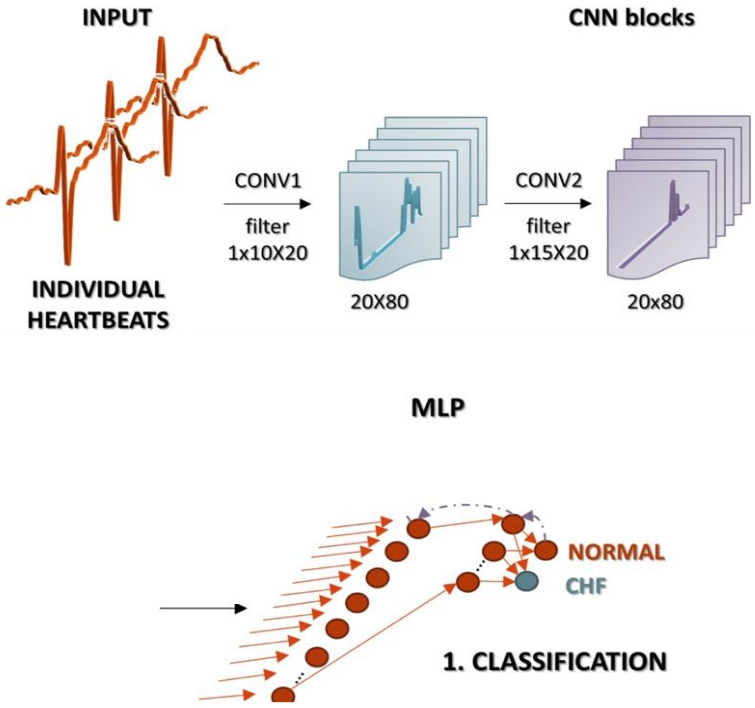

Figure 1.2. An example of using data-driving modeling for heartbeat signal analyses (Porumb et al., 2020).

With the wide use of data-driven modeling methods in various fields nowadays, it is worthwhile to explore the use of data-driving modeling in the oil and gas industry. Thanks to the various types and large numbers of available data in the industry, together with the recent developments of data analytics algorithms, there exists a great protentional to use data-driven modeling and prediction to explore the inter-relationships between different data types in the industry.

Well logs and rock facies data can be collected to explore their relationships. Data-driven models can be created between these two data types, and well logs can be used to predict rock

facies using the established data-driven models. This will reduce the amount of human work in interpreting rock facies from well logs.

The relationship between seismic attributes and petrophysical properties can be studied. Since seismic data is collected at the reservoir scale and is available at every location of the seismic survey, this relationship can be used to estimate the petrophysical and reservoir properties at all locations in the reservoir. It can also be utilized to predict petrophysical properties in other newly less explored areas.

During oil and gas exploration operations, after running and collecting seismic surveys, seismic attribute values can be extracted, and it can used to estimate petrophysical properties using the established relationships between seismic attribute values and petrophysical properties. In this case, only a smaller number of wells need to be logged to verify the estimated petrophysical properties. If the predicted values match the well log data, then the model can be used to estimate petrophysical properties in other locations within the reservoir. This reduces the cost of performing additional well logging operations in the field.

Seismic images, together with interpreted seismic facies can be used to train a deep learning convolutional neural network model to explore their relationships. The model can be used to help predict seismic facies automatically from seismic images. Subsurface features such as salt bodies can be identified from the trained model. This can save a lot of human efforts from the manual interpretation of seismic images.

In order to save reservoir simulation time, reservoir dimension reduction is a necessary step in creating a coarse grid reservoir model for reservoir simulation. Since SVD has been proved to be effective in imaging compression, the effect of SVD on dimension reduction of reservoir petrophysical properties can be evaluated for reservoir modeling. Different cases with different

SVD schemes can be created in order to evaluate its effectiveness in dimension reduction. By comparing the simulation results of different cases, the effectiveness of SVD in dimension reduction can be evaluated.

This research creates a comprehensive workflow for reservoir characterization and reservoir simulation using data-driven modeling methods. It explores the possibility to establish the relationships between different subsurface data types. Well log data is used to predict rock facies. Petrophysical properties in the reservoir can be estimated by exploring the relationship between seismic attribute values and these petrophysical properties. Seismic images can be used to automatically identify seismic facies through trained data-driven models. Besides, SVD is performed on the reservoir permeability field to evaluate its effectiveness in dimension reduction of reservoir parameters.

1.3 Workflow

Figure 1.3 shows the workflow and the research methodology for each step. The whole process includes two parts; the upper part represents the process of exploration the relationships between different data types including well logs, seismic attributes and facies. The lower part represents the process to evaluate the performance of SVD on dimension reduction of the permeability field for reservoir modeling.

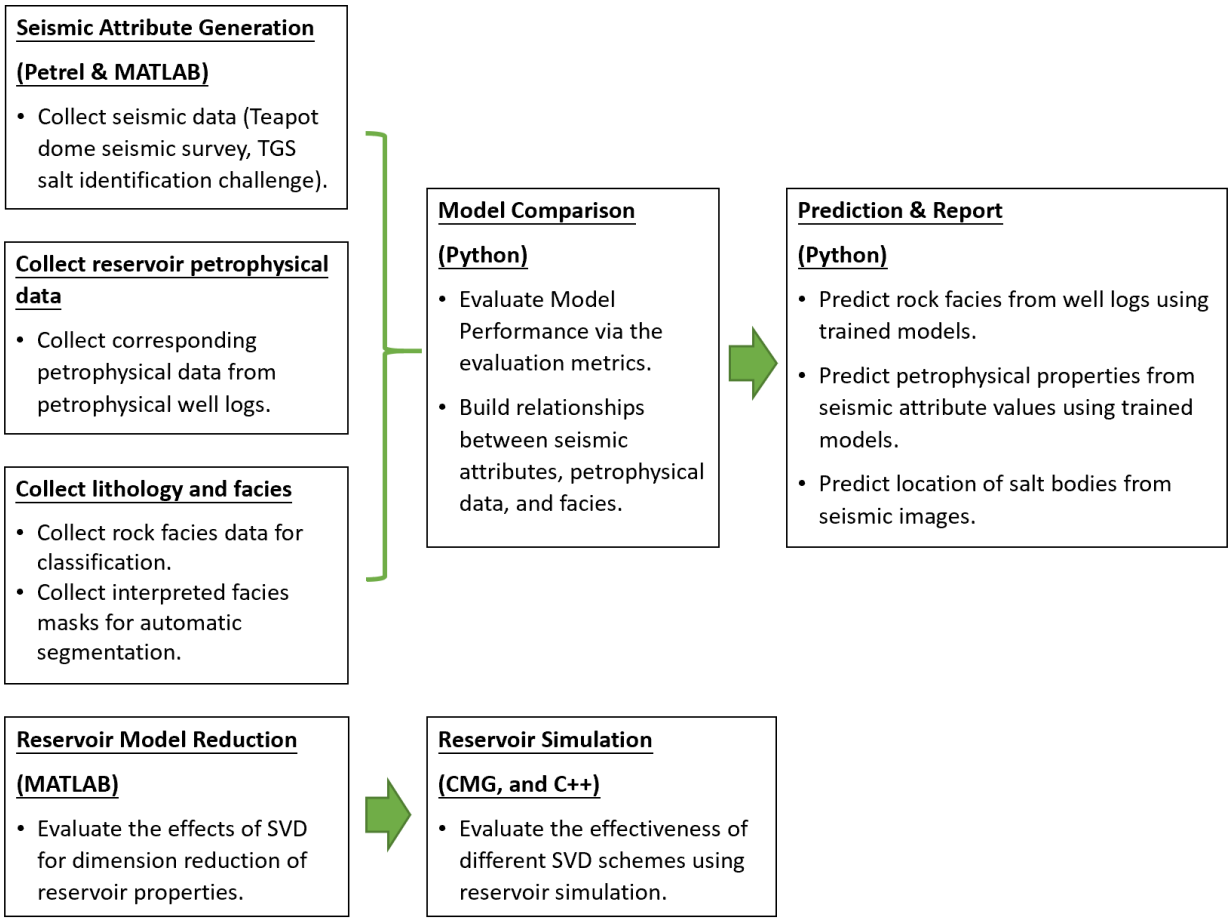

Figure 1.3. Workflow and research methods for each step. The upper part represents the process of exploring the relationships between different data types. The lower part represents the process to evaluate the use of SVD on dimension reduction.

In the first part, various datatypes are collected in order to explore the relationships using data-driven modes. Well logs and rock facies data are collected and imported into data-driven models to look for the relationship between these two data types. Different data analytics models, including decision tree, random forest, support vector machine and artificial neural network, are created using Python to evaluate each model's performance. Evaluation metrics, which include accuracy, precision, recall, and F-1 score, are calculated for each method to compare their prediction performance.

Seismic attributes and petrophysical data are also compiled and imported into data analytics models to explore the possibility of using seismic attributes to predict petrophysical properties. The seismic and well log data is collected from the Teapot dome 3D seismic survey. Petrel is used as the reservoir modeling tool to extract the seismic attribute values, and to co-relate the seismic attribute values with the well log data. Deep learning neural network models are created in Python using the TensorFlow library.

Besides, seismic images and interpreted salt images are imported into CNN models to explore the possibility of creating an automatic segmentation tool that can help identify salt bodies from seismic images. The application of a novel Wavelet CNN model is introduced and evaluated for its performance in identifying salt bodies. 2D seismic images of the subsurface and pre-interpreted salt bodies are collected from TGS Geophysical Survey. The trained model takes seismic images as the input and can automatically create an output image with predictions of salt bodies. The output predicted image has the same dimension with the input seismic image. Python and TensorFlow library are used as the tools to create the Wavelet CNN model.

In the second part, SVD is performed on the reservoir permeability field to evaluate the effects of SVD on reservoir dimension reduction. MATLAB is used as the tool to perform the SVD processing on the permeability field, as well as the reconstruction process. When performing SVD on a parameter, the parameter matrix can be decomposed and represented by the product of three separate matrices, U, S, and V. Different numbers of singular values can be used in S during the reconstruction of the original parameter matrix.

After the reconstruction of the parameter matrix, reservoir simulation is performed for both the original and the reconstructed parameter values to evaluate whether SVD processing is valid for reservoir dimension reduction. A two-phase flow reservoir model was created in C++ and CMG

for reservoir simulation. Reservoir models with different SVD processed permeability fields are simulated in order to evaluate the effectiveness of SVD processing in reservoir dimension reduction.

Chapter 2. Data-Driven Classification of Rock Facies Using Petrophysical Well Logs

This chapter explores the application of using petrophysical well logs and different data analytics methods to automatically classify rock facies. Well logging is a commonly used tool to understand subsurface properties. However, the interpretation of well logging data may take a significant amount of time. Thus, it is worthwhile to explore the statistical relationship between well logging responses and the interpreted rock facies. This helps create a workflow to automatically interpret rock facies from well logging responses.

2.1 Introduction

During hydrocarbon exploration and production, it is of great significance to understand the rock facies in the subsurface to target the potential hydrocarbon zones. Well logging is a very powerful tool to characterize rock facies and properties in the subsurface. There are many different types of logging methods. Some are passive methods, and others are active methods. Giving the fact that a hydrocarbon well is usually thousands of feet, there is huge amount of data generated during a single well logging operation. Thus, it typically takes a lot of human time and effort to manually interpret and validate all the well logs. Moreover, an oil field usually contains dozens or hundreds of wells during exploration and production. So, there is tremendously amount of well log data that needs to be interpreted. Thus, it is meaningful to use data-driven modeling methods to find the statistical relationship between the well logging responses and the different rock facies. This saves time and human effort for geologists and petroleum engineers.

In order to explore the relationship between well logging responses and the rock facies, a set of well logs and corresponding rock facies data are collected. Correlation plots are created to visualize the relationship between each well logging responses. Various data-driven models including decision tree, random forest (RF), support vector machine (SVM) and artificial neural

network (ANN) models are trained and tested to establish the relationship between well logging responses and rock facies.

The whole process contains three steps. In the first step, data that contains well logs and pre-interpreted rock facies is collected and pre-processed. Various types of well log data can be collected to increase the dimension of dataset. For each sample data point in the dataset, the location of the pre-interpreted rock facies needs to match the location of well log values. Besides, dataset that has missing values needs to be pre-processed. The data points with missing values may be removed, or their values can be estimated based on the values of other relevant data points or features.

In the second step, correlation plots are created to visualize the relationships between each well logging responses. Correlation matrix can be created to quantitatively describe the interrelationships between each input feature.

The third step is to create the data-driven models to make predictions. Categorical models that uses different algorithms can be established for rock facies classification using well logs. Evaluation metrics, which include accuracy, precision, recall, and F-1 score, can be used to compare the model performance.

2.2 Literature Review

Facies classification is a fundamental step for reservoir modeling. Accurate understanding of facies plays a significant role in improving the characterization of reservoir properties during oil and gas exploration and production (Xiong et al., 2010; Liu et al., 2017; Liu et al., 2020). It helps geologists and engineers to have better understanding about the geological, petrophysical, and reservoir properties.

The vigorous development of data-driven modeling methods has attracted lots of attention in the oil and gas industry (Sebtosheikh and Salehi, 2015; Xie et al., 2018). Different methods are studied and utilized in various applications including well log interpretation, seismic signal analysis, and seismic interpretation (Zu et al., 2018; Chen et al., 2018; Qu et al., 2019, Lopez et al., 2020). The inter-relationship between various input data and the output data have been explored using different data analytics approaches (Cracknell and Reading, 2014).

Zhao and others (2014) used a Proximal Support Vector Machines (PSVM) model in order to separate limestone and shale in a Barnett Shale gas field. Li and Zhang (2016) explored the application of data-driven models to predict sand, shale, and sand/shale mix from well logs. Different data analytics algorithms including logistic regression, gaussian discriminant analysis, random forest, and support vector machine were tested and compared in order to find the model that has the best predicting result.

Liu and others (2020) developed a Multikernal Relevance Vector Machine to improve the accuracy of lithology identification using a set of inverted elastic attributes. Their method preserves advantages of the regular Support Vector Machine algorithms and implements the optimizing processes with Bayesian analyses. Their results show advantageous properties including better generalization and accuracy in identification of rock facies comparing to traditional methods.

Kim et al. (2018) developed random forest models to assist seismic facies classification aided by stratigraphical interpretation and well logs. Their model also determines the importance of each input feature in classifying seismic facies. This helps select the important features and reduce the amount of computational power in establishing more complex models afterwards. Lopez et al. (2020) developed different data-driven models, including least-squares polynomial approximation,

random forest and support vector machine, to help differentiate lithology types including sand, silt and clay. Their results show that the data-driven models are valid to distinguish different lithology types using data from electrical resistivity and seismic wave velocity values. Results also show that the predicted lithology from the random forest model has a better statistically correlation with the actual lithology for their study.

2.3 Methodology

Thanks to the development of data-driven modeling methods, there are numerous data-driven theories and techniques that are available to be used by scientists and engineers in various fields. Based on the different objectives of different data-driven models, they can be categorized into two major types; supervised learning, and unsupervised learning.

Supervised learning is performed by feeding training data which has input variables and output variables. The main objective of a supervised learning model is to look for the relationship between the input variables and the output variables from the data (Kotsiantis 2007). These relationships can be linear or highly non-linear and are usually explored though the optimization of cost functions. One common cost function is the mean squared error, which is the average squared difference between the predicted results from the trained model and the true results. By establishing the relationship between the input variables and the output variables, new data can be input into the trained model to predict the values of the output variables.

Unsupervised learning is another type of data-driven algorithm that can be used to explore the relationship between input variables without the labeled output variables (Figueiredo and Jain, 2002). The commonly used unsupervised learning method includes clustering analysis, and principle component analyses (PCA). These methods aim to look for the inner patterns between the input variables and can be used for grouping samples in different categories.

In this study, different models are created using different data-driven algorithms, including decision tree, random forest (RF), support vector machine (SVM), and artificial neural network (ANN). These models are evaluated to select the one that performs the best in classifying rock facies.

2.3.1 Decisions Tree

A tree structure has many analogies in various fields in the real life. During a decision-making process, the decision tree model can be utilized to visually demonstrate the decision-making steps. The decision tree method is also commonly used in data-driven modeling in many different fields, including both classification and regression problems.

A decision tree model has an opposite direction to a real tree structure. Figure 2.1 displays the structure of a decision tree with its root on the top (Safavian and Landgrebe, 1991). Each circle, which represents an internal node, is based on where the tree structure splits into more branches. The end of the tree structure that doesn't split further is called the leaf or the decision. They represent the different labels or classes in data-driven models (Safavian and Landgrebe, 1991; Friedl and Brodley, 1997).

At each conditional node, the algorithm calculates the cost for each split using a cost function. The node that has the lowest cost is selected as the root node. After the root node is selected, the tree splits at the root and the algorithm looks for the nodes at lower levels. A maximum depth value, which represents the maximum distance from the top of the structure to the bottom, can be preset to determine when the splitting stops.

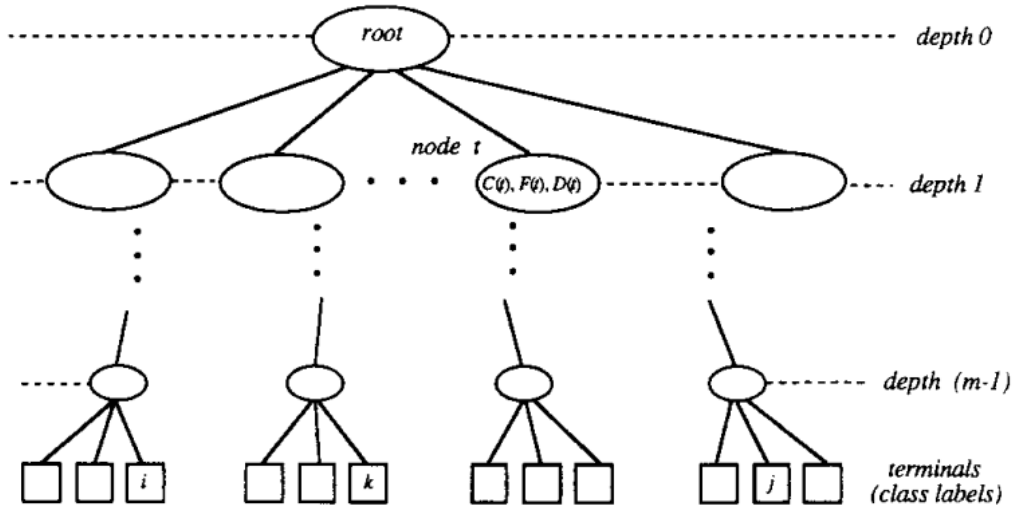

Figure 2.1. An illustration of the structure of a decision tree model (Safavian and Landgrebe, 1991).

2.3.2 Random Forest

Random forest, which is an ensemble data-driven modeling method, consists of large numbers of single decision trees (Liaw and Wiener, 2002; Pal, 2005). Each decision tree forms a sub-model and makes a prediction. The random forest model considers the prediction result from each single decision tree and determines the final output of the model by taking the prediction that has the maximum occurrence.

During training processes, a randomly selected group of data points are used to train each single decision tree. These groups of data points are selected with replacement, meaning that some data points may be used to train different decision trees multiple times. Besides, the selection of features in each single decision tree is also random. It means that a randomly selected sub-group of features is used to construct each single decision tree. By considering predictions from large amount of randomly created independent decision trees trained from randomly selected group of data points, the random forest model helps reduce bias of the model and increase the model performance.

Figure 2.2 illustrates the structure of a random forest model (Yiu, 2019). There are totally nine single decision trees in the illustration. These nine trees are formed with random combinations of input features. Each tree is trained with randomly selected number of data points. These single decision trees predict either 1 or 0. Among these nine trees, there are six trees predicting 1 and three trees predicting 0. Thus, the output of the random forest model is 1 by taking the majority voting from all the nine single trees.

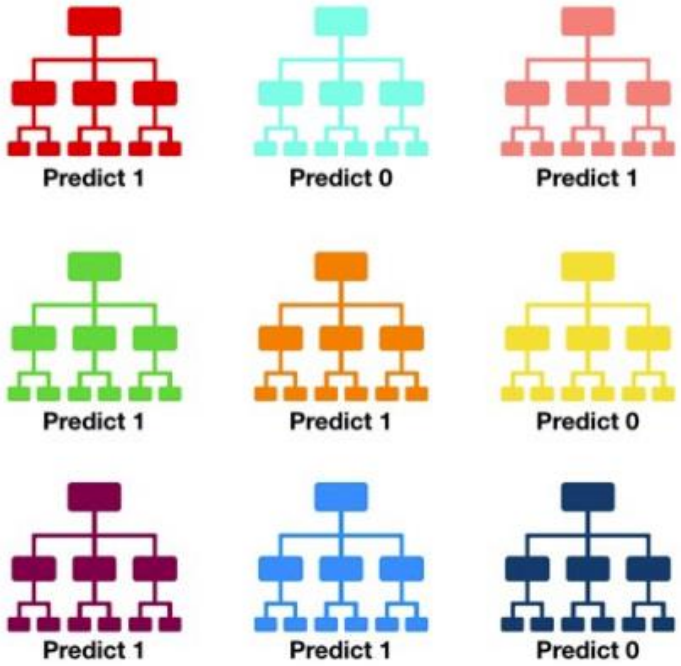

Figure 2.2. An example of the structure of a random forest model (from Yiu, 2019).

2.3.3 Support Vector Machine

Support vector machine (SVM) is a commonly used data analytics method. It is widely used for classification problems and can be used for regression tasks as well. The goal of the SVM method is to look for hyperplanes that can be used to separate all the samples in the N-dimensional space, in which N represents the dimension of input variables (Suykens and Vandewalle, 1999; Fung and Mangasarian, 2005).

As shown in figure 2.3, in order to distinguish the two different classes of samples, there exists many different hyperplanes that can be selected. The goal of the SVM algorithm is to select the best hyperplane that has the largest margin between two classes (as shown figure 2.4). This helps provide stronger reinforcement to the model so that future samples can be predicted by the model with higher confidence.

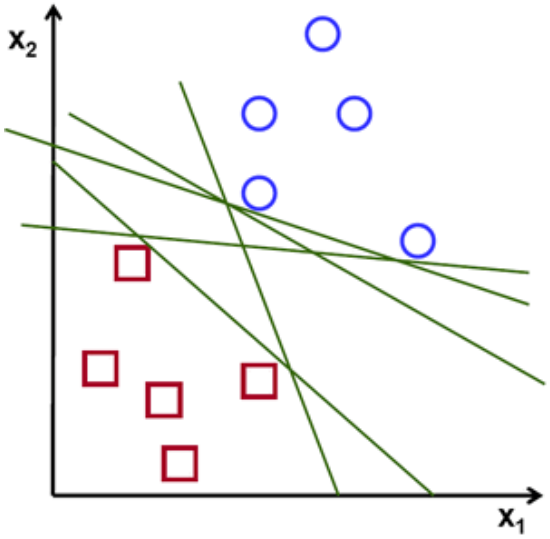

Figure 2.3. Illustration of the possible hyperplanes that could be chosen to separate two classes of data points (Gandhi, 2018).

Hyperplanes can be considered as boundaries that separates all input data samples. The data points that locate on two sides of the hyperplane can be classified into different groups. The dimension of the hyperplane that differentiate the input data depends on the dimension of the input features. As shown in figure 2.3 and 2.4, the dimension of the input data is two, and the optimal hyperplane is a line that separates the two classes. In other cases, if the dataset has the dimension of three, the hyperplane is a plane. If the input data has larger dimension more than three, which is the common situation, the hyperplane is difficult to visualize or imagine.

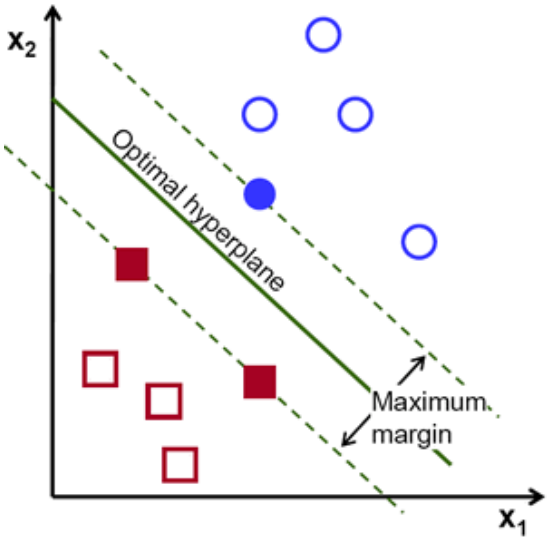

Figure 2.4. Illustration of the optimal hyperplane and support vectors (Gandhi, 2018).

Support vectors represent for sample points that locate close to the hyperplane, which help establish the model. The algorithm utilizes these support vectors to determine the location of the optimal hyperplane by maximizing the margin between the datapoints belonging to different classes. A loss function is used to help maximize the margin. It searches the optimal hyperplane that separates different classes.

2.3.4 Artificial Neural Network

Artificial neural networks are initially inspired by the biological neural network inside animal brains (Zupan and Gasteiger, 1993; Jain et al., 1996). These neural network systems can learn from collected data or examples to improve performance of certain tasks. Artificial neural networks have a broad range of applications for both classification and regression problems.

Neural network systems can be considered as simple math functions that defines the relationship between input variables (X) and output variables (Y). Giving the dataset that contains both X and Y, the neural network model can be trained to look for the relationship between X and Y. This relationship can be used to make predictions on new values of X.

More specifically, a neural network model includes several components, the input layer, the hidden layers, and the output layer. The input layer contains the input variables. The number of input layer consists of X_1, X_2, …, X_m, which represents the input variables and can pass values from the training dataset to the next hidden layer.

The hidden layers are layers of neurons that are between the input layer and the output layer. They can transfer information from the neurons in the previous layer to the neurons in the next layer. The number of the hidden layers in a model is changeable and can be optimized based on specific dataset for different tasks.

Based on the number of hidden layers in the structures of the model, neural network models can be categorized into two different types; a single layer neural network or a multi-layer neural network. The major difference between a single layer versus a multi-layer neural network model is that a multi-layer neural network has more than one hidden layer. A multilayer neural network is also called a deep learning neural network since it has more than one hidden layer, and can be used to model complex non-liner relationships.

Figure 2.5 and 2.6 shows the structure of a single hidden layer and a multi-layer neural network model. In both figure 2.5 and 2.6, each layer contains several neurons which receive input signals from previous layers and can be transmitted to next layers. This resembles the transmitting of neural signals in animal brains. The importance of the transmitting relationship can be descripted as the weights between the input variables and each neuron.

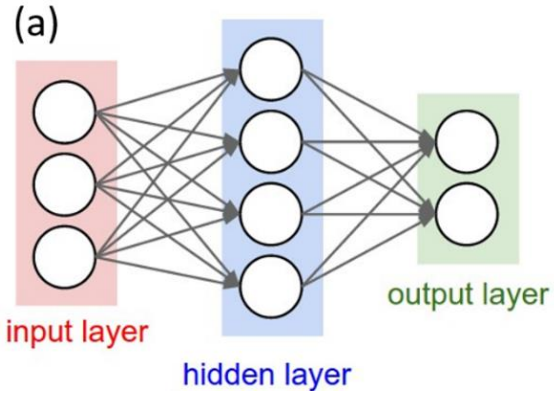

Figure 2.5. Structure of a single hidden layer artificial neural network model. Circles represent for input variables, neurons and output variables (Li, 2019).

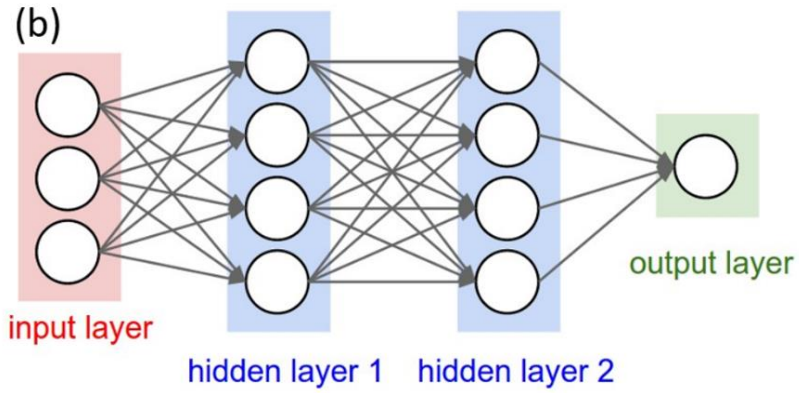

Figure 2.6. Structure of a multi-layer artificial neural network model (Li, 2019).

Figure 2.7 shows a single neuron unit inside the hidden layer. The neuron applies a non-linear function to the weighted sum of inputs to produce a final output. This non-linear function is called the activation function f. Commonly used activation functions includes ReLU, Sigmoid, Tanh, etc (Li, 2019).

The output layer of a neural network model contains the output neurons, which are included the last layer of a neural network model. They receive information from neurons in the previous hidden layer and perform computations to transfer data that passes from the previous layer to the final output variables (Y).

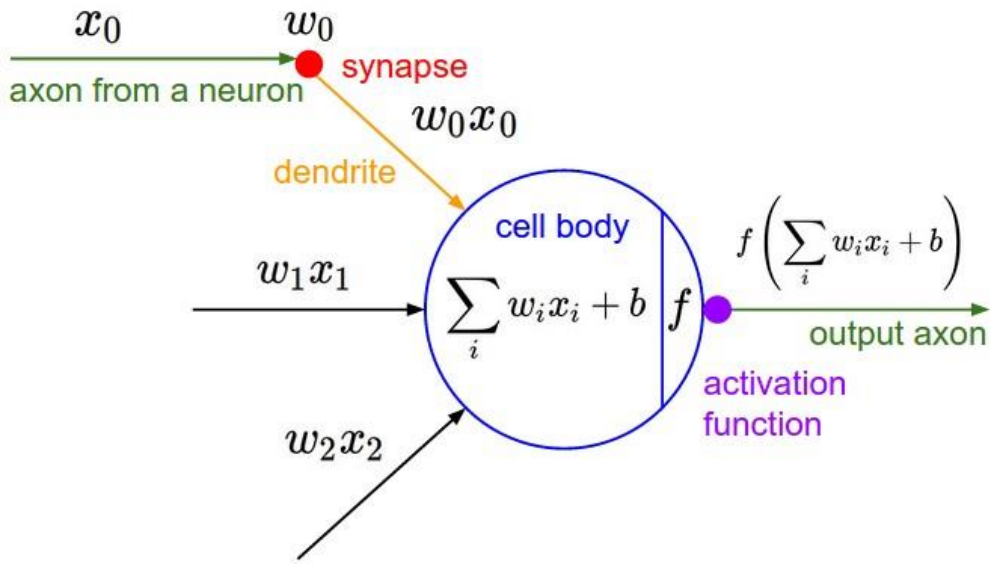

Figure 2.7. The structure of a single neuron. f represents the activation function. It applies a non-linear function to the weighted sum of inputs to produce an output (Li, 2019).

The training process of a neural network system is to look for the optimal values of the weights. These values from weights, together with the number of hidden layers and the number of neurons, define the trained neural network model. By calculation these values, the relationship between X and Y can be quantitatively established.

During the training process, there are many different activation functions that can be used based on the objects of each specific problem. A commonly used activation function is rectified linear unites (ReLU). Its function and derivative function are shown as follows. Figure 2.8 shows the plots for the function and its derivative. It is a non-linear function and provides good performance for neural network models. It is computationally efficient since it contains relatively simple math equations.

$$R(z) = \begin{cases} z, & \text{for } z > 0 \\ 0, & \text{for } z \leq 0 \end{cases}$$

$$R'(z) = \begin{cases} 1, & \text{for } z > 0 \\ 0, & \text{for } z < 0 \end{cases}$$

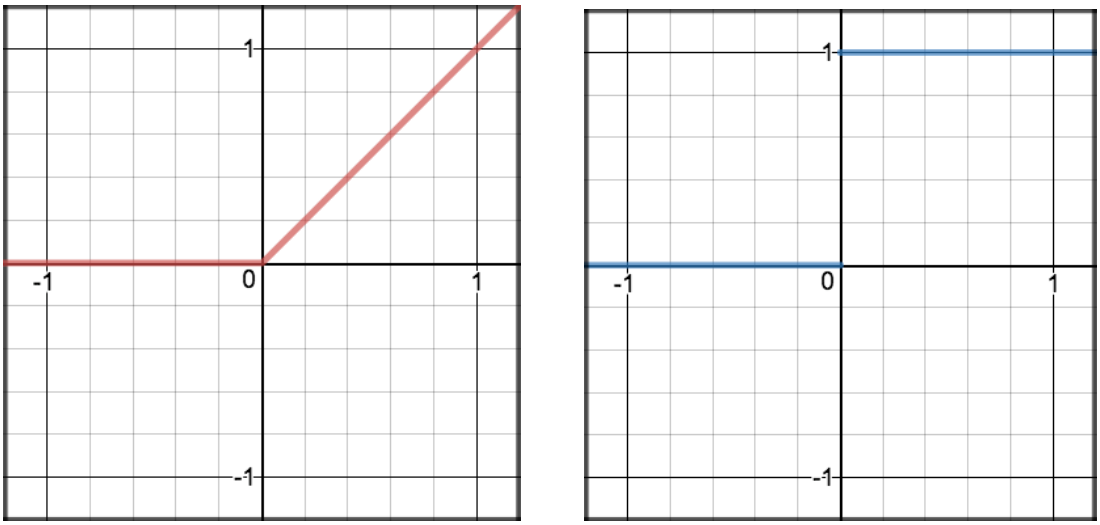

Figure 2.8. Illustration of the ReLU function and the derivative form (Li, 2019).

Sigmoid function is another commonly used activation function for neural network models. It is a non-linear function. The Sigmoid function and its derive function are shown as follows. Figure 2.9 shows their plots. Both the sigmoid function and its derivative are continuous. It has a fixed output range to be between 0 and 1.

$$S(z) = \frac{1}{1 + e^{-z}}$$

$$S'(z) = \frac{1}{1 + e^{-z}} \cdot (1 - \frac{1}{1 + e^{-z}})$$

Tanh is another commonly used activation function. The Tanh function and derivative function are shown as follows. Figure 2.10 shows their plots. It is a non-linear function and has a continuous derivative form. It also has a fixed bound like the Sigmoid function. However, the output range is zero-centered from -1 to 1, which is larger than the Sigmoid function.

$$\tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

$$\tanh'(z) = 1 - \tanh\,(z)^2$$

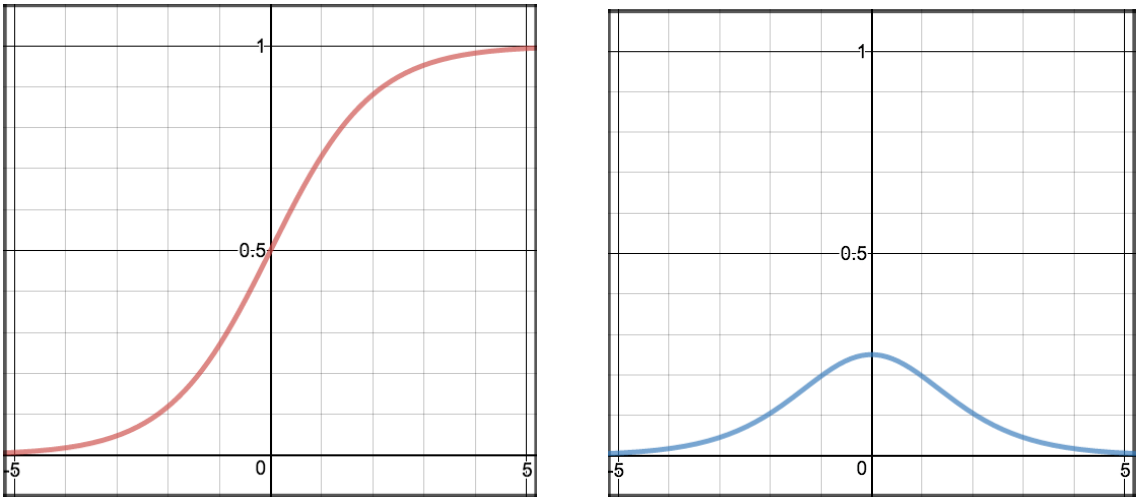

Figure 2.9. Illustration of the Sigmoid function and the derivative form (Li, 2019).

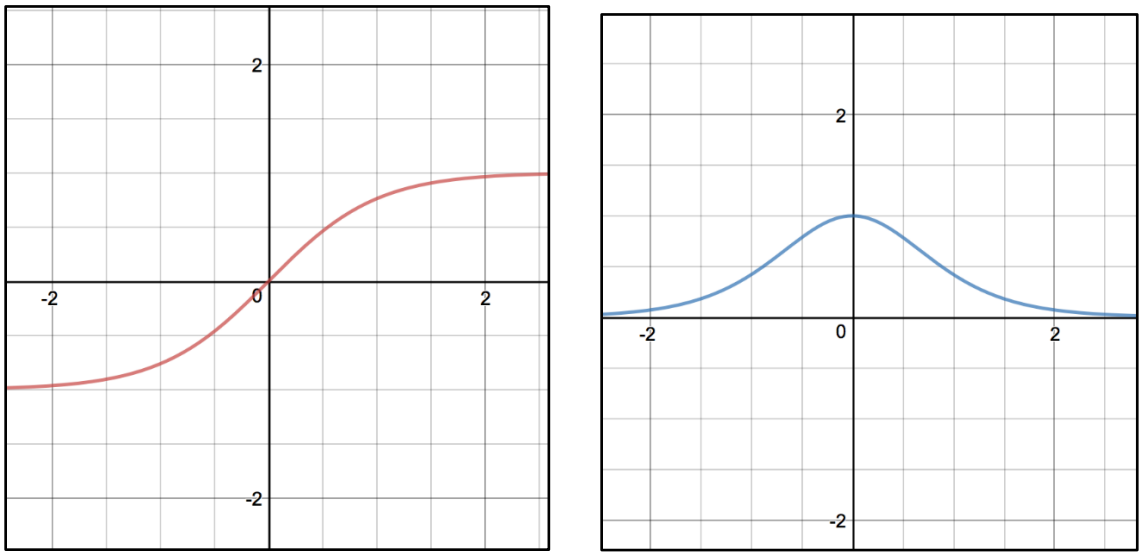

Figure 2.10. Illustration of the Tanh function and the derivative form (Li, 2019).

2.3.5 Model Evaluation

After a data-driven model is created, it is important to use evaluation metrices to evaluate the model performance. In a pattern recognition or classification problem, each item in the total population has a true label and a predicted label. As shown in figure 2.11, the relevant elements refer to the elements that are positive in reality, and the selected elements are these that are predicted to be positive by the model.

Because both the actual label and the prediction label have two different situations to be either positive or negative, there are totally four different situations that can happen regarding the results of the predicted label and the actual label. Firstly, If the predicted label and the actual label are both positive, this can be considered as a true positive prediction (TP). Secondly, if the predicted label and the actual label are both negative, this can be considered as a true negative prediction (TN). Thirdly, if the predicted label is positive but the actual label is negative, this can be considered as a false positive (FP) prediction. Fourthly, if the predicted label is negative but the actual label is positive, this can be considered to be a false negative (FP) prediction.

Ture positive and true negative predictions are correct predictions since the predicted values match their actual values. On the contrary, both false positive and false negative predictions can be counted as false predictions since their predicted values do not match the actual values.

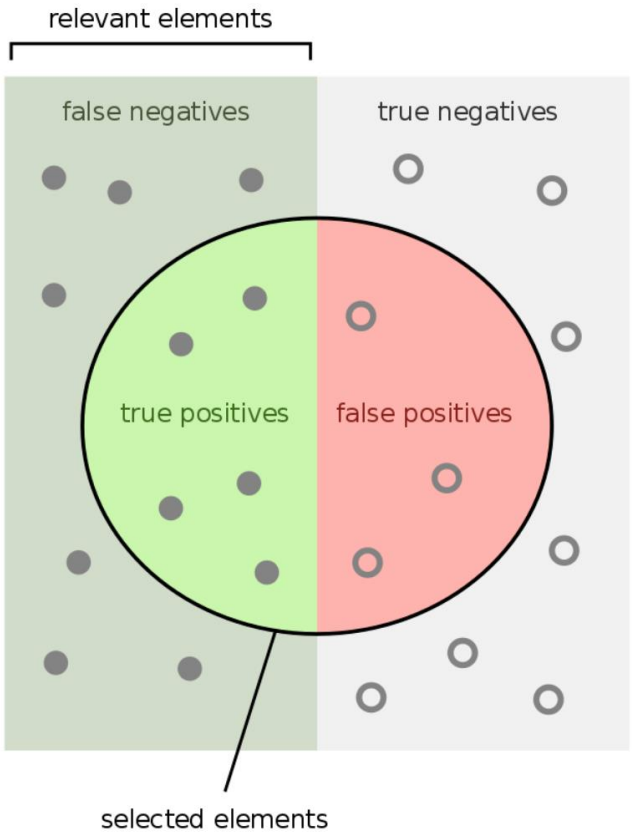

Figure 2.11. Illustration of predicted values and actual values (Chris, 2019).

Take a medical screening test that predicts whether a person has a disease or not for example, each person provides a data point in the dataset, and either has or does not have the disease in reality. The relevant elements refer to the people who have the disease in reality.

By taking the test, it predicts whether the patient has or does not have the disease. The selected element refers to these who are predicted to have the disease by the test. If the outcome of the test is positive, the test predicts the patient has the disease. If the outcome is negative, it predicts the patient does not have the disease. For each patient, the result may or may not be correct based on each person's real situation. There are totally four different cases:

True positive case: the person has the disease and the test predicts positive;

True negative case: the person does not have the disease and the test predicts negative;

False positive case: the person does not have the disease and the test predicts positive;

False negative case: the person has the disease and the test predicts negative.

Based on the number of different predictions for a test, there are a variety of parameters that can be utilized to evaluate models' performance, including accuracy, precision, recall, and specificity, etc.

Accuracy represents the fraction of the total amount of correct predictions among all the samples. Figure 2.12 shows the common parameters including precision, recall, and specificity. Precision, which is also called positive predictive value, represents the portion of the true positive predictions in all selected elements (all positive predictions). Recall, which is also called sensitivity or true positive rate, represents the portion of true positive predictions among all relevant elements (all positive data points in reality). Specificity, which is called the true negative rate, represents the fraction of true negative predictions among the un-relevant elements (all negative samples in reality).

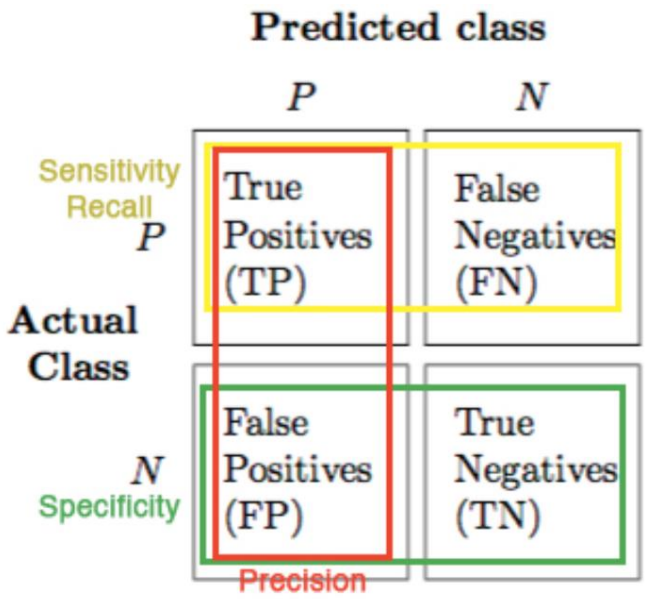

Figure 2.12. Illustration of recall, precision, and specificity (Tran, 2016).

Besides, F1 score is a parameter that considers both precision and recall. It is calculated from the harmonic average of both precision and recall. The equations for the above parameters are shown as follows.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{F1} = 2\,\frac{Precision \cdot Recall}{Precision + Recall}$$

2.4 Dataset

The dataset of this study is from Society of Exploration Geophysicists (https://github.com/seg). The origin of the dataset is from Dubois et al. (2007). The dataset contains seven features that come from seven different well logs from the Panoma gas field in the

Kansas state, United States. The dataset also contains a facies label that is pre-interpreted by geologists. There are totally 2783 observations.

The dataset contains seven variables. They are all created by well logging tools during logging operations as the tools travel from the bottom of the well to the surface ground. The dataset contains Gamma ray log (GR), which measures the amount of naturally emitted gamma radiation from the formation, can help differentiate lithology. Resistivity log (ILD_log10) is another log type contained in the dataset. It measures the electrical resistivity of the formation. The dataset includes Photoelectric effect log (PE) that measures photoelectric absorption ability of the formation and can be related to the minerality of the formation. The dataset also contains porosity logs including average neutron-density porosity log (PHIND) and Neutron-density porosity difference (DeltaPHI). In addition, the dataset includes a categorical parameter, Nonmarine/marine index (NM_M), to indicate whether it is nonmarine or marine environment. Besides, the depth information is included in the Relative position (RELPOS) log.

The dataset contains a total of nine different rock facies. They are represented by numbers from 1 to 9 in table 2.1. As showing in table 2.1, facies 2, 3, 6, and 8 have more than 300 observations, and the rest have less than 200 observations. Figure 2.13 shows the distribution for the number of samples for each label.

Table 2.1. Nine different rock facies in the dataset.

Facies Labels	Facies Name	Number of observations	Adjacent Facies
1	Nonmarine sandstone	170	2
2	Nonmarine coarse siltstone	649	1,3
3	Nonmarine fine siltstone	498	2
4	Marine siltstone and shale	177	5
5	Mudstone	198	4,6
6	Wackestone	391	5,7,8
7	Dolomite	81	6,8
8	Packstone-grainstone	458	6,7,8
9	Phylloid-algal bafflestone	161	7,8

The last column in table 2.1 is the adjacent facies. This column is added since the facies in the dataset aren't discrete. In fact, it is common that facies gradually mix with one another. Some samples have adjacent facies that are very close to each other. Mislabeling within these adjacent facies are expected to happen. Thus, information for the adjacent facies is added in the last column for each facies.

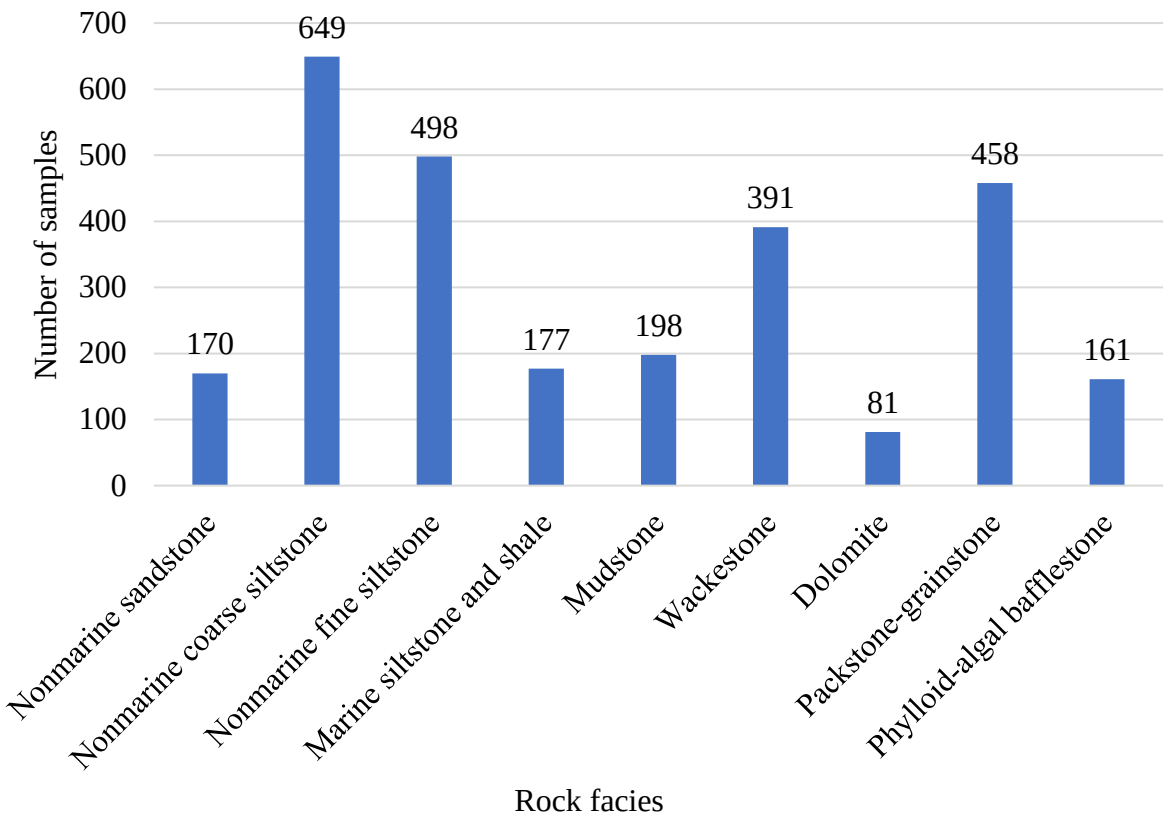

Figure 2.13. Distribution of number of each rock facies in the dataset.

The statistics of the five numerical features are shown as follows. It contains information for the count, mean, standard deviation (Std), minimum (Min), 25th percentile, 50th percentile (median), 75th percentile, and maximum (Max) values for each feature.

Table 2.2. Statistics of the input features.

Feature Name	Count	Mean	Std	Min	25%	50%	75%	Max
GR	2783	66.25	31.61	13.25	46.08	65.54	80.71	361.15
ILD_log10	2783	0.64	0.24	-0.03	0.50	0.63	0.81	1.48
DeltaPHI	2783	3.75	5.05	-21.83	1.30	3.58	6.50	18.50
PHIND	2783	13.12	7.39	0.55	8.17	11.90	16.14	84.40
PE	2783	3.81	0.89	0.20	3.20	3.60	4.40	8.09

The five numerical features are normalized using the following equation. The statistics after normalization are shown in table 2.3. After normalization, the mean of each feature becomes zero, and the standard deviation becomes one.

$$x \leftarrow \frac{x - mean(x)}{std(x)}$$

Table 2.3. Statistics of the input features after normalization.

Feature Name	Count	Mean	Std	Min	25%	50%	75%	Max
GR	2783	0.00	1.00	-1.68	-0.64	-0.02	0.46	9.33
ILD_log10	2783	0.00	1.00	-2.77	-0.61	-0.07	0.69	3.45
DeltaPHI	2783	0.00	1.00	-5.07	-0.49	-0.03	0.54	2.92
PHIND	2783	0.00	1.00	-1.70	-0.67	-0.16	0.41	9.65
PE	2783	0.00	1.00	-4.03	-0.68	-0.23	0.66	4.80

A correlations plot is created for the normalized numerical features to show the overall relationship for each input feature. As showing in Figure 2.14, ILD_log10 (resistivity) and PE (Photoelectric effect) have a weak positive relationship. In contrast, PHIND (average neutron-density porosity) and PE (Photoelectric effect) show a negative relationship. The diagonal part of figure 2.14 shows the distribution of each feature.

Figure 2.14 also indicates that only by plotting these features on a scattered plot is not enough to understand the relationships among all the features, since the co-relationship between each single feature is not strong. Thus, data-driven models that are established from more numbers of features are needed to better characterize the complex non-linear relationship between all the features.

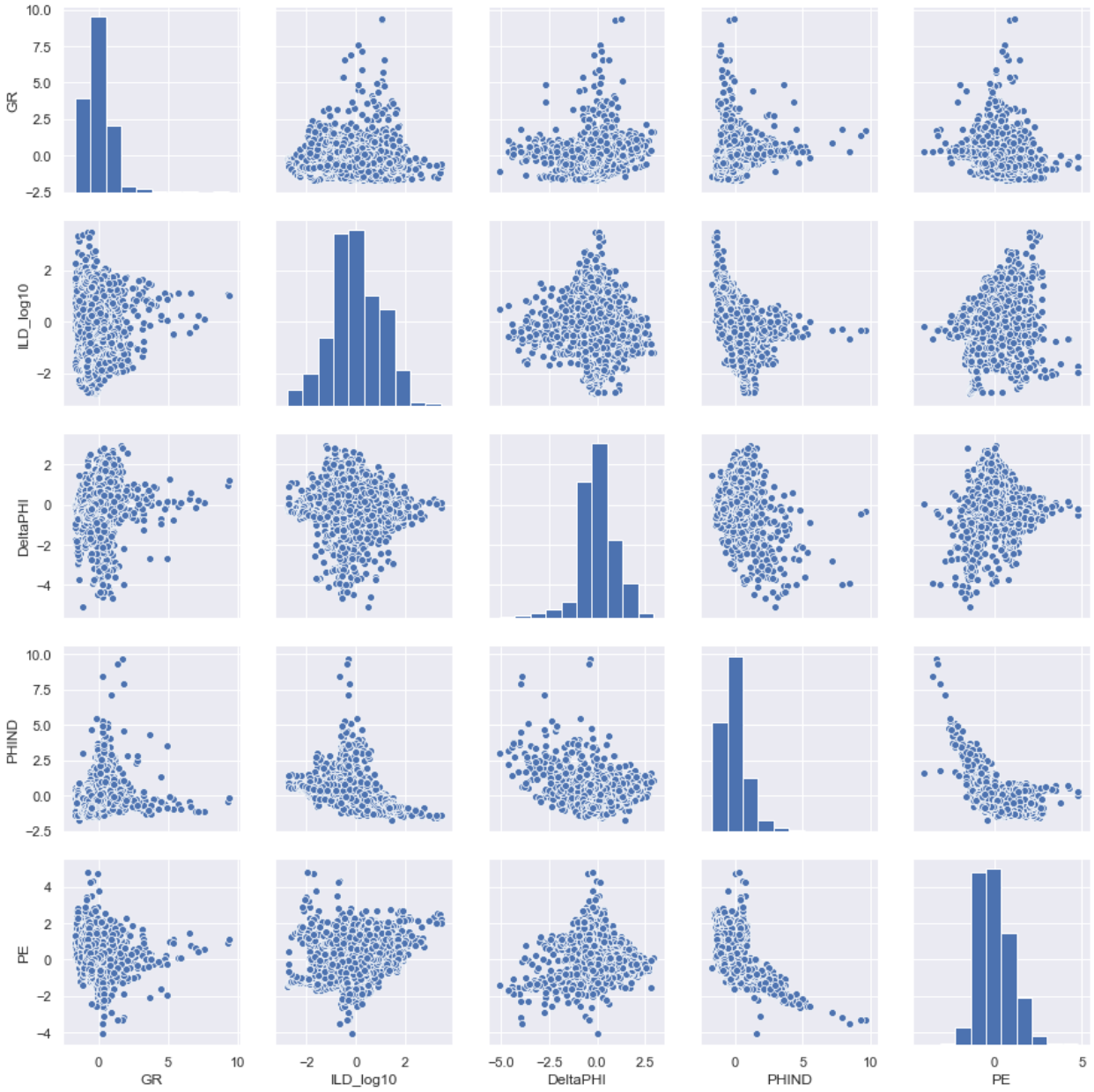

Figure 2.14. Correlation plot for the input features after normalization.

Figure 2.15 shows the correlation matrix between the input features. Red color represents a positive relationship, blue color represents a negative relationship. As consistent to the correlationship in figure 2.14, ILD_log10 (resistivity) and PE (Photoelectric effect) have a weak relationship. PHIND (average neutron-density porosity) and PE (Photoelectric effect) show a negative relationship.

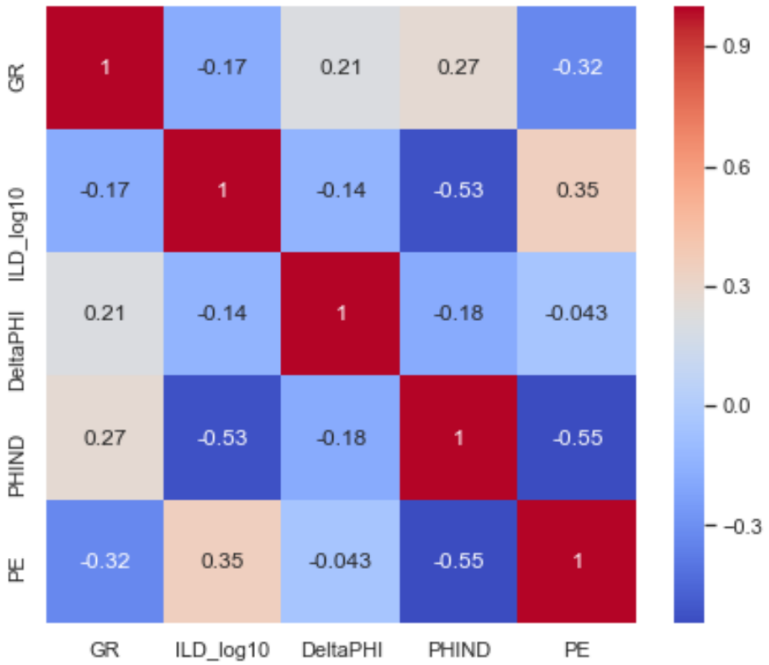

Figure 2.15. Correlation between each variable. Red color represents a positive relationship. Blue color represents a negative relationship.

2.5 Results

The dataset is split into a training set and a testing set. The models are trained using 80% of the total data points and the rest 20% are used for testing. All four models are evaluated based on their performance on the testing set to select the best model.

The testing accuracy for each model are shown in table 2.2. Both the facies accuracy and the adjacent facies accuracy are used to compare the model performance. Among the four models, the random forest method has overall the best performance. It has the highest testing accuracy as 0.691 for the facies, and the second highest testing accuracy as 0.921 for the adjacent facies. Decision tree has the second highest testing accuracy for facies as 0.618 and the third highest accuracy for adjacent facies as 0.903. SVM has the third highest testing accuracy for facies as 0.585 and the highest testing accuracy for adjacent facies as 0.926. Neural network has relatively lowest accuracy for both the facies and the adjacent facies as 0.553 and 0.895 respectively.

Table 2.4. Testing accuracy for each model.

Method	Accuracy for Facies	Accuracy for Adjacent Facies
Decision Tree	0.618	0.903
Random Forest	0.691	0.921
SVM	0.585	0.926
Neural Network	0.553	0.895

Figure 2.16 shows the confusion matrix for the predicted facies and actual facies from the random forest model results. The diagonal part shows the number of correct predictions. Dark blue color represents a higher number indicating the model is performing good for that particular class. Overall, the diagonal boxes have the largest number for each row and column meaning that the model is generally performing well for most classes.

In figure 2.16, there exist boxes that are neighboring to the diagonal boxes and they also have slightly dark colors, indicating that there are misclassified labels for the adjacent facies. This is because facies in the dataset aren't discrete, and gradually blend into one another. Thus, it is common for facies to be predicted as their neighboring facies. Figure 2.17 shows the normalized confusion matrix for the predicted facies and actual facies from the random forest model results. The diagonal part shows the percentage of correct prediction for each label.

Figure 2.18 and shows the confusion matrix for the predicted adjacent facies and actual adjacent facies from the random forest model results. The diagonal boxes have larger values than these in figure 2.16, meaning that there are more correct predictions. This is because neighboring predictions are considered to be correct predictions in this situation. Figure 2.19 shows the normalized confusion matrix for the predicted adjacent facies and actual adjacent facies from the random forest model results. The diagonal boxes also show larger values than these in figure 2.17, which is due to the same reason.

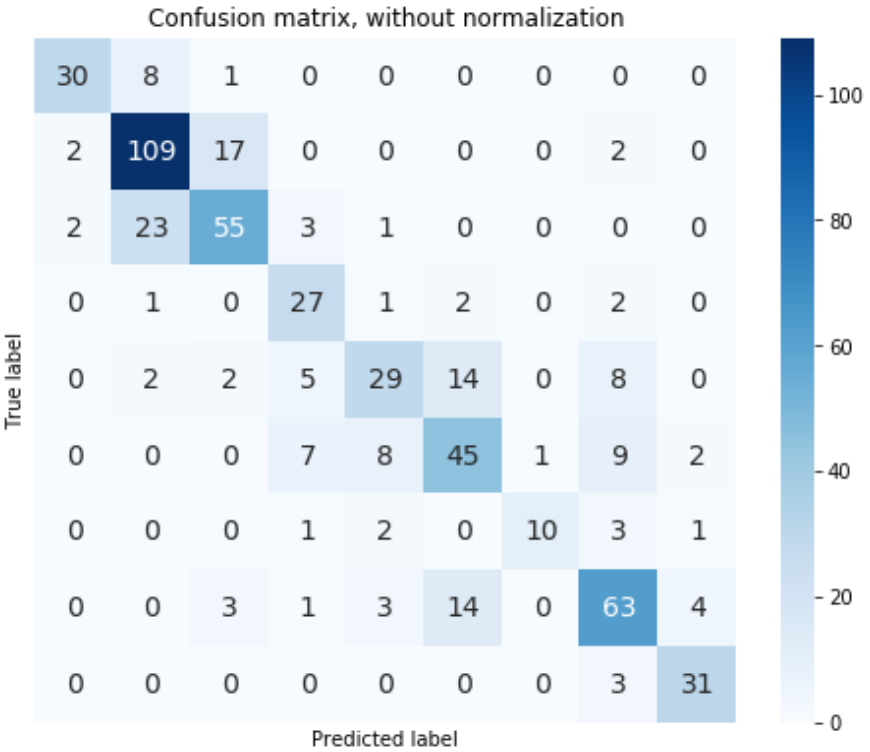

Figure 2.16. Confusion matrix for the predicted facies and actual facies from the random forest model results. The diagonal boxes show the number of correct predictions.

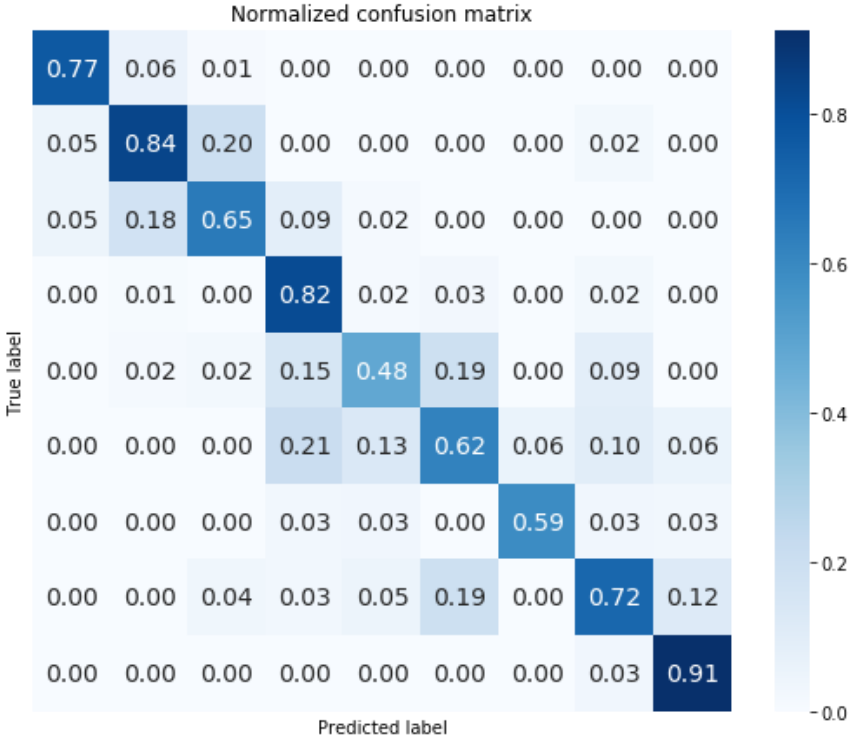

Figure 2.17. Normalized confusion matrix for the predicted facies and actual facies from the prediction results of the random forest model.

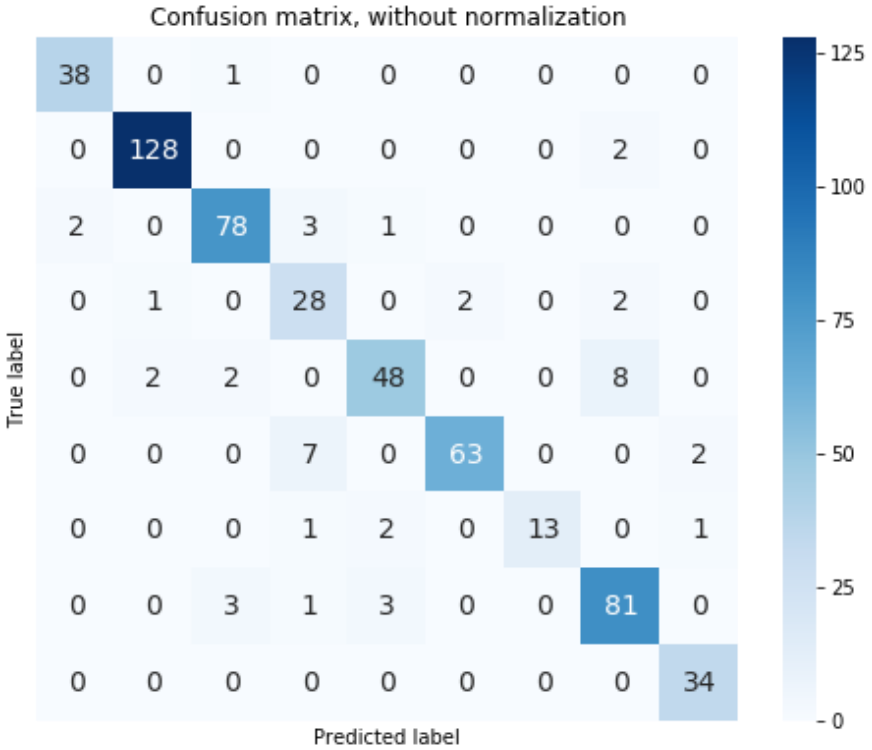

Figure 2.18. Confusion matrix for the predicted adjacent facies and actual adjacent facies from the prediction results of the random forest model.

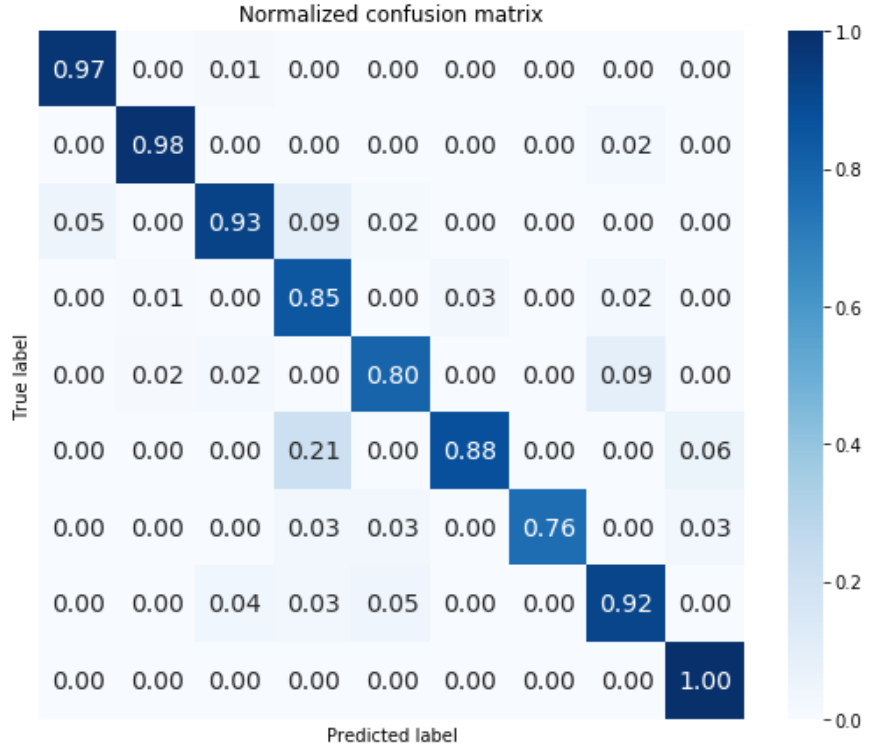

Figure 2.19. Normalized confusion matrix for the predicted adjacent facies and actual adjacent facies from the prediction results of the random forest model.

Evaluation metrics are calculated for each facies from the prediction results of the random forest model. Table 2.3 shows the precision, recall, and F1-score for each label. Figure 2.20 shows the plot for these evaluation metrices. The last row in table 2.3 shows the total precision, recall and F-1 score to be 0.72, 0.72, and 0.71 respectively.

Table 2.5. Evaluation metrics for each facies from the random forest model results.

Facies Labels	Facies Name	Adjacent Facies	Precision	Recall	F1 score
1	Nonmarine sandstone	2	0.88	0.77	0.82
2	Nonmarine coarse siltstone	1,3	0.76	0.84	0.8
3	Nonmarine fine siltstone	2	0.71	0.65	0.68
4	Marine siltstone and shale	5	0.61	0.82	0.7
5	Mudstone	4,6	0.66	0.48	0.56
6	Wackestone	5,7,8	0.6	0.62	0.61
7	Dolomite	6,8	0.91	0.59	0.71
8	Packstone-grainstone	6,7,8	0.7	0.72	0.71
9	Phylloid-algal bafflestone	7,8	0.82	0.91	0.86
Total	na	na	0.72	0.72	0.71

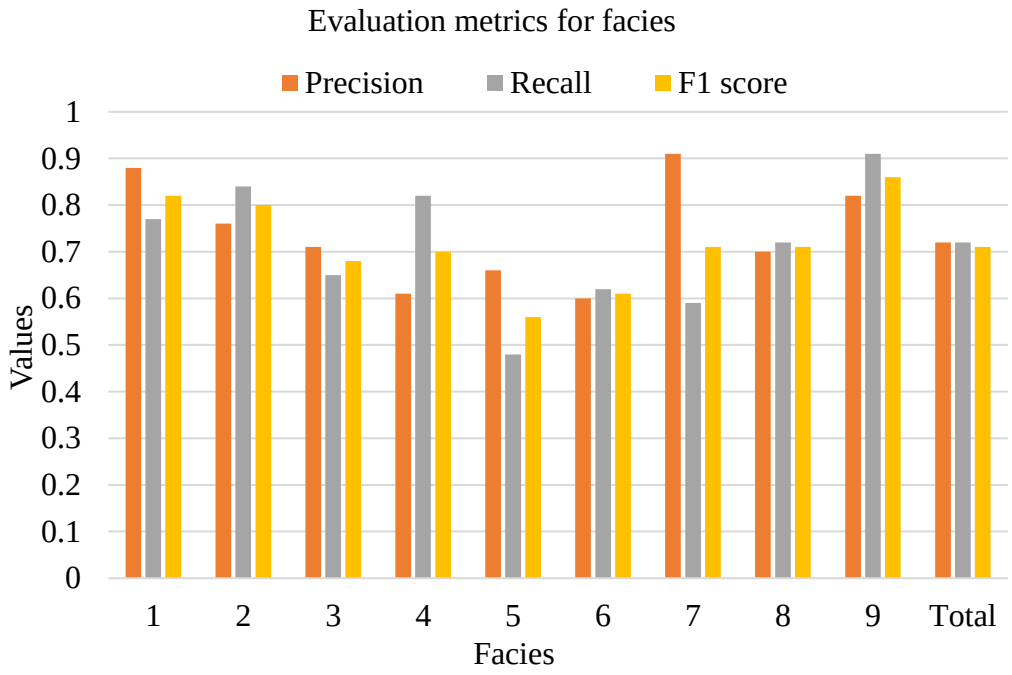

Figure 2.20. Evaluation metrics for each facies from the random forest model predictions.

Evaluation metrics are also calculated for the adjacent facies from the prediction results of the random forest model. Table 2.4 shows the precision, recall, and F1-score for each adjacent facies. Figure 2.21 shows the plot for these evaluation metrices. The last row in table 2.4 shows the total precision, recall and F-1 score to be 0.92.

It is also noticed that the precision, recall and F1-score for the adjacent facies in table 2.4 are higher than these for the facies in table 2.3. This is because neighboring predictions are considered to be correct predictions in this situation.

Table 2.6. Evaluation metrics for each adjacent facies from the random forest model results.

Facies Labels	Facies Name	Adjacent Facies	Precision	Recall	F1 score
1	Nonmarine sandstone	2	0.95	0.97	0.96
2	Nonmarine coarse siltstone	1,3	0.98	0.98	0.98
3	Nonmarine fine siltstone	2	0.93	0.93	0.93
4	Marine siltstone and shale	5	0.7	0.85	0.77
5	Mudstone	4,6	0.89	0.8	0.84
6	Wackestone	5,7,8	0.97	0.88	0.92
7	Dolomite	6,8	1	0.76	0.87
8	Packstone-grainstone	6,7,8	0.87	0.92	0.9
9	Phylloid-algal bafflestone	7,8	0.92	1	0.96
Total	na	na	0.92	0.92	0.92

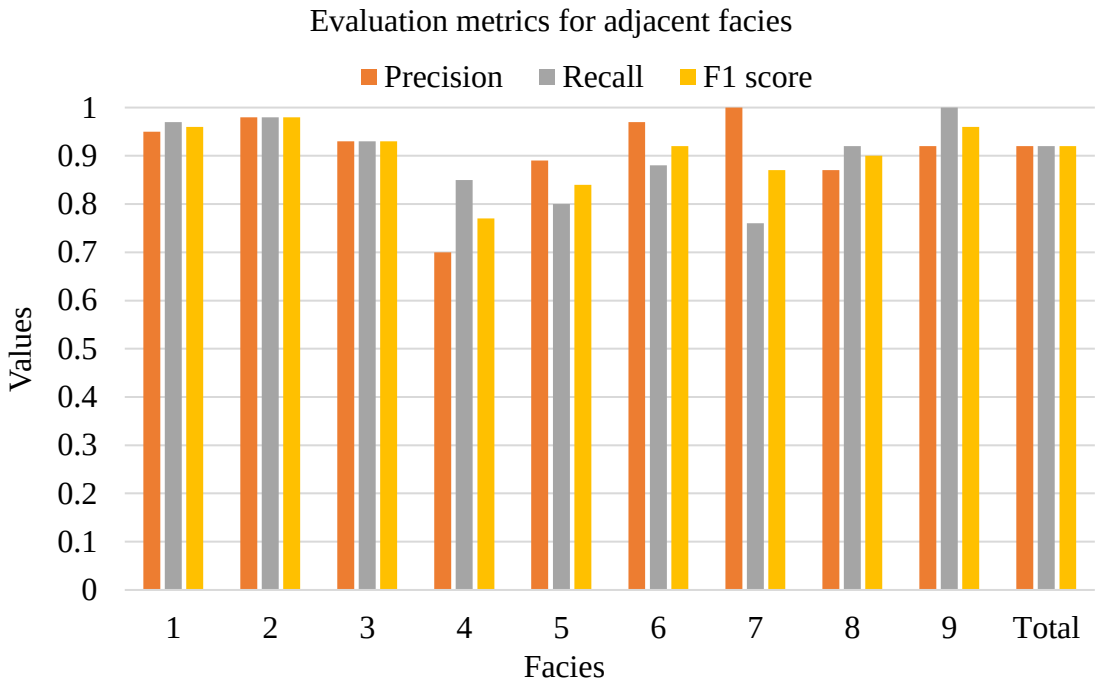

Figure 2.21. Evaluation metrics for each adjacent facies from the random forest model predictions.

2.6 Discussion

It is important to evaluate the model performance using a combination of evaluation metrics including accuracy, precision, recall, and F1 score. Accuracy represents the portion of correct predictions among all labels. A higher number means that there are a greater number of correct predictions. Precision (positive predictive value) represents the fraction of true positive among the selected elements (all positive predictions). Recall (sensitivity or true positive rate) represents the fraction of true positive among the relevant elements (all positive samples in reality). There are cases when two models may have similar accuracy. Thus, it is important to combine accuracy, precision, and recall to select the optimal model.

Particularly in cases such as medical tests, the amount of false negative is an important factor to be considered. A false negative prediction represents the case that the model predicts a patient that is positive in reality to be negative, thus may result in delay of diagnostics and treatment. Even

a small increase of the number or percentage of false negative predictions may significantly affect the model performance. In these cases, recall (sensitivity or true positive rate), which is the fraction of true positive among the relevant elements (all positive samples in reality) needs to be considered to evaluate the model's effectiveness, because it considers the prediction results for all relevant elements including both true positive and false negative predictions.

Besides, the models are trained from the dataset collected from the Hugoton and Panoma gas fields (Dubois et al., 2007). So far, the trained models are evaluated using testing data collected in the same field. In order to acquire a model that can be applied to more areas, new data can be added to the model to re-train the model to explore more general relationships between well logs and rock facies.

2.7 Conclusion

This chapter describe the application of using petrophysical well logs and different data analytics models to automatically classify rock facies. Four different types of models, which include decision tree, random forest, support vector machine and neural network, are created to find the best model in predicting rock facies from well log values.

Among the four models, random forest method has overall the best performance for this dataset. It has the highest testing accuracy as 0.691 for the facies classification. The testing accuracy is 0.921 for the adjacent facies classification, which is the second highest. It was slightly less but very close to the highest accuracy of classifying adjacent facies from the support vector machine method, which is 0.926. However, the accuracy for classifying facies from the support vector machine model is 0.585, which is much less than that of the random forest. So, the random forest method has overall the best performance among the four tested models for this dataset.

For all the four different methods, the testing accuracy and evaluation metrics including precision, recall and F-1 score for the adjacent facies classification are higher than these for the facies classification. This is because neighboring predictions are considered to be correct predictions for the classification results of adjacent facies.

Future work may include adding more types of well logs to further increase the prediction accuracy. More data points and more data analytics methods may also be tested to see whether they can help increase the model performance in classifying rock facies using well logs.

Chapter 3. Predicting Petrophysical Properties Using Seismic Attributes and Artificial Neural Networks

This chapter explores the application of using data-driven modeling methods to establish the statistical relationships between seismic attribute values from a 3D seismic survey and petrophysical properties from well logs. Artificial neural network models are created to explore the relationship between these two data types. Such relationships and models can be used for the optimization of exploration and production operations.

3D seismic data can be used to extract various seismic attribute values at all locations within the seismic survey. Well logs provide accurate constraints on the petrophysical values along the wellbore. Artificial neural network models are utilized to establish the statistical relationships between seismic attributes and petrophysical data. Since seismic data are at the reservoir scale and are available at every sample cell of the seismic survey, these relationships can be used to estimate the petrophysical properties at all locations inside the seismic survey.

In this study, the Teapot dome 3D seismic survey is selected to extract seismic attribute values. A set of instantaneous seismic attributes, including curvature, instantaneous phase, and trace envelope, are extracted from the 3D seismic volume. Deep learning neural network models are created to establish the relationships between the input seismic attribute values from the seismic survey and petrophysical properties from well logs. Results show that a deep learning neural network model with multi-hidden layers is capable of predicting porosity values using extracted seismic attribute values from 3D seismic volumes. Utilization of a subset of seismic attributes improves the model performance in predicting porosity values from seismic data.

3.1 Introduction

A 3D seismic survey provides information about the subsurface. There are many attributes that can be calculated from 3D seismic data. For example, curvature measures the extent of deviation away from a straight line (Robert, 2001, Chopra and Marfurt, 2007). Curve lines have greater values of curvature, and strait lines' curvature values are zero. Most Positive Curvature (MPC) measures the speed of the positive change of slope on a plane. MPC can be used to highlight bumps in seismic reflections, and it can be used to detect fractures (Khair et al., 2012). As it shown in figure 3.1, different geologic features have different curvature values. An anticline has a positive curvature value, and a syncline has a negative curvature value. MPC is also related to Most Negative Curvature (MNC), which is another seismic attribute that can be calculated. It highlights sags and synclines.

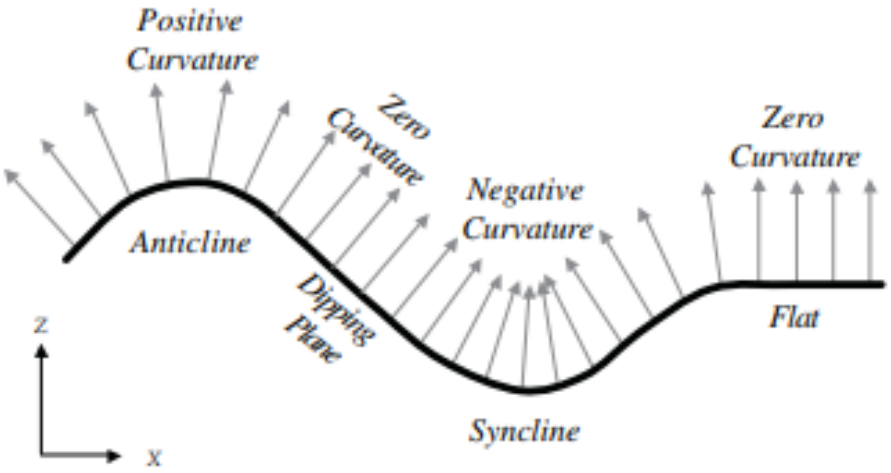

Figure 3.1. Different geologic features have different curvatures. Anticlines have positive curvature values, and synclines have negative curvature values. From Robert, 2001.

Figure 3.2 shows a case study which uses seismic attribute to identify a major fault and other fracture zones (Khair et al., 2012). In this study, the MPC and MNC attributes were calculated and the result clearly displays the existence of structural features including major faults and several fracture networks. Fault population and trace information is collected from the curvature attribute

maps. Regional stress information is determined from fault population analyses, and it matches the regional stress field in the area (Khair et al., 2012).

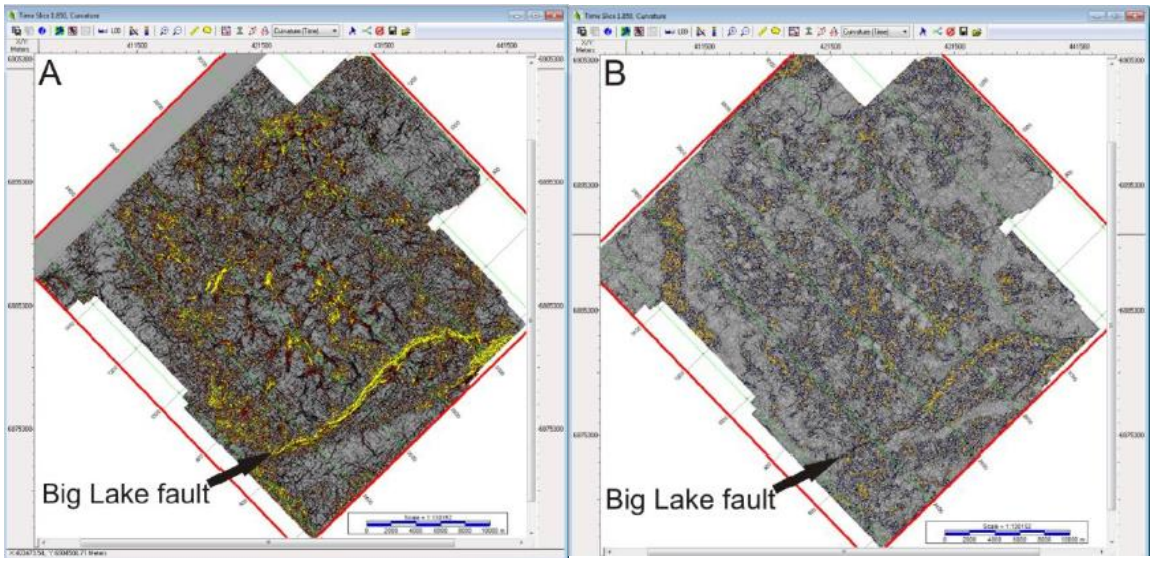

Figure 3.2. Use most positive (left) and most negative (right) curvature attributes to identify fault zones. From Khair et al., 2012.

Usually, a function may be used to describe a relationship. For example, for the equation Y=F(X) in which X is the input and F is the function, Y is the output. Typically, when we know X as the input, and F as the function, we can look for Y. This can be considered as a forward modeling problem. In the industry, there is a similar situation. As shown in figure 3.3, when a geologic model is created, we know the formations boundaries and the structures. If we know the physics of how seismic wave propagates in the subsurface, a synthetic seismic response can be created. This is a forward modeling problem.

In the second case, when we have Y as the output and the inverse relationship F^{-1}, X can be estimated. This is considered as an inverse modeling problem. In the industry, when a seismic survey is conducted, seismic inversion can be performed to estimate the formations boundaries and structures in order to create the geologic model.

In the third case, if there is X and Y, the relationship between X and Y which is F can be explored. This process can be called data-driven modeling. We look for relationship using input and output data via data analytics, or machine learning. This is called data-driven modeling.

In this study, seismic attribute values are extracted from a 3D seismic survey to explore the relationship between these seismic attribute values and petrophysical properties from well logs. Data-driven models are created to quantify these relationships. Because seismic data is collected at every grid cell within the seismic survey, it is available at every location inside the reservoir. Thus, this established relationship between seismic attribute values and petrophysical properties can be used to estimate the petrophysical properties at all locations in the reservoir. This significantly reduces the cost of running additional well logging operations in the field.

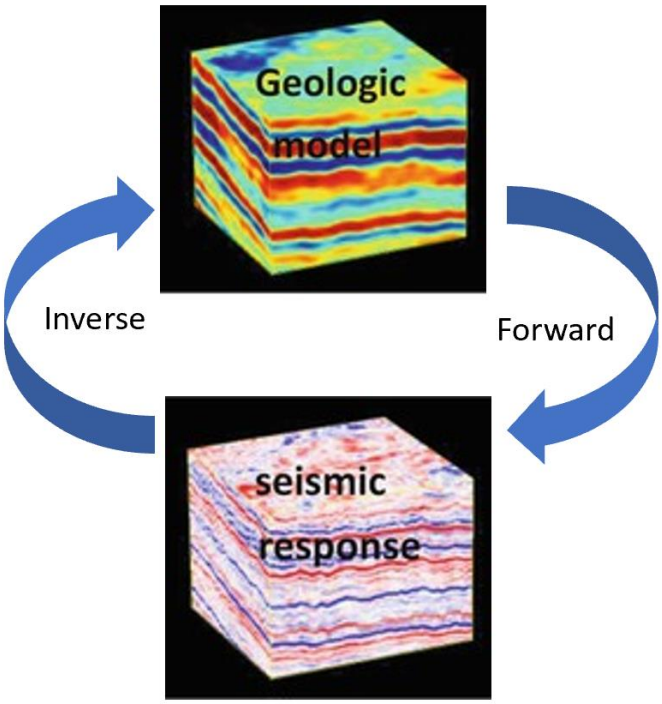

Figure 3.3. Illustration of inverse and forward modeling between geologic models and seismic responses (Abokhodair, 2008).

3.2 Literature Review

Encouraged by the technological improvement in data collection and the increased computational power, data-driven modeling has become an emerging technique in the oil and gas industry. Recently it has been increasingly explored in many aspects including reservoir modeling, optimization of drilling and completion design, production prediction, etc. It has been used in drilling engineering to improve drilling performance and efficiency (Chen et al., 2019; Yin et al., 2020; Luo et al., 2020). It also has been used in the optimization of completion parameters for unconventional resources (Schuetter et al., 2015; Mishra and Lin, 2017; Schuetter et al., 2018; Wang and Chen, 2016).

Understanding subsurface properties is significant in reservoir modeling and characterization. Research has been conducted to better characterize subsurface properties including petrophysical properties, reservoir properties, fracture properties, etc. (Zhou et al., 2014; Wang et al., 2018, 2019; Huang et al., 2018; Feng et al., 2019; Yang et al., 2020; Wei et al., 2020). The application of data-driven modeling has been used to better characterize petrophysical properties (Li and Zhang, 2016, Liu et al., 2020). Existing research also has been conducted to improve efficiency for well log interpretation (Roy et al., 2013, Zhao et al., 2014, Wrona et al., 2018).

3D seismic data provides important information about the subsurface. Combining with data-driven modeling methods, 3D seismic data has been used to better characterize subsurface properties such as faults and fracture properties. Zhang and others (2014) used data-driven modeling methods for automatic fault detection from seismic traces. Udegbe and others (2018) used amplitude-based statistics for seismic fracture identification. They verified the methods for identifying discrete fractures by combining both 3D seismic data and full-bore micro-image well

log data. Di and Gao (2017), Zheng and others (2014) have used 3D seismic data and machine learning techniques for fault detection and extraction.

One advantage of 3D seismic data is that it is collected and available at every grid within the seismic survey. Thus, it can provide information at every location inside the reservoir. In this study, 3D seismic data and well log data are analyzed to build the relationship of seismic attribute values and petrophysical and reservoir properties. The model uses 3D seismic data to predict porosity values from the well log. Since 3D seismic attribute values are at all locations of the seismic survey, which is usually conducted in a relatively large scale. The established model can be utilized to estimate these properties at all sample cells of the reservoir.

3.3 Methodology

3.3.1 Seismic Attribute Analyses

There are many attributes that can be created from seismic data. Table 3.1 and 3.2 lists the common curvature attributes and rock-solid attributes that can be generated from seismic data (Holdaway, 2014).

Table 3.1. List of common curvature attributes (Holdaway, 2014).

Primary Outputs	Geometric	High Resolution	Shapes	Semblance
Min. Curvature	Dip Azimuth	Dip Curvature	Dome	Cross Correlation of Real vs. Imaginary
Max. Curvature	Dip Magnitude	Gaussian Curvature	Ridge	Derivative of Total Energy
Most Positive Curvature	Inline Apparent Dip	Strike Curvature	Saddle	Outer Product
Most Negative Curvature	Cross-line Apparent Dip	Angular Unconformity	Bowl	

Table 3.2. List of common rock-solid attribute (Holdaway, 2014).

Instantaneous Attributes	Wavelet Attributes	Geometrical Attributes
Real Part of Complex Trace	Wavelet Phase	Event Continuity
Imaginary Part of Complex Trace	Wavelet Frequency	Sand/Shale Ratio
Trace Envelope	Wavelet Q Factor	Dip Variance
Instantaneous Phase	Dominant Frequency	Instantaneous Dip
Instantaneous Q Factor	Apparent Polarity of Wavelet	Dip Azimuth

In this study, three instantaneous attributes were extracted from the seismic volume, including curvature, instantaneous phase, and trace envelope.

Curvature represents the radius of a circle that is tangent to a curve or a surface for a 3D problem. Mathematically, it can be simply represented by the following equation:

$$K = \frac{d\omega}{dS} = \frac{1}{r}$$

Where K represents the curvature, and r represents the radius of the circle that is tangent to a surface or a curve. Figure 3.4 shows an illustration to determine the curvature on a curve. Curvature is a useful seismic attribute in quantifying the change of seismic amplitude. It helps improve the visualization of large-scale subsurface features. It can be used to identify discontinuities such as fault edges, fractures, channels, etc.

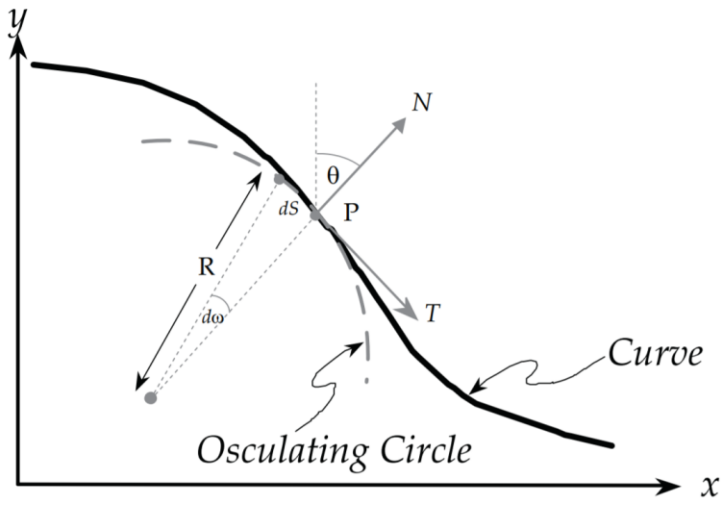

Figure 3.4. Illustration of the curvature attribute (Robert, 2001).

Instantaneous phase is a seismic attribute that is related to the propagation phase of the seismic wave front. The equation to calculate instantaneous phase can be shown as:

$$\phi(t) = arctan\,(\frac{Im\,F(t)}{Re\,F(t)})$$

Where F(t) is the seismic trace, Re F(t) and Im (Ft) are the real part and imaginary part of the complex trace. Instantaneous phase is useful in indicating lateral continuity, calculating the phase velocity, and visualizing bedding stratigraphy.

Trace envelope is also called instantaneous amplitude. It is a function of the real part and imaginary part of seismic trace. It can be calculated as

$$E(t) = \sqrt{(Re\,s(t))^2 + (Im\,s(t))^2}$$

Where Re s(t) represents the real part of the analytic trace S(t), and Im s(t) represents the imaginary part of the analytics trace. Trace envelope is useful in locating major subsurface features. It is a good indicator of bright spots in the subsurface since it is related to acoustic impedance contrast. It also helps identify unconformities, as well as changes in lithology, faulting, and depositional environment.

3.3.2 TensorFlow

This study uses artificial neural network to find the relationship between seismic data and petrophysical data. TensorFlow is used as the library to perform neural network analyses. TensorFlow is an open-source software library that is capable to process large amount of data. It was developed after DistBelief, which was Google's first generation of deep learning neural network algorithm. TensorFlow is the second generation of neural network system and released to be an open-source software library in November 2015. It is a cross platform software library and can run on various operation systems and platforms. It also can run on multi CPU and GPU serves and can reduce the running time.

With the strong capabilities of processing large amount of data, TensorFlow has been widely used in various fields to analyze data with high dimensions. It has been applied in speech recognition, text identification, sound classification, image processing and classification, computer vision, and any many other fields. In this study, TensorFlow is used to build the multi-layer neural network models to explore the relationships between seismic attribute values and petrophysical properties.

3.4 Dataset

3D seismic data is accessed from the Teapot Dome 3D Seismic Survey, 2007, which includes 2D and 3D Seismic, well log, core images, and GIS data. 3D seismic data and well log data are imported into a geologic model in Petrel. Seismic data is originally imported in the time domain and was converted into the depth domain so that seismic data and well log data can be correlated together.

When compiling the data that is used for training the neural network model, a single-point or multi-point convolution method can be used. This method is introduced in Hampson et al., 2001.

As showing in the figure 3.5 below, there are two different types of data compiling methods. Figure 3.5.a represents the method of using one seismic value as the input variable to correlate with the well log value. In figure 3.5.b, instead of using only one seismic data as the input data, five seismic amplitude values are compiled as the input variables and are correlated to one well log value. The utilization of multi-point convolution is to increase model performance, since the well log value can be affected by rock properties in its surrounding zones. In additional, Zhang and others (2018) compared the method of using one seismic value to correlate with one well log value with the method of using ten seismic values to correlate with one well log value. Their results showed that the model with ten seismic input variables can help improve the model performance in predicting lithology using seismic amplitude data.

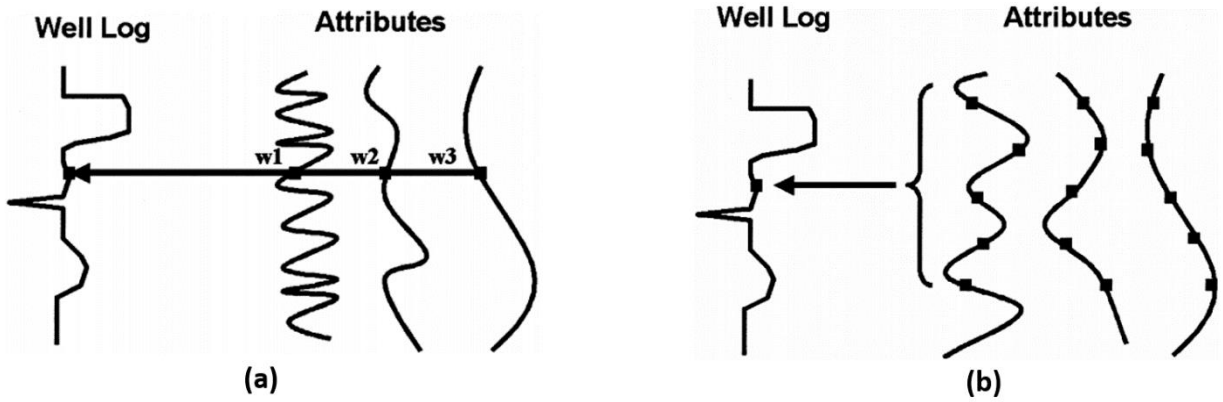

Figure 3.5. Two ways of data compilation. Figure 3.5a represents the case of correlating one seismic value to one well log value. Figure 3.5b represents the 5-point convulsion method, which uses five seismic values to correlate one well log value. (Hampson et al., 2001)

In this study, the 10-point convolution method is used to better address the impact of surrounding rocks on the log responses. A total of 26713 samples from four wells are collected to train the neuron network model. Four different cases were tested to compare the impact of adding different seismic attributes. The base case contains 10 seismic amplitude values as the 10 input variables. The second case contains 20 input variables which consists of 10 seismic amplitude values and 10 additional curvature attribute variables. The third case contains 30 input variables,

in which 10 additional instantaneous phase attribute values are added. The fourth case contain 40 input variables, which consists of 10 additional trace envelope amplitude variables and the 30 input variables from case 3. The output variable is neutron porosity log values for all four cases.

3.5 Results

The four datasets were compiled and fit into the deep learning neural network separately. In each case, the data was randomly split into a training set with 80% of the total amount of sample, and a testing set with 20% of the total amount of sample. The algorithm trains the model based on the values of the input variable and optimize the cost functions to find the best fit model. Figure 3.6 and 3.7 shows the change of Mean Absolute Errors (MAE) and Mean Squared Errors (MSE) with increasing epoch values. MAE and MSE values reach to the steady values around 300 epochs.

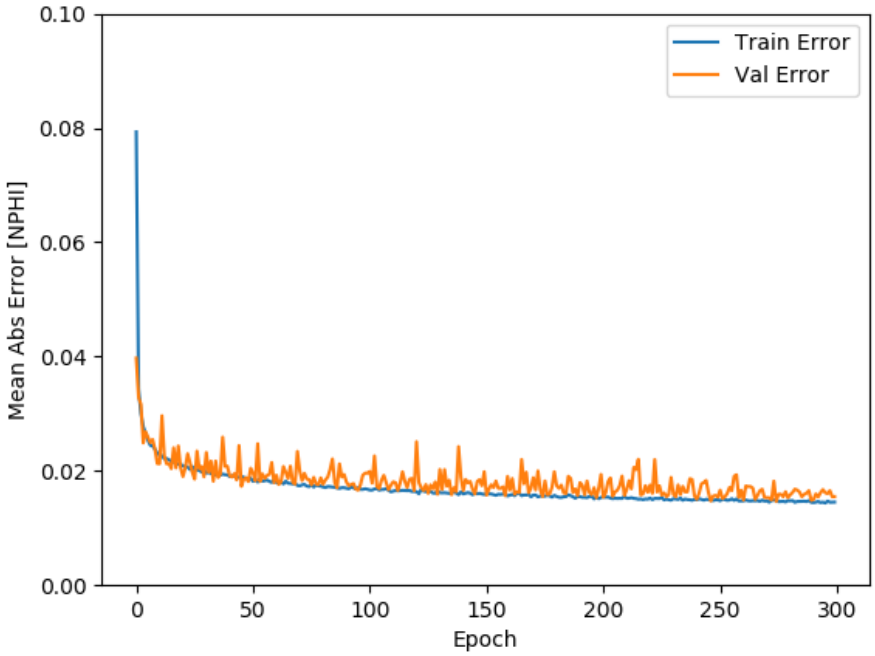

Figure 3.6. Mean absolute error during the training process of the neural network model. The total epoch number is 300.

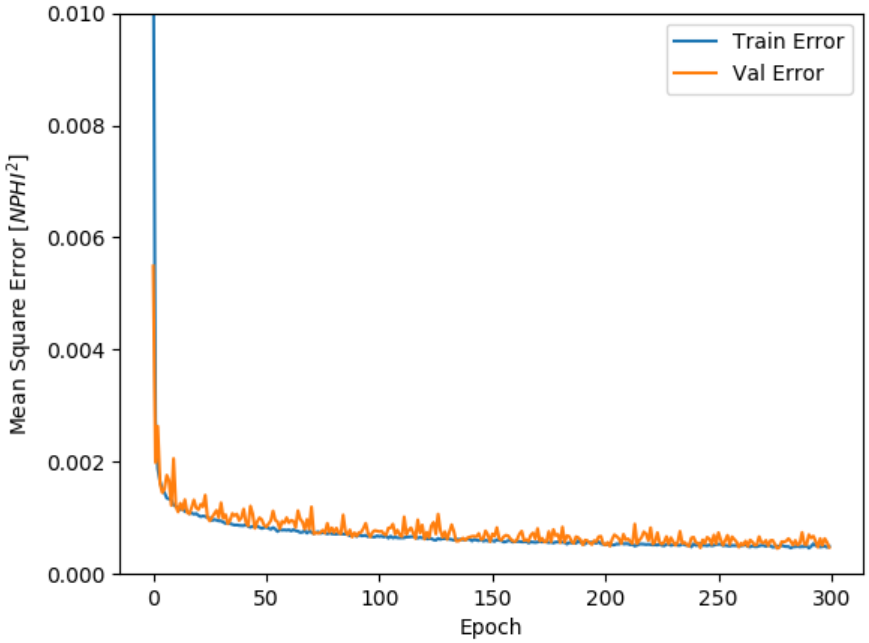

Figure 3.7. Mean square error during the training process of the neural network model. The total epoch number is 300.

After the optimal model is found, the testing dataset is input into the model to calculate the parameters that can be used to evaluate the model performance. These evaluation parameters include Mean Absolute Error (MAE), Mean Squared Error (MSE), and Coefficient of Determination (R^2). Table 3.3 shows the calculated MAE, MSE, and R^2 for four cases. These values are plotted in figure 3.8 and 3.9. As showing in table 3.3, figure 3.8 and 3.9, the model performance increases from case 1 to case 4, meaning that adding more seismic attributes improves the model performance in predicting porosity values.

Table 3.3. MAE, MSE, and R^2 for four cases

	MAE	MSE	R^2
Case 1	0.0324	0.0017	0.2783
Case 2	0.0209	0.0007	0.6517
Case 3	0.0180	0.0006	0.6936
Case 4	0.0161	0.0005	0.7775

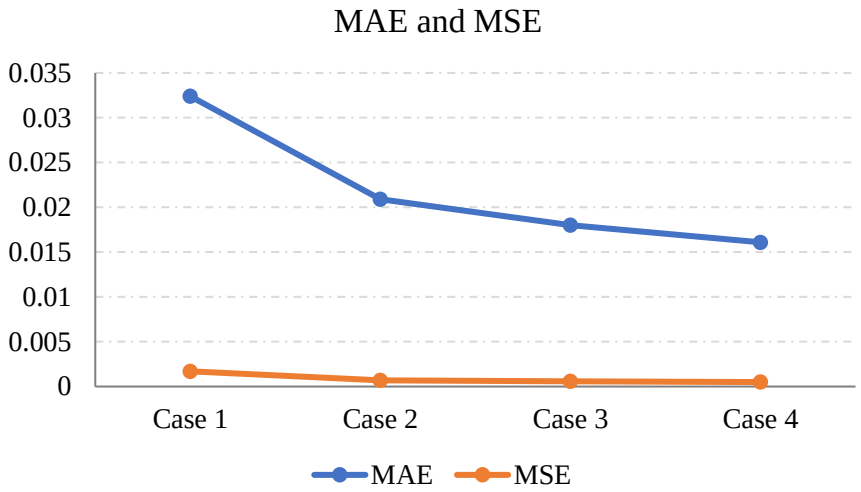

Figure 3.8. Calculated values of MAE and MSE for each case.

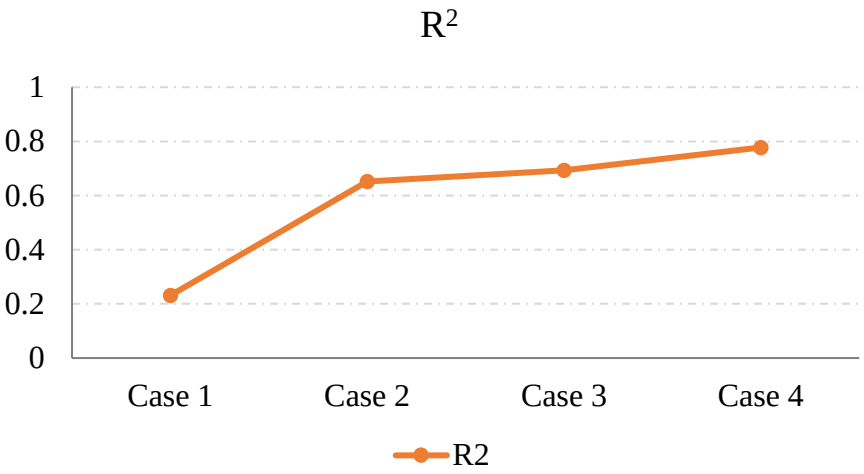

Figure 3.9. Calculated value of R^2 for each case.

Figure 3.10 shows the scattered plot of the predicted values with the true values for four different cases. Subplot a, b, c, and d are for case 1, 2, 3, and 4 respectively. In subplot a, the relationship between the predicted value and the actual value is not clear. With increased number of seismic attributes and increased number of input variables, it shows stronger co-relationship between the predicted value with the actual value. It indicates that the prediction accuracy is increased by adding more seismic attributes.

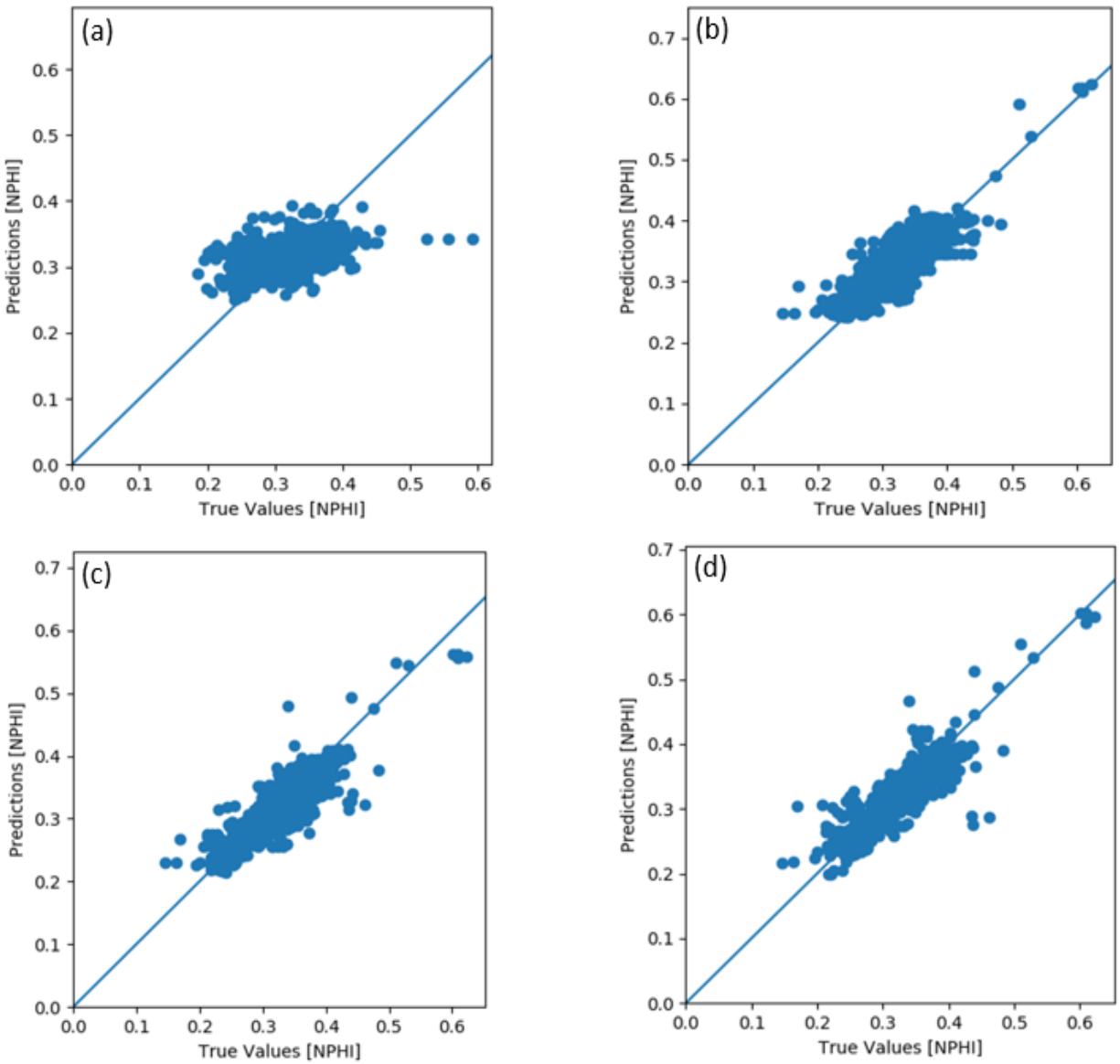

Figure 3.10. Plots of predicted values with true values for each case. Figure a, b, c, and d are for case 1, 2, 3, and 4 respectively.

After the model has been trained, the model was used to predict neutron porosity on two separate wells to compare the prediction results. The prediction results are show in the figure 3.11. Overall, there is a good match between the predicted values and the actual values. The predicted values show a similar trend with the change of porosity values on the actual log curves.

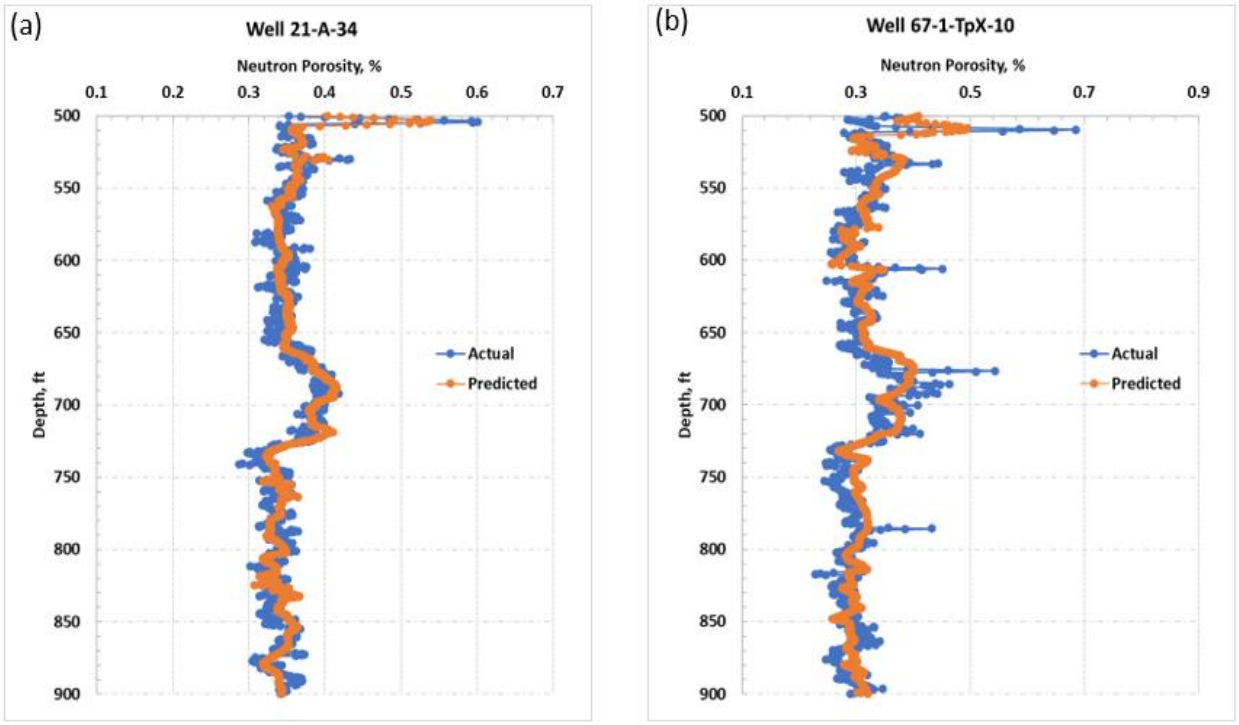

Figure 3.11. Plots of predicted values with true values for wells. Figure a and b represents cases for two wells.

It is noticed that the predicted values change less frequently compared to the values on the actual log. The regional variances on the predicted values are also smaller than that on the actual log. It could be because the input data was taken from a range of input values at surrounding zones, rather than from a single location or a smaller zone. In other words, the predicted value could be a smoothed estimation from each sample's neighboring zone, such that the predicted value has a relatively smaller regional variance than the actual log.

3.6 Discussion

In this study, seismic attributes are extracted from the 3D seismic survey to predict the neutron porosity well log values. Neuron porosity is the only porosity log that is available in most wells in the study area. Thus, it is used to co-relate the seismic attribute data.

Generally, there are different types of porosity logs, including density porosity, neutron porosity, and sonic porosity. Density porosity log measures the electron density of a formation.

The logging tool contains a device that emits gamma ray into the formation. The gamma ray interacts with the formation electrons and then scatters. The returned gamma ray from the formation can be collected by the detector inside the tool. The amount of returned gamma rays indicates the bulk density of the formation, which can be used to estimate the porosity using a mass-balance relationship.

The neutron log mainly measures amount of hydrogen in a formation. The logging device contains a source that emits neutrons into the formation. The emitted neutrons interact with nuclei of the formation and lose energy. When a neutron collides with a hydrogen atom, there is maximum neutron energy loss since a neutron has similar mass with a hydrogen atom. Thus, maximum energy loss happens when the formation contains large amount of hydrogen. The porosity of a formation can be related to the amount of hydrogen since the pore volumes contains fluid such as water and hydrocarbon, which contains high concentration of hydrogens.

Besides, acoustic well logs also can be used to evaluate porosity since sonic velocity is different for formation fluid and rock. If there exists pore space in the formation that contains fluid, it takes more time for the acoustic signal to travel from the transmitter to the receiver in the logging tool. Thus, the acoustic velocity, or the transit time of acoustic signal from the transmitter to the receiver, can be used to estimate porosity values.

Given the fact that different porosity values are estimated from different measurements of the properties of rocks and fluids, it is worthwhile to explore the possibility of creating the relationships between seismic attributes with different porosity types. Particularly, both seismic attributes and sonic porosity are related to the acoustic wave propagation properties of rocks and fluids in the subsurface. The model between seismic attributes and sonic porosity can be explored and may exhibit a stronger co-relationship.

3.7 Conclusion

In summary, deep learning neural network models are created to build the relationship between the input seismic attribute values from the seismic survey and petrophysical properties from well logs. Four different cases with different types of seismic attributes are created to compare the impact of each seismic attribute on the model performance.

The results show that a deep learning neural network model with multi-hidden layers can be used to predict porosity values using extracted seismic attribute values from 3D seismic volumes. The model has higher accuracy in predicting porosity values with more seismic attribute values being added.

In future studies, more attributes can be added to evaluate their impacts on the predictive accuracy. Different combinations of seismic attributes may be tested to understand the impact of each seismic attribute on the model performance. Different types of convolution methods may be explored to look for the best number of vertical points in the seismic trace that can be compiled to corelate the well log value. For instance, five-point, seven-point, or fifteen-point convolution methods can be explored to see which method shows the best performance in predicting porosity values.

Chapter 4. Enhanced Automatic Segmentation of Salt Bodies from Seismic Images Using Wavelet Convolutional Neural Networks

Seismic images provide important information about the subsurface geological features. After a seismic survey, the raw seismic data can be processed and displayed as seismic images, which can be used by geologists to make interpretations about subsurface features. Important subsurface features such as salt bodies, formation boundaries, and faults can be interpreted by geologists. However, manual interpretation takes significant amount of human efforts and time to interpret large volumes of 3D seismic surveys. Manual interpretation may also bring subjective bias to the interpreted results.

This chapter explores the method in automatically identifying salt bodies from seismic images using a novel convolution neural network (CNN) combined with wavelet transformation analyses. Traditional CNN models use max pooling or mean pooling as the pooling layers. However, there exists limitations for these two pooling methods, which may result in loss of details of the original input.

This study combines wavelet transformation with CNN and applies it for the task of identifying salt bodies from seismic images. Adding wavelet analysis to the CNN model is expected to increase the model performance by using information from both the low-pass filters and the high-pass filters to the CNN model. The results show that the wavelet convolution neural network (Wavelet CNN) model has a better performance by incorporating wavelet transformation compared to traditional models. The model has a higher testing accuracy in identifying salt bodies. This novel technique of integrating wavelet analysis with conventional CNN, can be expanded for other geophysical interpretations to automatically identify other geologic features in the subsurface.

4.1 Introduction

Salt body is one of the common subsurface features. Figure 4.1 shows an illustration of a salt dome in the subsurface. There exist oil reservoirs around the salt body since salt provides a good sealing for hydrocarbons. Thus, accurate identification of salt bodies helps determine the location of oil reservoirs in the subsurface and is also significant in evaluating reservoir qualities.

Salt bodies identification from 2D or 3D seismic images is one of the major challenges in the oil and gas industry. Recent technologies have been used to improve the interpretation of salt bodies from seismic images (Jones and Davison, 2014). However, even with advanced seismic imaging methods, accurate determination for the location and property of salt bodies is still difficult. Significant amount of processing and interpretation are needed to precisely locate salt bodies in the subsurface.

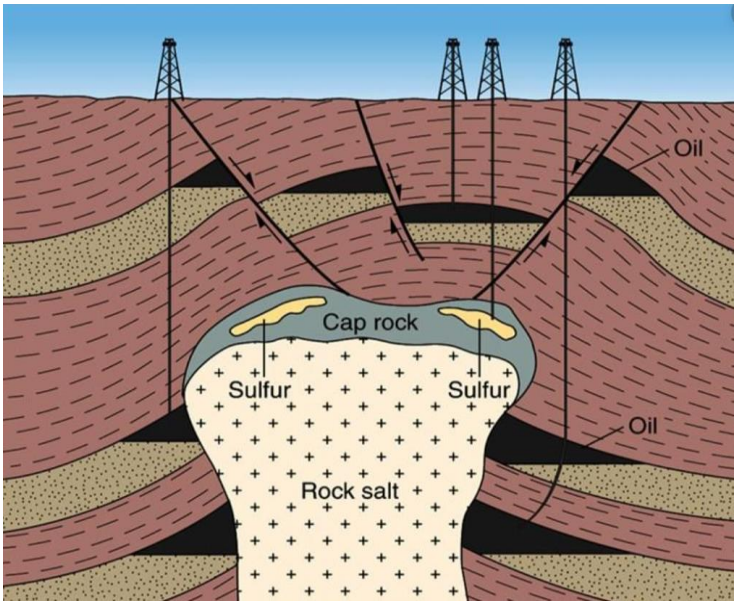

Figure 4.1. Illustration of a salt dome in the subsurface (Singh, 2012).

This study explores the method in establishing an automatic workflow for the segmentation of salt bodies from seismic images. A novel convolution neural network combined with wavelet

transformation analyses is used to predict locations of salt bodies. This is expected to increase the accuracy and efficiency for the automatic identification of salt bodies from seismic images.

4.2 Literature Review

Recently, development in data acquisition technologies in the oil and gas industry has created significant amount of subsurface data including seismic, well logs, and petrophysical data in order to have a better understand about the subsurface (Wang et al., 2012; Ildstad and Bormann, 2017; Zhao et al., 2018, Huang and Chen, 2019; Wang et al., 2018, 2019). Data-driven modeling methods have been used and proven to be effective in modeling complex nonlinear relationships using these different of data types (Udegbe et al., 2018, Zhang et al., 2018, Zhou et al., 2017). Meanwhile, research has been conducted to use automatic and semi-automatic workflows to process seismic data, and to explore the relationships between seismic data and other data types in the subsurface. These helps increase the speed of seismic interpretation and reduce the human bias from manual interpretation of seismic data (Zhu, 2005; Wu, 2016; Xie et al, 2017; Wrona et al., 2018; Yu et al., 2018). Besides, existing research also have been conducted to increase the efficiency and effectiveness in identifying salt bodies from seismic images (Pitas and Kotropoulos, 1992; Waldeland et al., 2018; Wang et al., 2018).

With the development of deep learning techniques, numerous methods have been developed for both supervised and unsupervised seismic data processing using deep learning neural networks (Hegazy and AlRegib, 2014; Wang et al., 2018; Roy et al., 2018). The development of convolutional neural networks (CNN) has brought impressive progress in various fields, and there exist studies that explore the use of CNN in seismic imaging (Waldeland et al., 2018; Karchevskiy et al., 2018).

Although CNN has shown improved accuracy and efficiency in seismic image recognition and segmentation (Waldeland et al., 2018; Karchevskiy et al., 2018), the relative small amount of public interpreted seismic data affects the predictive performance of the current deep learning CNN models since CNN prefers relatively large amount of input image data to train the model. The complexity of subsurface geologic features also requires the trained model to have enough capability of identifying geophysical structures from various subsurface features in seismic images. Thus, a robust model is needed to be able to identify subsurface features from the complex background in the subsurface.

In this study, a novel deep learning CNN-based method that combines wavelet transformation with a traditional CNN is proposed. Existing studies have explored the application of wavelet transformation in texture identification and classification (Fujieda et al., 2017; William 2017; Fujieda et al., 2018; Liu et al., 2019). However, its application for seismic image interpretation hasn't been tested, and therefore is the objective of this study, which is to explore the method in automatically identifying salt bodies from seismic images using a novel convolution neural network combined with wavelet transformation analyses.

4.3 Methodology

This study presents a novel deep learning approach to automatically identify salt bodies directly from seismic images. A wavelet convolutional neural network (Wavelet CNN) model, which combines wavelet transformation analyses with a traditional convolutional neural network, is developed and demonstrated to increase the accuracy in predicting salt bodies from seismic images.

4.3.1 Convolution Neural Network

A convolution neural network model is used to predict seismic facies from seismic images. A convolution neural network is a type of deep learning neural network. It has been widely used in image recognition such as digit and human facial recognition, texture classification, and image segmentation.

Within each convolution neural network model, there are different types of hidden layers, which consists of convolution layers, pooling layers and fully connected layers. As showing in figure 4.2, the input is a pixel image. The value for each pixel goes through the convolution layers, max pooling layers, second convolution layers, second max pooling layers and fully connected layers. The benefit of convolution neural network is its efficiency and effectiveness in analyzing images.

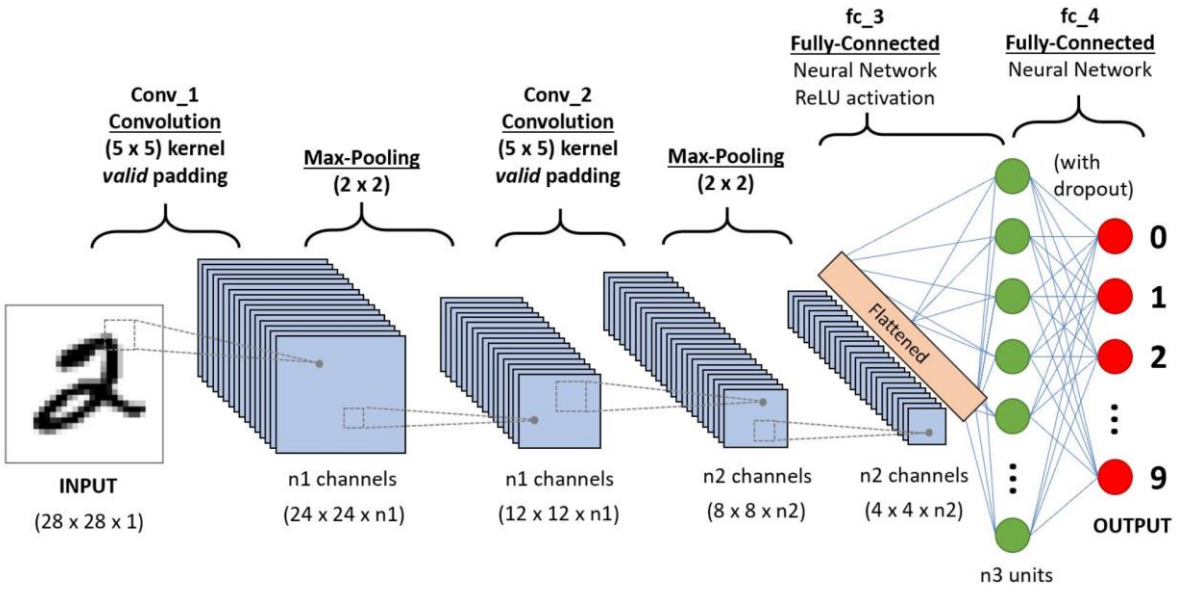

Figure 4.2. Illustration of the structure of a convolution neural network. The input is a pixel image. The output is predicted numbers (Saha, 2018).

Figure 4.3 shows the illustration of convolution and pooling layers in CNN (Fujieda et al., 2018). The convolution layer calculates a weighted sum of values from the previous layer. The

element that is used to perform the convolutional operation is called the kernel or filter. The model is not limited to only one convolution layer and can include multiple convolution layers for modeling complex problems.

The convolution operation can be expressed using the equation below (Fujieda et al., 2018). In the equation, y_i represents the output of the convolution layer, and x_j represents the input to the convolution layer, ω_j represents the weights.

$$y_i = \sum_{j \in N_i} \omega_j x_j$$

The above equation can be rewritten as the following equation using the convolution operator *, the input x, output y, and weights w (Fujieda et al., 2018).

$$\mathbf{y} = \mathbf{x} * \mathbf{w}$$

The function of the pooling layer is to decrease the dimension size of the previously convolved feature. As shown in figure 4.3, the output of the pooling layer has a reduced size comparing to its previous layer (Fujieda et al., 2018). This helps reduce the computational power that is needed to process the input data by reducing the dimension of the data from the previous layer.

Besides reducing the spatial size of the previous layer, the pooling layer is also effective in extracting information from important features that are positional or rotational invariant (Saha, 2018). This helps increase the effectiveness and efficiency of the model training process.

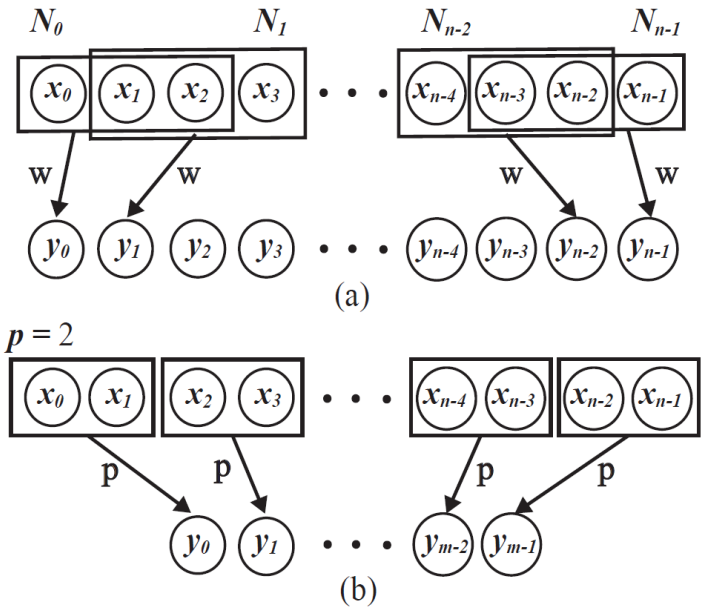

Figure 4.3. Illustration of convolution and pooling layers in CNN. (a) Convolution layers compute a weighted sum of the values from the previous layer. (b) Pooling layers reduce the size of the previous layer and perform down sampling (Fujieda et al., 2018).

The function of pooling layer can be represented by the below equation (Fujieda et al., 2018). In the equation, y_i represents the output of the pooling layer, and x_{pj+k} represents the input to the pooling layer, p represents the support of pooling.

$$y_i = \frac{1}{p} \sum_{k=0}^{p-1} x_{pj+k}$$

The above equation can also be rewritten by the following equation using the convolution operator *, input x, output y, and the averaging filter p (Fujieda et al., 2018).

$$\mathbf{y} = (\mathbf{x} * \mathbf{p}) \downarrow \mathrm{p}$$

Combining k and p into a generalized form of convolution network, the convolution and pooling processing can be represented by the following equation using the convolution operator *, input x, output y, and the averaging filter p (Fujieda et al., 2018).

$$\mathbf{y} = (\mathbf{x} * \mathbf{k}) \downarrow \mathrm{p}$$

The generalized weight **k** is defined as

k = **w** with p = 1 (convolution layer)

k = **p** with p > 1 (pooling layer)

k = **w*p** with p > 1 (convolution followed by pooling).

There are two types of commonly used pooling layers in a convolution neural network: maximum pooling and average (mean) pooling. As shown in figure 4.4, a maximum pooling layer outputs the maximum value from all the pixels that are covered by the kernel from the input image. In contrast, an average pooling layer outputs the average value of all the pixel values that are covered by the kernel from the input image.

Besides, the maximum pooling layer can serve as a noise suppressant during model training process (Saha, 2018). By taking the maximum values in the areas covered by the kernel, it reduces the noisy activations and helps denoising during the pooling operations.

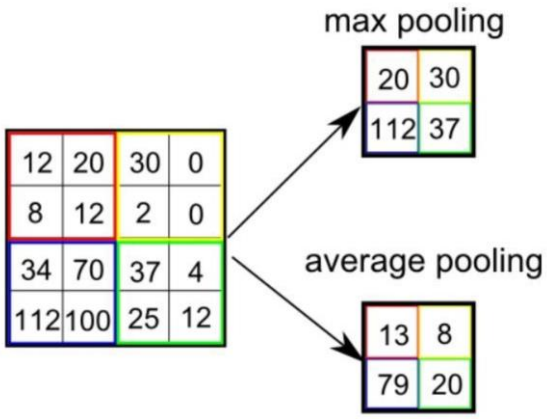

Figure 4.4. Illustration of maximum and average pooling layers in a convolution neural network model (Saha, 2018).

4.3.2 U-Net

U-Net is one type of convolutional neural network that was originally developed for biomedical image segmentation (Ronneberger, 2015). Its main idea is to add up-sampling layers to traditional down-sampling layers in a conventional convolutional neural network. The objective

is to increase the resolution of the output so that the output can have the same dimension with the input.

Figure 4.5 shows an illustration of a U-Net model. It consists of a down-sampling path (on the left side) and an up-sampling path (on the right side). The down-sampling path is similar as the typical structure of a traditional conventional neural network model, which contains different levels of convolution layers and pooling layers. At each level, there are two convolution layers and one pooling layer to reduce the dimension from the previous layers.

In the up-sampling path, the model contains up-convolution layers that are used to increase the dimension of the output layer. After running the same levels of up-sampling processes, the final output of the model has the same dimension as the input layer. Totally, there are 23 convolutional layers in the U-Net model.

As shown in figure 4.5, the input of the model can be a pixel image, the output of the model is a segmented image with the same dimension. The model trains from the input image and pre-segmented masks and can make pixel-wised predictions on new input images. This is useful and significant to perform image segmentation tasks.

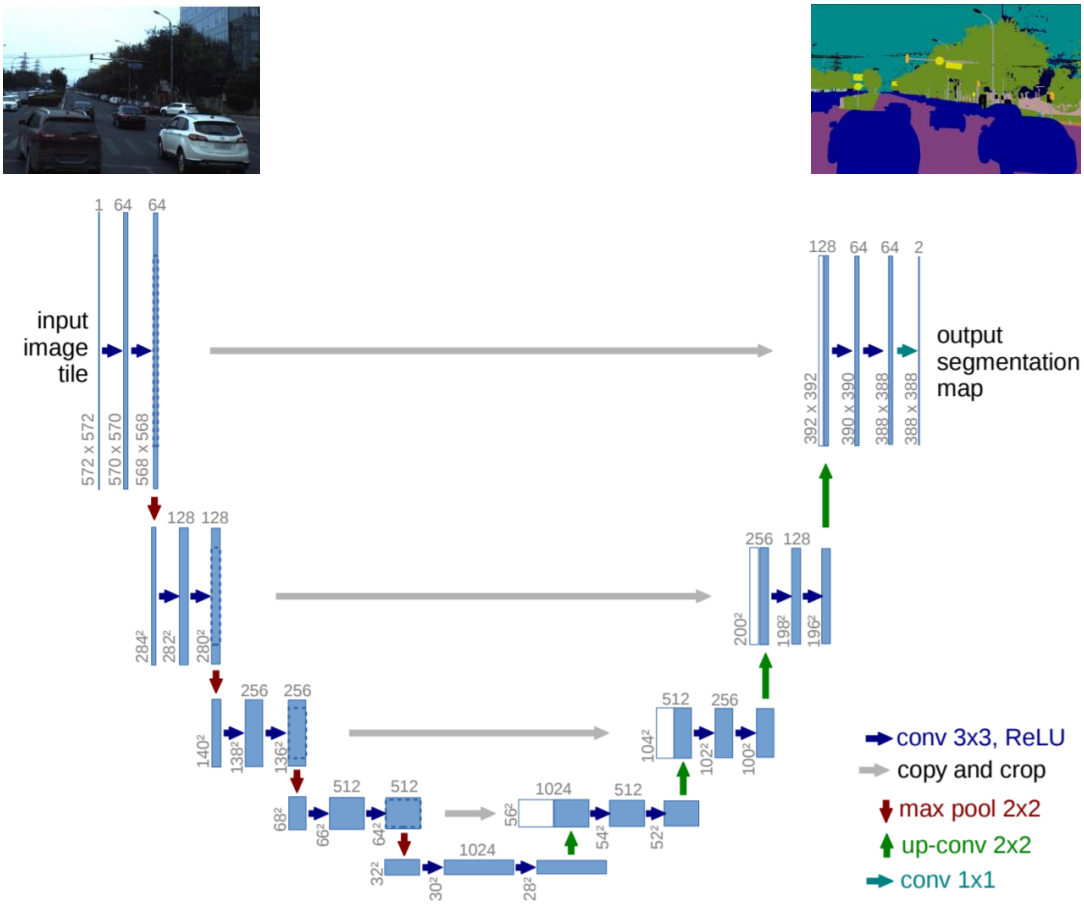

Figure 4.5. Illustration of a U-Net model (Ronneberger, 2015). The input of the model can be a pixel image, the output of the model is a segmented mask with the same dimension.

4.3.3 Wavelet Convolution Neural Network

This study introduces a new Wavelet CNN model that combines wavelet transformation with CNN and applies it in identifying salt bodies from seismic images. Traditional CNN model uses max pooling or mean pooling as the pooling layers. However, there exist several limitations for these two pooling methods (William and Li, 2018). For instance, figure 4.6 shows an original pixel image of a linear feature. After mean pooling, the intensity of the black line is reduced. After max pooling, it is even filtered out. These two traditional pooling methods may result in loss of details of the original image.

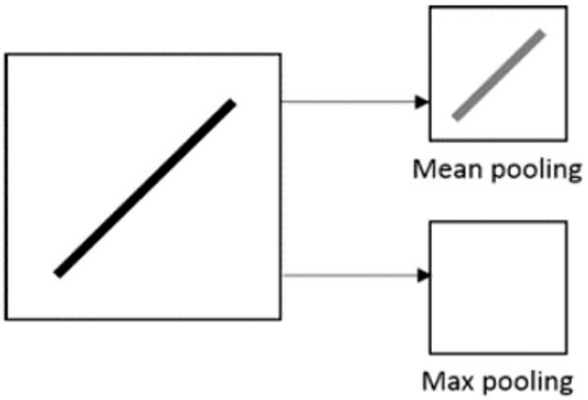

Figure 4.6. Limitations of mean pooling and max pooling for conventional CNN. Traditional max pooling and mean pooling may result in loss of details of the original image (William and Li, 2018).

Instead of using max pooling or mean pooling, this study utilizes Harr wavelet transformation as the pooling method in order to reduce the size of input. Previous studies have utilized Harr wavelet transformation for image compression to reduce image size (Lo et al., 2003, Qureshi and Deriche, 2016). Harr wavelet transformation is a type of discrete wavelet transformation, which can be shown as follows.

$$W_{\psi}[j+1,k] = h_{\psi}[-n] * W_{\varphi}[j.n]\,|_{n=2k,k\leq 0}$$

$$W_{\varphi}[j+1,k] = h_{\varphi}[-n] * W_{\varphi}[j.n]\,|_{n=2k,k\leq 0}$$

The Harr wavelet's function can be represented as the follows:

$$\psi(t) = \begin{cases} 1 & 0 \leq t < \frac{1}{2} \\ -1 & \frac{1}{2} \leq t < 1 \\ 0 & otherwise \end{cases}$$

The Harr wavelet's scaling function can be described as follows:

$$\varphi(t) = \begin{cases} 1 & 0 \leq t < 1 \\ 0 & otherwise \end{cases}$$

Figure 4.7 and figure 4.8 shows an illustration of the first level wavelet transformation. There are four components for the output of the first level wavelet transformation, which contains the

approximation, horizontal, vertical, and diagonal components. The approximation component is similar to the original image and is also called the “a” component or LL component, in which L represents for the low-pass filter. The other three components show edge features of the original image. The diagonal component is also called the “d” component or HH component, in which H represents for the high-pass filter. The horizontal component is also called the “h” or the LH component. The vertical component is also called the “v” or HL component.

In figure 4.8, the vertical component effectively saves the vertical features such as the vertical edges of the tall building in the right part of the image. The horizontal component effectively saves the horizontal features such as the horizontal edges of the houses on the right part of the image. These three components effectively save edge features of the input image and may be used for feature extraction applications.

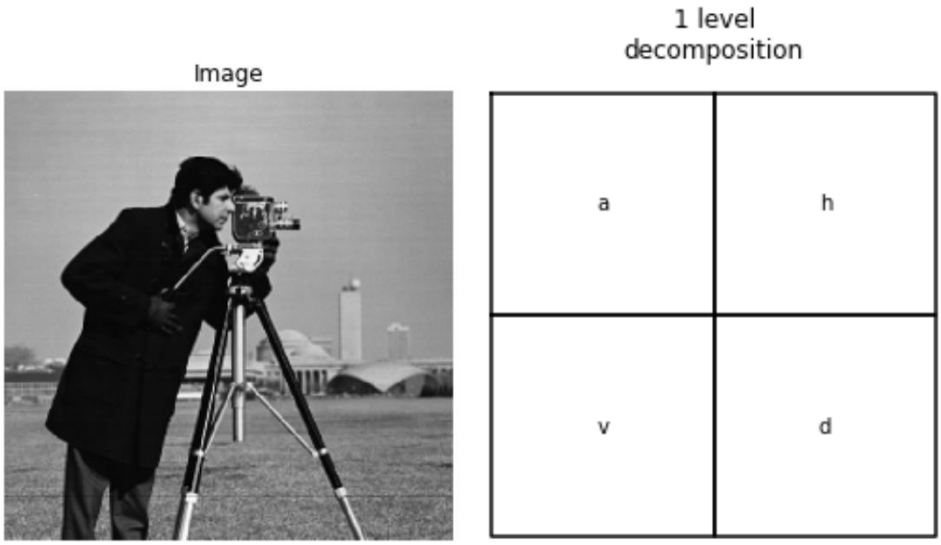

Figure 4.7. Illustration of the first level wavelet transformation. There are four components for the output of the first level wavelet transformation, which contains the approximation, horizontal, vertical, and diagonal components.

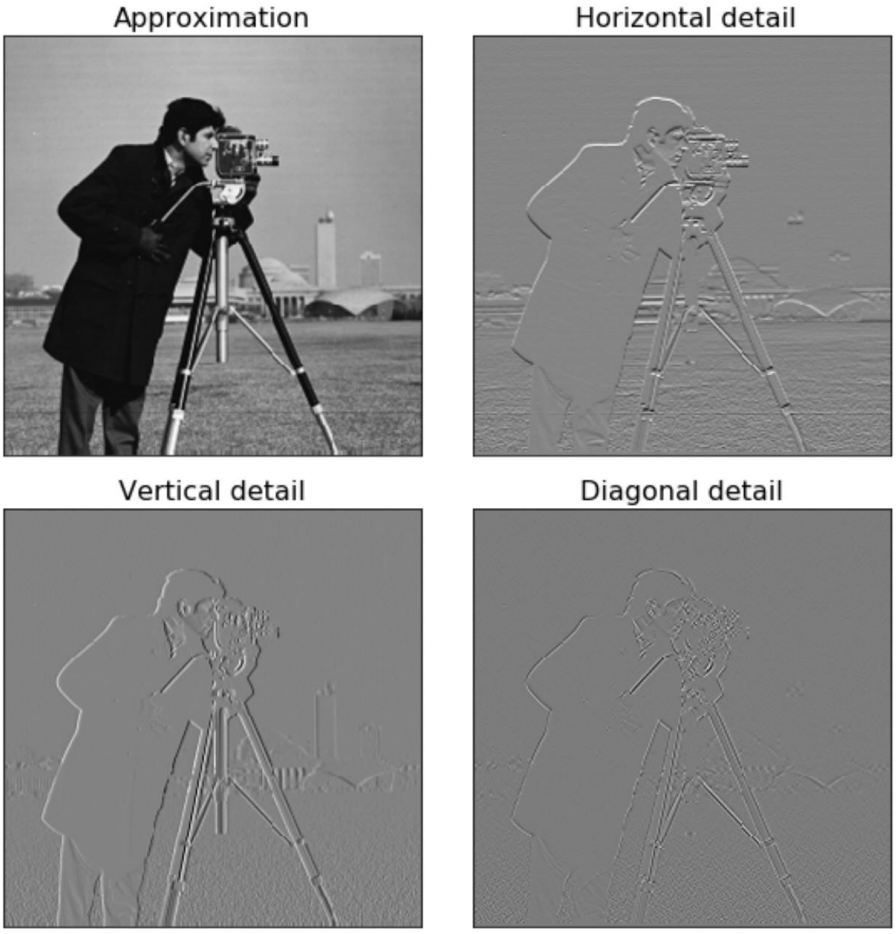

Figure 4.8. Illustration of four components after the first level wavelet transformation. The approximation component is similar to the original image. The other three components show edge features of the original image.

Figure 4.9 shows an example for the outputs of three levels of wavelet transformation. At each level, the wavelet transformation produces four sets of coefficients corresponding to the four possible combinations of the wavelet transformation filters on the two separate axes. For subsequent levels of transformation, only the approximation coefficients (the lowpass sub-band) are further decomposed.

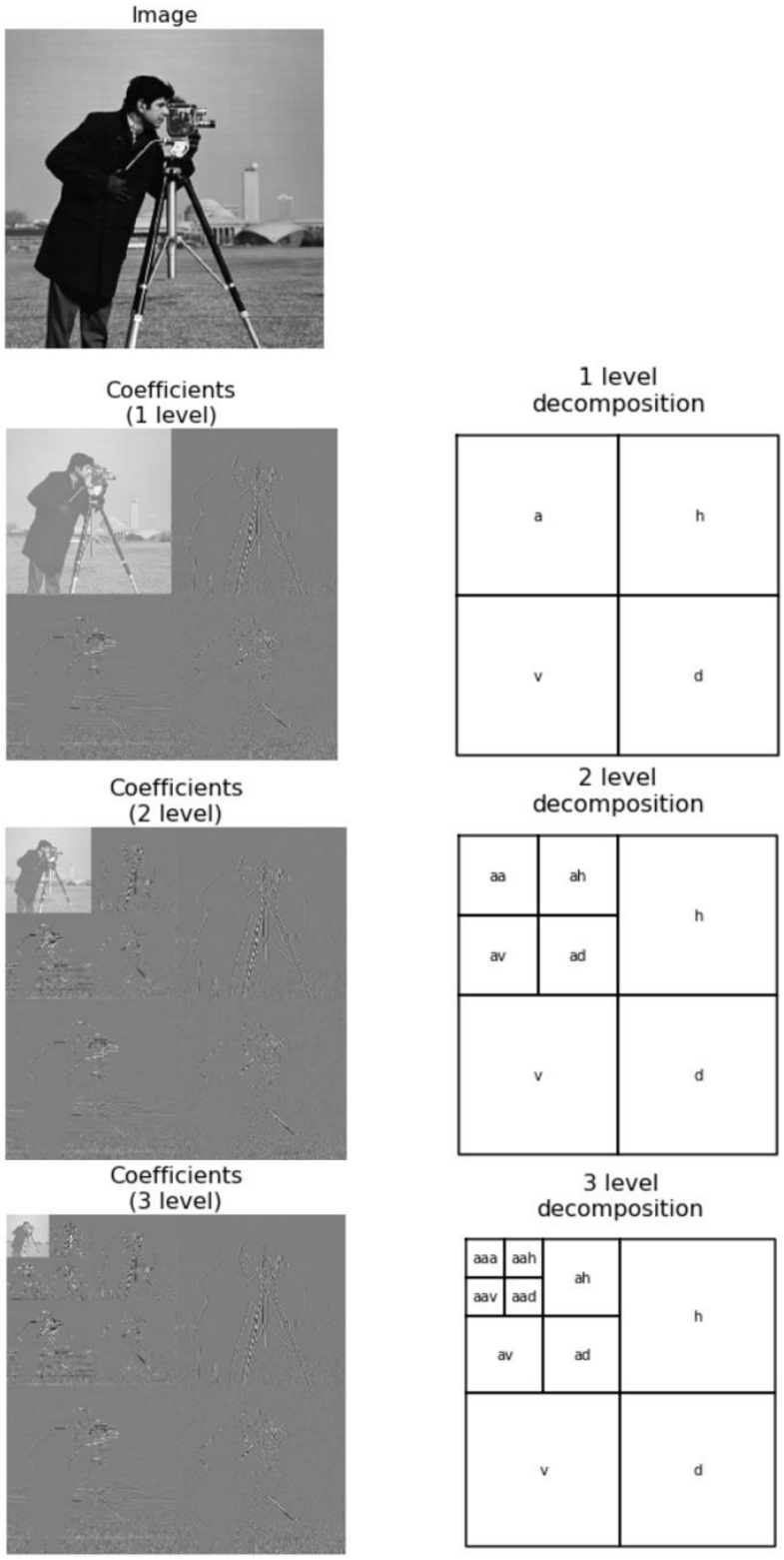

Figure 4.9. Illustration of wavelet transformation for three levels. For subsequent levels of transformation, only the approximation coefficients (the lowpass sub-band) are further decomposed.

Figure 4.10 shows another example of a line feature after the first level wavelet transformation. The vertical component in the lower left part effectively saves vertical edges of the line feature. Figure 4.11 shows an additional example of three-level wavelet transformation on several linear features. It is noticed that the vertical component, horizontal component, and diagonal component effectively capture feature edges of the input image. This indicates that these three high-pass filter components of wavelet transformed outputs may be useful for edge identification.

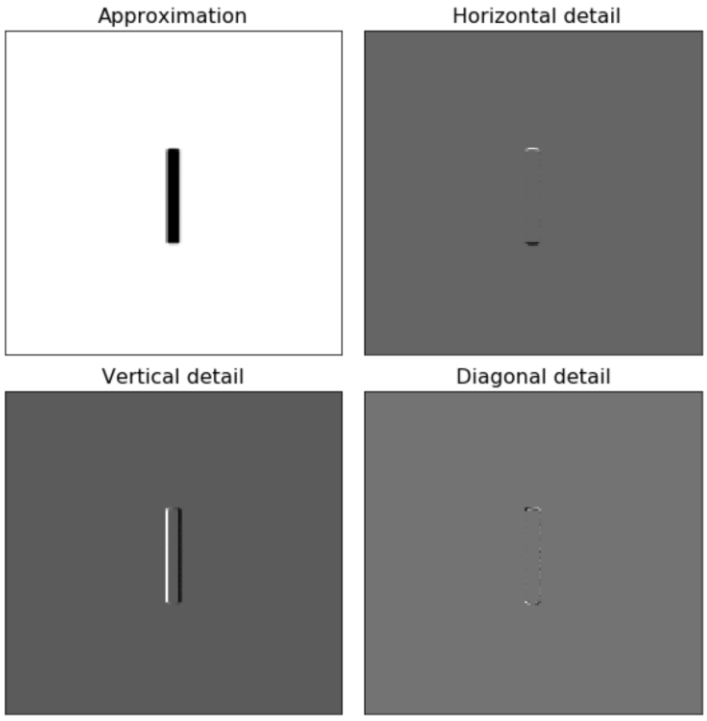

Figure 4.10. Illustration of a vertical line feature after first level wavelet transformation. The vertical component in the lower left part effectively saves vertical edges of the line feature.

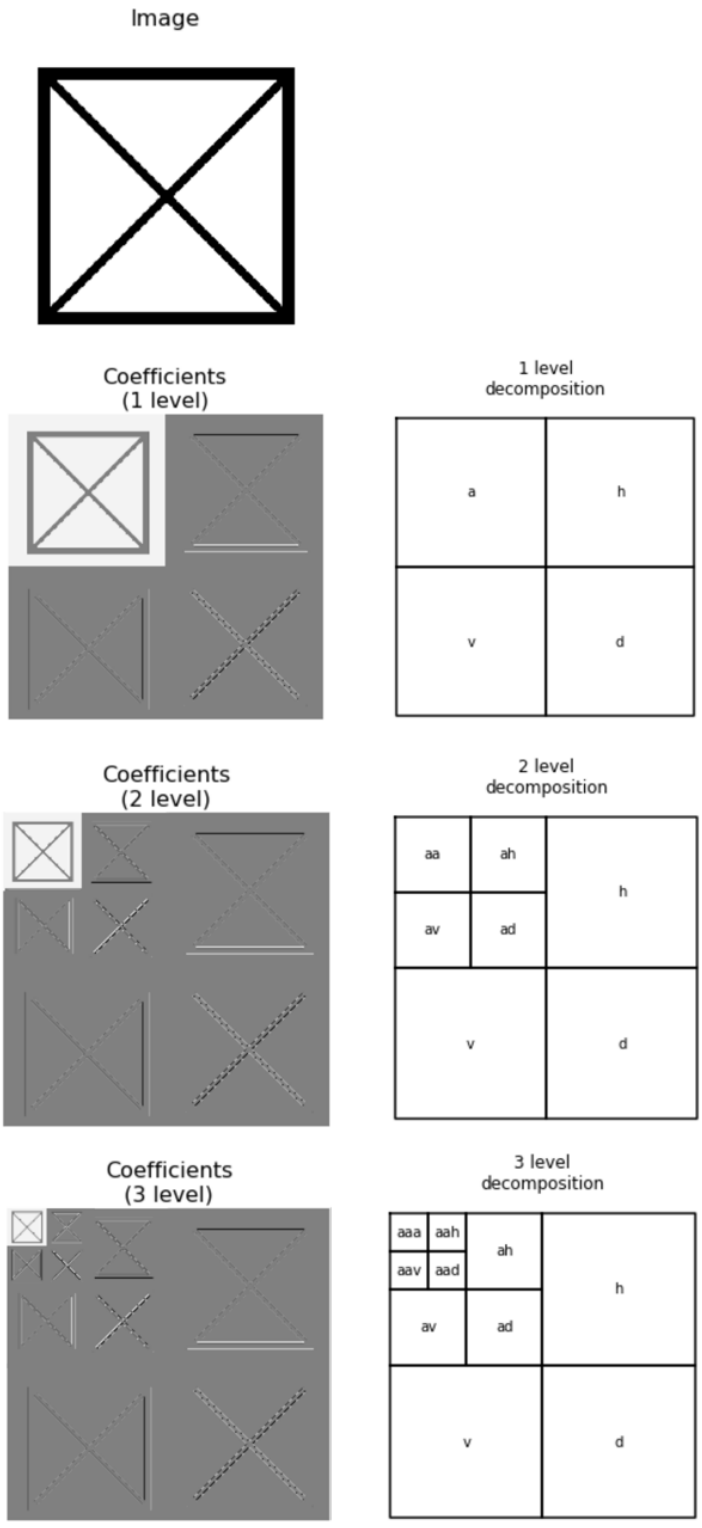

Figure 4.11. Illustration of linear features after three levels of wavelet transformation. The vertical component, horizontal component, and diagonal component effectively capture feature edges of the input image.

Existing research has shown that traditional CNN models that use max pooling or mean pooling may result in loss of details of the original image (William and Li, 2018). To address these limitations of conventional CNN models, this study utilizes wavelet transformation as the pooling method instead of using max pooling or mean pooling. Figure 4.12 shows the major difference between a conventional CNN and Wavelet CNN (Fujieda et al., 2017). The red arrows indicate the pooling layers for a conventional CNN, which only uses low-pass filters. This is similar to only taking the low-pass filtered information (approximation component) from the output of wavelet transformation and thus may lose important information from the high-pass filters, including the horizontal, vertical, and diagonal components.

The Wavelet CNN model uses both the high-pass filter and the low-pass filter to construct the pooling layers (Fujieda et al., 2017). In other words, both the approximation component (low-pass filter), and the rest horizontal, vertical and diagonal components (high-pass filters) are used to save filtered information in the model. This is expected to help save important edge information during pooling operations.

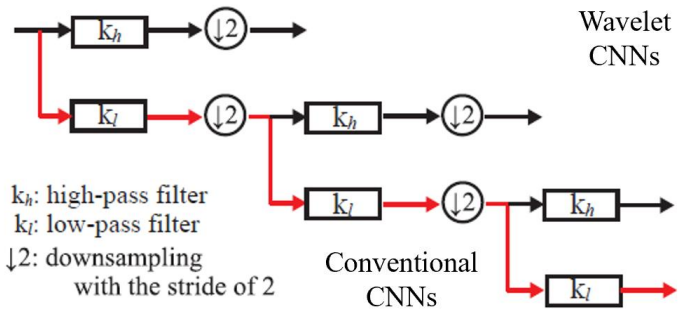

Figure 4.12. Illustration of the high-pass and low-pass filters in CNNs (Fujieda et al., 2017). The red arrows indicate the pooling layers for a conventional CNN, which only uses low-pass filters. The Wavelet CNN model uses both the high-pass filter and the low-pass filter.

Figure 4.13 shows the structure of the Wavelet CNN model. A U-Net model is utilized as the base model. There are four levels of wavelet transformation that are performed to reduce the

dimension of the output from the previous layer and save the important features from both high-pass and low-pass filters.

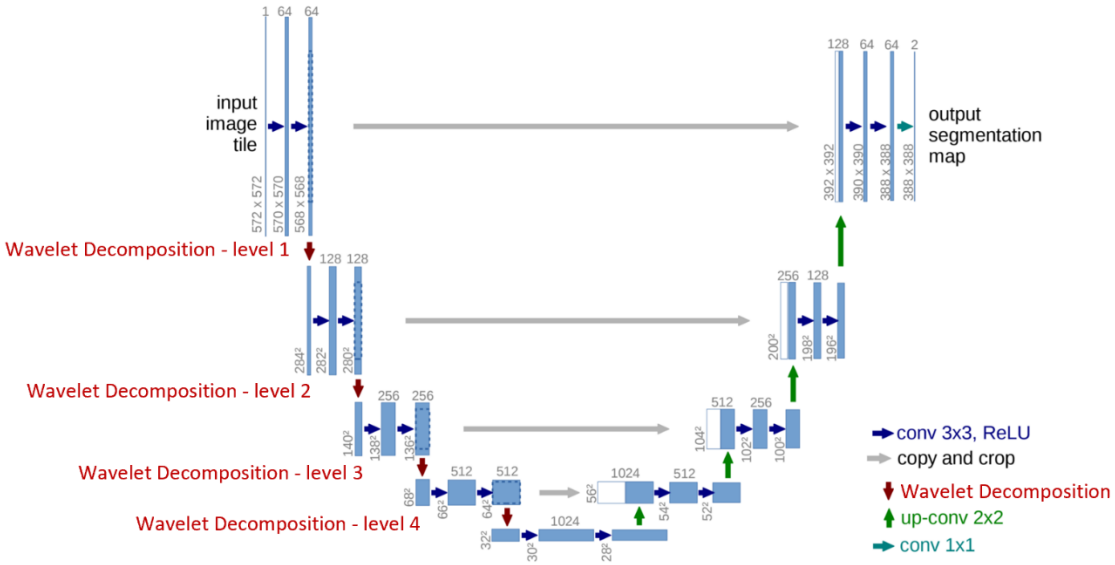

Figure 4.13. Structure of the Wavelet CNN model. A U-Net model is utilized as the base model. There are four levels of wavelet transformation that are performed to reduce the dimension of the output from the previous layer. Image is modified from Ronneberger, 2015.

4.4 Dataset

The dataset of the study was collected by TGS-NOPEC Geophysical Company (TGS) and was provided in a competition organized by TGS and Kaggle. The dataset consists of 2D seismic image slices, which are displayed as grayscale images showing the subsurface features including formations, structures, and facies. The images provided in the dataset contain randomly cropped 101×101 small size seismic images from the original seismic survey. The goal is to create an automatic workflow, which trains from given seismic images and masks, and can be used to predict locations of salt bodies from seismic images.

The dataset contains 4000 images and masks. The inputs of the model are 2D seismic images. The outputs are pre-interpreted salt body masks. Figure 4.14 shows two sample input and output

images in the dataset. In each mask, white color represents pre-interpreted salt bodies, and black color represents non-salt areas.

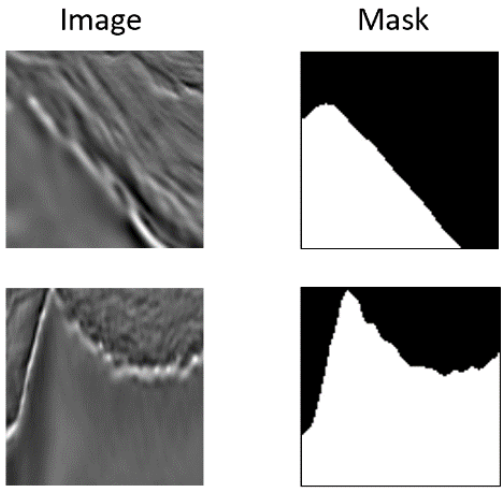

Figure 4.14. Input images and output masks of the model. The inputs are 2D seismic images. The outputs are masks with pre-interpreted salt bodies.

4.5 Results

Figure 4.15 shows an illustration of a sample original seismic image in the left, as well as its wavelet transformed image on the right (approximation component). The size of the approximation component is reduced in half.

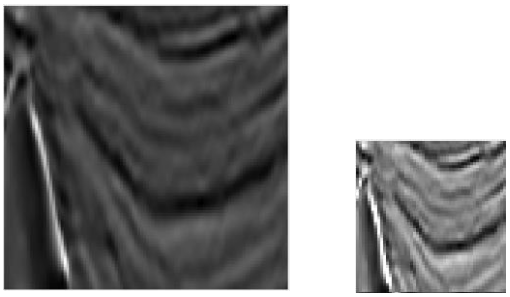

Figure 4.15. Illustration of the original and wavelet transformed seismic image.

We compared a total of four cases to evaluate the effect of combining wavelet transformation analyses with CNN. First three cases are to test whether a direct wavelet transformation on the original input images affect the model performance. For case 1, an original U-Net model is trained on the original given images form the dataset. For case 2, an original U- Net model is trained only

on the wavelet transformed images (approximation component). For case 3, an original U-Net model is trained on the combination of the original and the wavelet transformed images (approximation component).

The dataset is split into a training set and a testing set, each with 80% and 20% of total samples. Figure 4.16 shows the testing accuracies for the first three cases. Case 1 has the highest accuracy to be 0.939 and case 2 has the lowest accuracy to be 0.929. The accuracy for case 3 is between case 1 and case 2 as 0.934.

Since the accuracy for the original U- Net trained on the approximation component of wavelet transformed images (case 2) is not improved comparing to case 1 with the same model trained on the original images, it indicates that simply changing the training images to be approximation component of wavelet transformed images does not improve the model performance. Adding the approximation component of wavelet transformed images to the dataset (case 3) also does not increase the test accuracy.

We interpret the reason to be that the wavelet transformed images in case 2 and 3 are approximations of the original images, meaning that there is no new information added to the model only by training the model with added approximation component of wavelet transformed images. Thus, training a traditional U-Net model using the approximation component of wavelet transformed images does not improve the model performance.

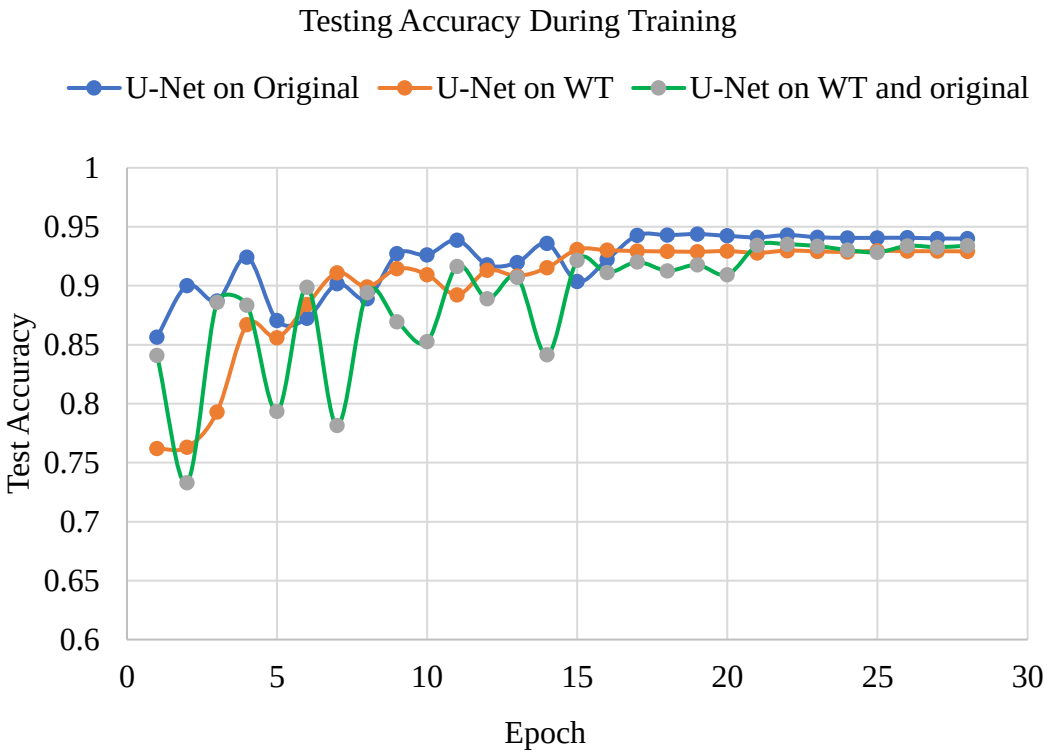

Figure 4.16. Plot of the testing accuracy during training for case 1, 2, and 3.

For case 4, the model performance was evaluated on four different sub-cases. Each sub-case uses a different pooling method including max pooling, average pooling, mixed pooling and wavelet pooling. Mix pooling was performed by taking two levels of max pooling, and additional two levels of average pooling.

Figure 4.17 shows the comparison of the testing accuracy for the four sub-cases. Figure 4.18 shows the training accuracy at each epoch for sub-case 1 (with max pooling) and sub-case 4 (the Wavelet CNN model). The testing accuracy for the original U-Net with max pooling (sub-case 1) is 0.939 after training for 28 epochs. The accuracies for mean (sub-case 2) and mixed (sub-case 3) pooling are 0.941 and 0.942 respectively. The Wavelet CNN model (sub-case 4) shows 0.949 testing accuracy after 35 epochs, which performs the best among all four models.

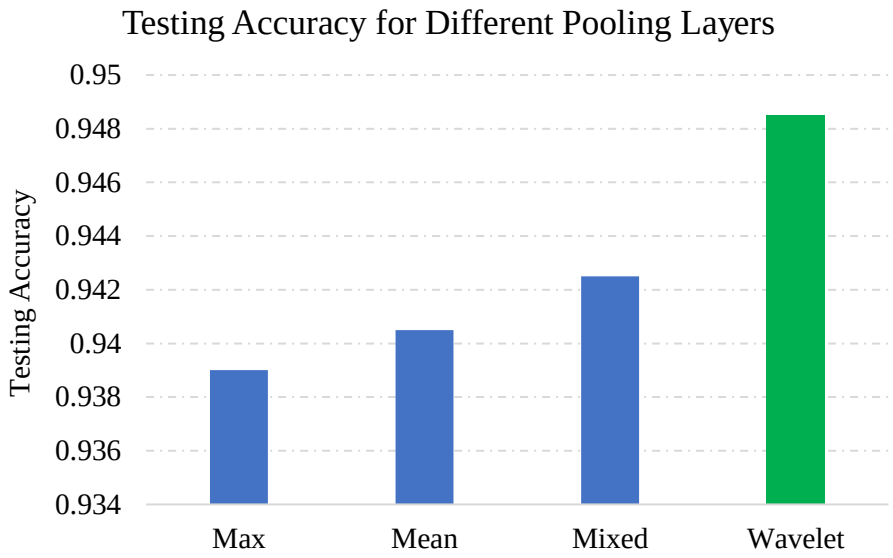

Figure 4.17. Comparison of the testing accuracy for the U-Net model with max pooling, mean pooling, mixed pooling and the Wavelet CNN model. The Wavelet CNN model has the highest testing accuracy among the four models.

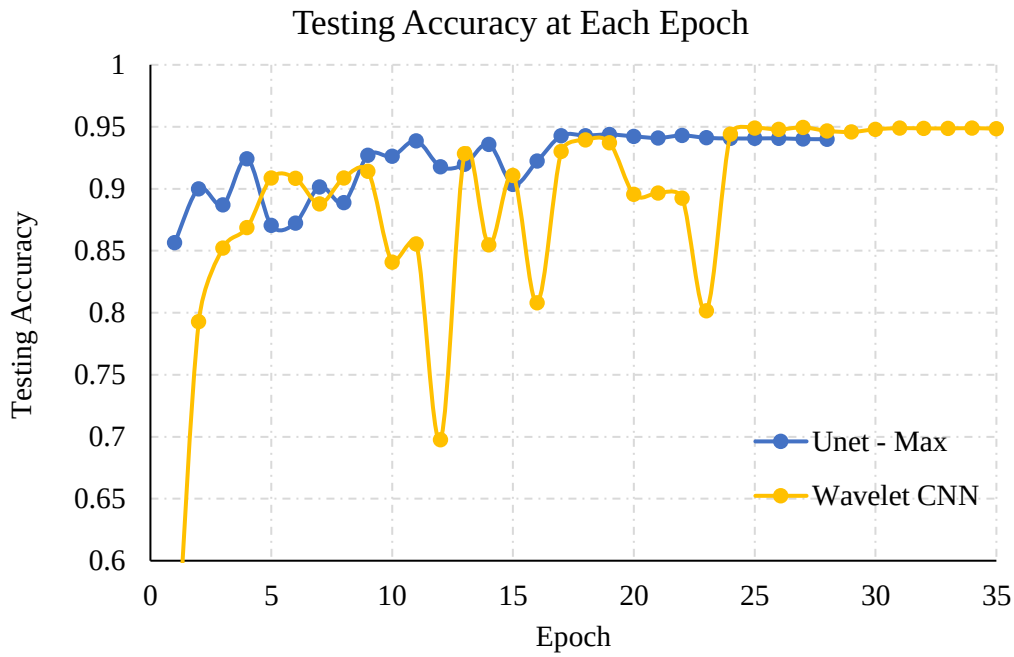

Figure 4.18. Comparison of the testing accuracy for the U-Net model with max pooling, mean pooling, mixed pooling and the Wavelet CNN model. The Wavelet CNN model shows 0.949 testing accuracy after 35 epochs, which has the highest testing accuracy.

Figure 4.19 shows predicted results from the model. The left column shows the original input seismic image. The black line represents the pre-interpreted salt boundaries. The second column

shows the masks. The third and fourth column display the predicted boundaries of salt bodies. They show that the trained model is capable to identify the boundaries of salt bodies from seismic images.

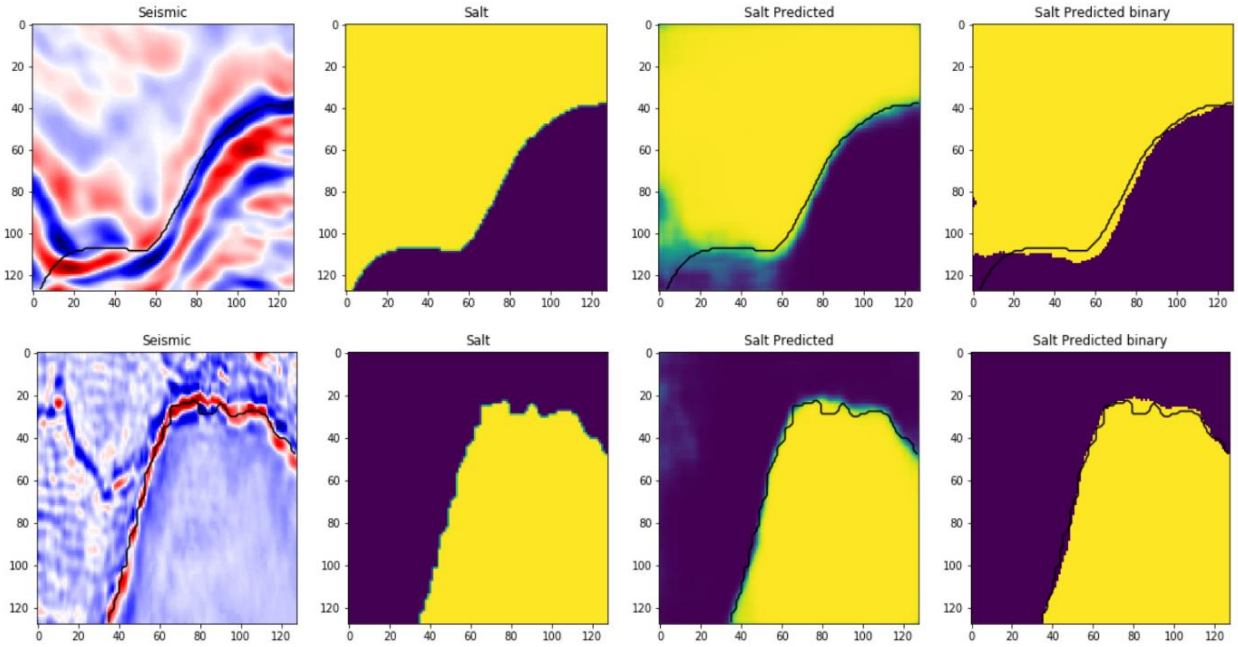

Figure 4.19. Predicted salt bodies from the trained model. The left column shows the original input seismic image. The second column shows the mask. The third and fourth columns show the predicted boundaries of salt bodies.

4.6 Discussion

This study used Harr wavelet to perform a 2D wavelet transformation on the input images. Harr wavelet is a common type of discrete wavelet transformation method that can be used for image compression and edge detection. Harr wavelet transformation is computationally efficient since it only involves linear transformation during the process. Besides Harr wavelet, other wavelet transformation methods may also be tested in the future work to perform the pooling operation in the CNN model.

The model was trained from seismic images and masks with pre-interpreted salt bodies. The masks contain only black and white colors (with values of zero and one) representing two classes

as salt body and non-salt. Besides using the Wavelet CNN model for salt bodies segmentation, the model can also be used for predicting other subsurface seismic facies using seismic images as the input.

If a new model needs to be created for the sake of segmenting more seismic facies other than salt, masks with more facies needs to be interpreted from the seismic images. The model that is trained from the input seismic images and output masks with more labels can be used to predict segmented seismic facies maps for more numbers of facies.

4.7 Conclusion

This study combines wavelet transformation with CNN and applies it for the task of identifying salt bodies from seismic images. The Wavelet CNN model uses all four components from the output of wavelet transformation to construct the pooling layers. It saves important information from the high-pass and low-pass filters during pooling operations.

The results show that the Wavelet CNN model demonstrates improved performance as compared to conventional methods that use max and mean pooling. The Wavelet CNN model shows 0.949 testing accuracy after 35 epochs, which performs better than the other models. By utilizing multi-levels wavelet transformation as the pooling layers, the input image size is reduced with the key edge features being preserved, resulting in improved prediction accuracy in segmenting salt bodies from seismic images.

In the future work, this novel technique of integrating wavelet transformation with conventional CNN, can be expanded to other study areas to validate its application in different scenarios. Datasets with more interpreted subsurface features can be added to train the Wavelet CNN model that can identify more geologic features in the subsurface. Besides, different wavelet

transformation methods, and CNN structures with different combination of pooling layers can be tested to explore their impacts on the model performance in automatic segmentation of salt bodies.

Chapter 5. Evaluation of Singular Value Decomposition (SVD) on Dimension Reduction of Permeability Field for Reservoir Modeling

This chapter evaluates the effect of singular value decomposition (SVD) in dimension reduction of permeability field for reservoir modeling. A two-phase flow reservoir model was created using data from the SPE tenth comparative solution project. Simulation results show that SVD is valid in the parameterization of permeability values. The reconstructed permeability matrices using certain amount of singular values are good approximations of the original permeability values. Simulation results using SVD processed permeability field are similar to that using the original values.

SVD is then applied on the upscaled permeability values to evaluate the effectiveness on upscaling. Simulation results were compared between the base case, upscaled case, and SVD upscaled case. The simulation results did not show a significant improvement in the accuracy of predicting oil production by applying SVD on the upscaled permeability values. It could be because the reconstructed permeability matrix has the same size before and after the SVD processing, thus the model accuracy and efficiency are not significantly improved.

5.1 Introduction

Dimension reduction is an important step for reservoir modeling. It reduces the size of a reservoir model to save computational time. Parameterization of the petrophysical properties, which is a dimension reduction method, has been used to remove redundancy of the reservoir parameters and can assist history matching (Jafarpour and McLaughlin, 2008). It transforms the original parameter field into lower dimension and expedite history matching processes.

Upscaling, or homogenization, is to substitute a heterogeneous model that consists of high-resolution fine grid cells with a lower resolution reduced-dimensional homogeneous model using averaging schemes. The benefit of upscaling in reservoir simulation is that it saves simulation time,

and effectively preserves key features of data for flow simulations. The critical point in upscaling is the selection of an appropriate scheme to effectively represent properties for the fine grid cells (King, 2007, Preux, 2016).

Singular Vector Decomposition (SVD) is a matrix decomposition method. It has been used in various applications such as image compression and facial recognition. Figure 5.1 shows the application of SVD in image compression (Gibiansky, 2013). The original image is the full rank tiger in the upper left corner of figure 5.1. It can be represented by a matrix with the same size as the pixel dimension of the image.

After SVD, different numbers of singular values can be used to approximate the original matrix during the reconstruction process. In figure 5.1, the rest lower rank images use less amount of singular values to reduce the storage. It shows that, as the number of singular values decreases, the reconstructed images are less alike to the original image. But reconstructed images with less numbers of singular values take less storage space, and thus achieve the goal for image compression (Cao, 2006).

More specifically, the original size of the image in figure 5.1 is 500×800 pixels. It can be represented by a matrix with the dimension of 500×800. After SVD processing, it can be decomposed and represented by the product of three separate matrices, U, S, and V, with the dimension of 500×500, 500×500, 500×800. S is a diagonal matrix that contains all the singular values in its diagonal part. When performing SVD to reconstruction the original image, only a small portion of the singular values may be used to reduce the storage. The lower right figure is the rank-3 image, meaning that it only uses 3 singular values. Thus, the dimension of U, S, V matrices becomes 500×3, 3×3, and 3×800. The product of U, S, V still creates a 500×800 image,

but the amount of values that needs to be stored is significantly reduced. So, SVD has been proved to be valid for the purpose of image compression and storage reduction.

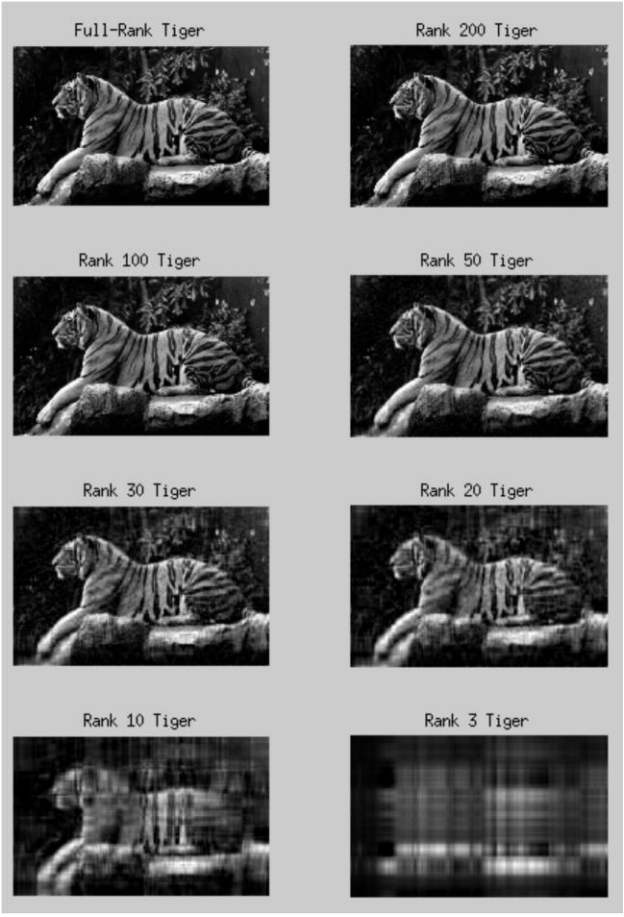

Figure 5.1. An example of the application of SVD in image processing. The top left figure represents the full-rank image. The rest lower rank images use less amount of singular values to reduce storage, while preserving key features from the full-rank image (Gibiansky, 2013).

SVD also has been used in noise attenuation. Figure 5.2 shows an illustration for a 25×15 image that has a noisy background. By performing SVD on the original image, the matrix that represents the original image can be decomposed and represented by the product of three separate matrices, U, S, and V with the dimension of 25×15, 15×15, 15×15. Instead of using all 15 singular values, only using three singular values in S can generate the reconstructed image shown in the right part of figure 5.2. The background of the improved image (right) has less noise comparing to the original image (left).

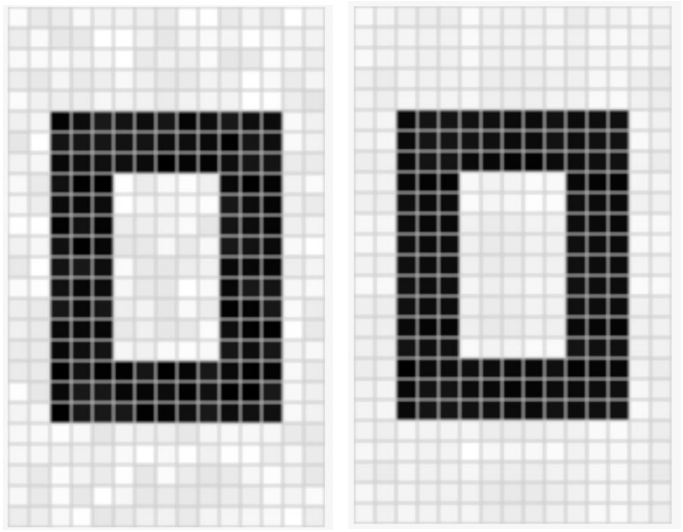

Figure 5.2. An example of the application of SVD in noise reduction. The left figure represents the original image with a noisy background. The right figure is the SVD processed image with only three singular values used. The background has less noise after SVD (Austin, 2009).

Existing research has used SVD to compress images. Thus, it is worthwhile to explore the use of SVD in reducing dimension of reservoir parameters. This chapter studies the application of SVD in dimension reduction of the permeability field for reservoir modeling. Both the effects of SVD in permeability parameterization and upscaling is analyzed.

5.2 Literature Review

During reservoir modeling, dimension reduction is significant to reduce the model size to improve the efficiency of reservoir simulation and history matching. Reduction of the model dimension can be achieved by parameterization of the petrophysical values. The output of the parameterized field is a representation of the original field with a lower dimension. Previous studies (Jafarpour and McLaughlin, 2008; Jafarpour, 2013) explored the application of an ensembled Kalman filter together with discrete cosine transformation in parameterization of permeability field for history matching. Their results show that parameterization of the permeability field is effective in eliminating redundancy during history matching and results in additional computational savings.

Besides, upscaling is an important step for dimension reduction during reservoir modeling. It converts highly detailed geological models to simulation grids with a lower dimension. Its objective is to use a coarse grid model to represent a fine grid model, thus to reduce simulation time. Christie (1997) summarized the techniques for upscaling, such as pressure-solver methods, and renormalization methods. He also pointed out that one of the main limitations of upscaling is that it usually has no indication of whether the upscaled value provides a good or bad approximation and whether the assumptions made in deriving the answer hold. Reservoir simulation needs to be performed in order to verify the effectiveness of upscaling.

Salazer and others (2007) evaluated the effectiveness of different permeability upscaling techniques including static and dynamic upscaling. The static upscaling includes arithmetic, harmonic and geometric averages. The dynamic upscaling processes can be categized into two different types; one-phase upscaling and two-phase upscaling. One-phase upscaling process only considers the upscaling of the distribution of the permeability values. On the contrary, two-phase upscaling process considers both the distribution of the permeability values and the relative permeability curve.

SVD has shown to be effective in image compression and facial recognition. Cao (2006) applied SVD to digital image processing. They tested different singular values and evaluated the compression results by compression ratio and quality measurement. Their results indicated that SVD has the advantage of providing a good compression ratio, and effectively saves digital image storage space. Among their tests, they found that some of images are quite simple so that only a small number of singular values are enough to obtain the approximation. However, some complex images need to use more values to maintain their quality.

SVD also has been used for signal denoising and enhancement. Bekara and Baan (2007) applied SVD processing to improve the signal-to-noise ratio for seismic data and their results showed positive signal enhancement by performing SVD on seismic data. Jha and Yadava (2011) explored the utilization of SVD in signal denoising for surface acoustic wave (SAW) sensors. The denoising process is performed by apply SVD on the original signal, and only using a few singular values to reconstruct the original signal. Their methods show that SVD is valid in removing noise from the sound sampling devices, as well as the delivery system.

5.3 Methodology

5.3.1 Singular Value Decomposition

This study evaluates the application of SVD in dimension reduction of permeability field for reservoir modeling. The process of SVD can be described as:

$$M = USV^T \tag{5.1}$$

where M is the original matrix, U is a unitary matrix after SVD whose columns are the vectors u_1 and u_2, S is a diagonal matrix whose entries are σ_1 and σ_2, which are the singular values, and V is a unitary matrix whose columns are v_1 and v_2.

Taken image compression as the example, an image can be represented by a matrix with the size of the pixel numbers of the image. After conducting SVD on the matrix of the image, the matrix can be represented by the sum of several matrices with the rank of one as follows:

$$A=\sigma_1\mu_1 v^T{}_1+\sigma_2\mu_2 v^T{}_2+\ldots+\sigma_r\mu_r v^T{}_r \tag{5.2}$$

After SVD, it is possible to determine how many numbers of singular values that are needed to approximate the original matrix. Different SVD schemes can be applied to reconstruct the original matrix. The reconstructed matrix and the original matrix can be compared to evaluate the effects of SVD.

5.3.2 Reservoir Simulation

After the reconstruction of the permeability values, reservoir simulation is performed using the reconstructed values to see whether the SVD performance is valid in dimension reduction. In this project, a two-phase flow IMPES simulator is used for reservoir simulation. It uses Mass conversation and Darcy's equation shown in Equation 5.3 and Equation 5.4 as follows (Chen, 2007).

$$\frac{\partial(\varphi\rho_\alpha S_\alpha)}{\partial t} = -\nabla\cdot(\rho_\alpha \boldsymbol{u}_\alpha + q_\alpha) \;\; \alpha = w, o. \tag{5.3}$$

$$\boldsymbol{u}_\alpha = -\frac{1}{\mu_\alpha}\boldsymbol{k}_\alpha(\nabla P_\alpha - \rho_\alpha g \nabla z) \;\; \alpha = w, o. \tag{5.4}$$

The simulator is based on block centered grid system using IMPES method. The IMPES pressure equation is shown in Equation 5.5, and saturation equation is shown in Equation 5.6 in the appendix A.

The pressure and saturation have spatial discretization forms that are shown in Equation 5.7 and Equation 5.8 in the appendix A. The pressure is solved fully implicitly using Gauss-elimination that is shown in in Equation 5.7. The saturation for two-phase reservoir simulator is solved explicitly by using Equation 5.8.

In this study, 2D reservoir simulation models were created and permeability values are imported from the SPE tenth comparative solution project. The results are presented in three parts. Part one is the application of SVD in parameterization of permeability field for an example problem that contains channel features. Part two is the application of SVD in parameterization of permeability values for the SPE tenth dataset. Part three is to evaluate the application of SVD in upscaling for the SPE tenth dataset.

In part three, three cases with different upscaling procedures are created in order to compare their results. Case 1 represents the base model, which has 20×10 grid blocks. In case 2,

permeability values are upscaled in the x direction, such that the number of grid blocks is reduced to 10×10. In case 3, permeability values are upscaled in both the x and the y direction, meaning that grid blocks are reduced to 10×5.

During upscaling, the permeability values of two neighboring blocks are averaged to get the new upscaled permeability value for the new upscaled block. After upscaling, SVD is performed for the upscaled permeability values via MATLAB. When performing SVD on the parameters, the matrix is decomposed and can be represented by the product of three separate matrices, U, S, and V. Different numbers of singular values are truncated in S for different levels of SVD processing. After SVD processing, these matrices are utilized for the reconstruction of the original permeability matrix.

After reconstruction of the permeability field for each SVD scheme, reservoir simulation is conducted for each case to evaluate whether the reconstructed permeability matrices are good approximations to the original permeability matrix. Production profile and oil rates are compared for each SVD processed case with the base case.

5.4 Results

In this section, 2D reservoir simulation models are created to evaluate the effects of SVD on dimension reduction of the permeability field. The results are presented in three parts described below.

Part one is the application of SVD in parameterization of permeability field for an example problem that contains channel features. The dimension of the permeability field is 100×100. There are two major channels that has high permeability values than the surrounding background areas. SVD is performed on the permeability field to test whether it can save the major features from the input.

Figure 5.3 shows the original 100×100 permeability field. The yellow parts represent the high permeability channels and the purple areas represent low permeability background. SVD is performed on the 100×100 permeability matrix. Figure 5.4 shows the plot for the singular values and the cumulative energy versus different numbers of singular values. As shown in the right plot of figure 5.4, using 60 out of 100 singular values preserves more than 95% of the cumulative energy. Using 40 singular values preserves 90% of the cumulative energy.

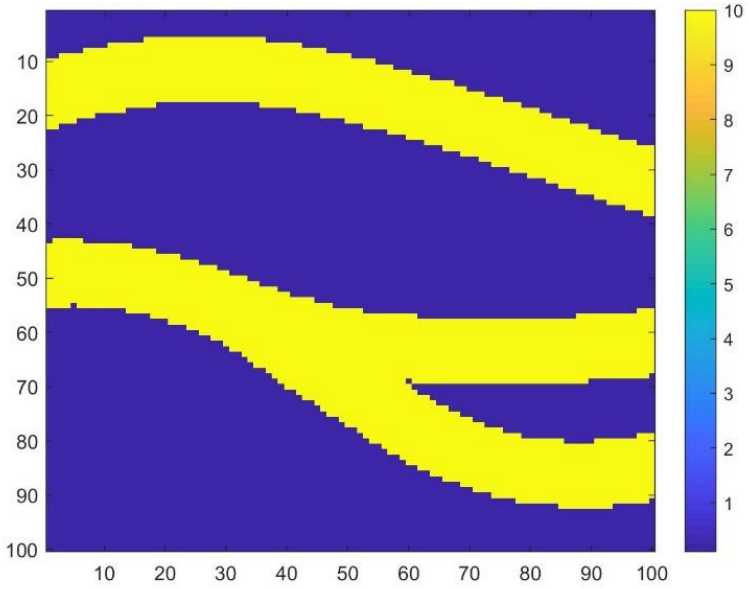

Figure 5.3. Distribution of permeability values for an example problem that contains channel features. The dimension of the permeability field is 100×100. There are two major channels that has high permeability values than the surrounding background areas.

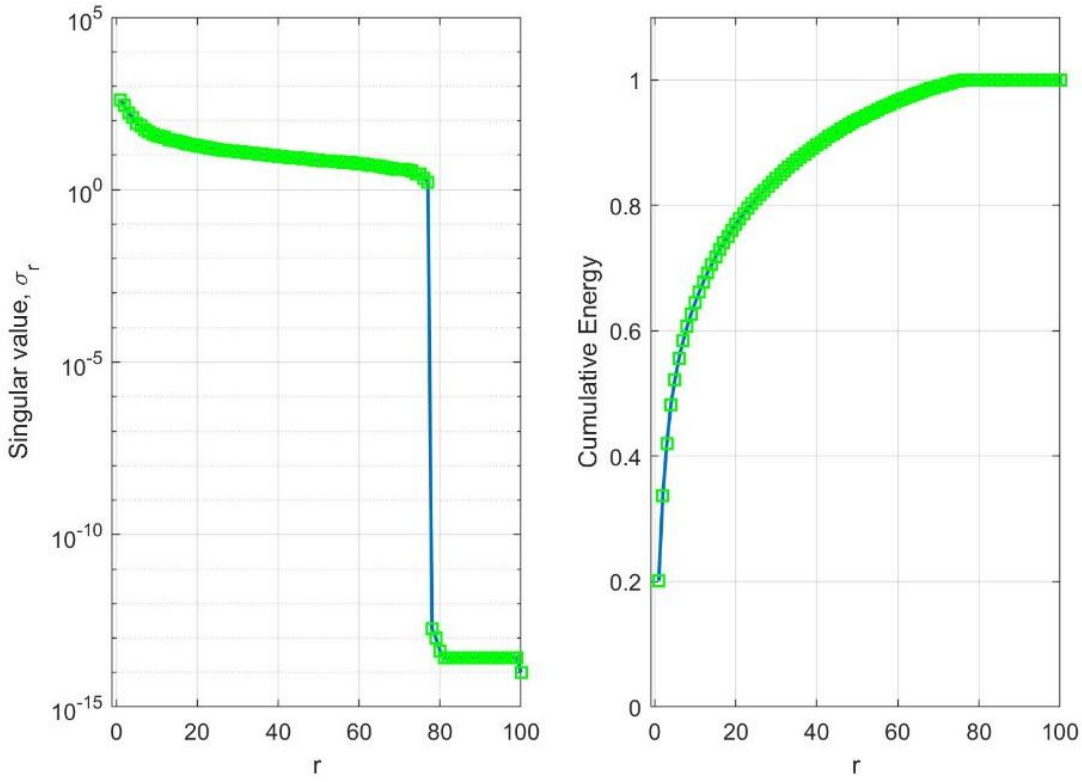

Figure 5.4. Plot of singular values and cumulative energy for the 100×100 permeability field.

Figure 5.5 shows the reconstructed permeability fields using different numbers of singular values k = 80, 60, 40, 20, 10 and 5. The reconstructed field effectively saves the major channel features even with small numbers of singular values being used. The lower left plot uses ten singular values (k = 10) and the two channels are still distinguishable. This shows that SVD effectively saves the main features from the original permeability field.

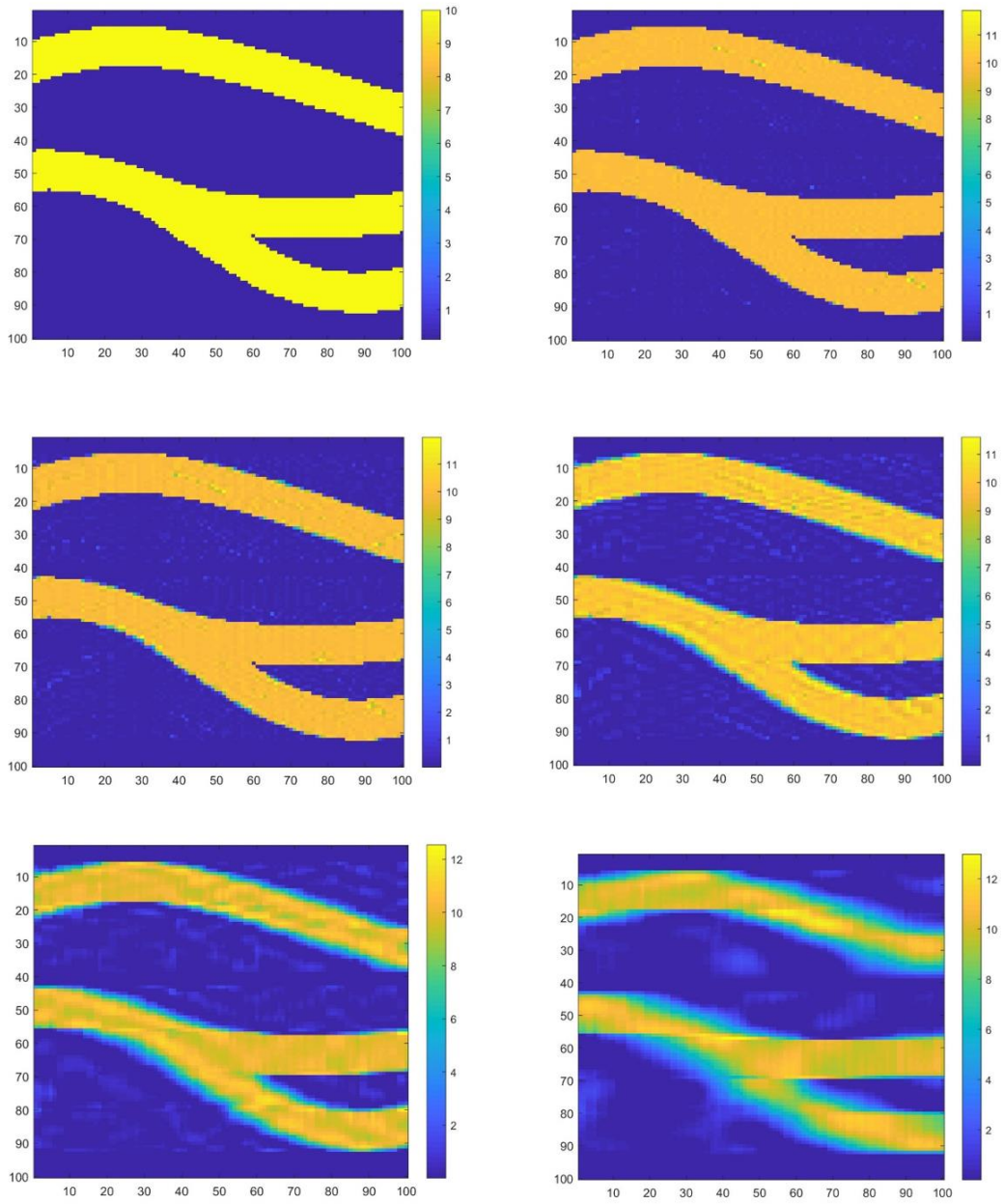

Figure 5.5. Example of the reconstructed permeability field using different levels of SVD processing. From upper left to lower right: k = 80, 60, 40, 20, 10 and 5.

The second part is to evaluate the application of SVD in parameterization of permeability field for the SPE tenth dataset. Figure 5.6 shows the distribution of the original permeability values of the reservoir model. The dimension is 100×20. High permeabilities zone represents highly conductive fluvial channel features.

SVD is then performed on the permeability matrix and less amount of singular values are used to reconstruct the original permeability matrix. The total number of singular values is 20. In figure 5.7 and figure 5.8, 15 and 10 singular values are used to reconstruct the original matrix. The major channels are visible using smaller amount of singular values in both figure 5.7 and figure 5.8, meaning that the SVD processed permeability matrices are good representations of the original permeability values.

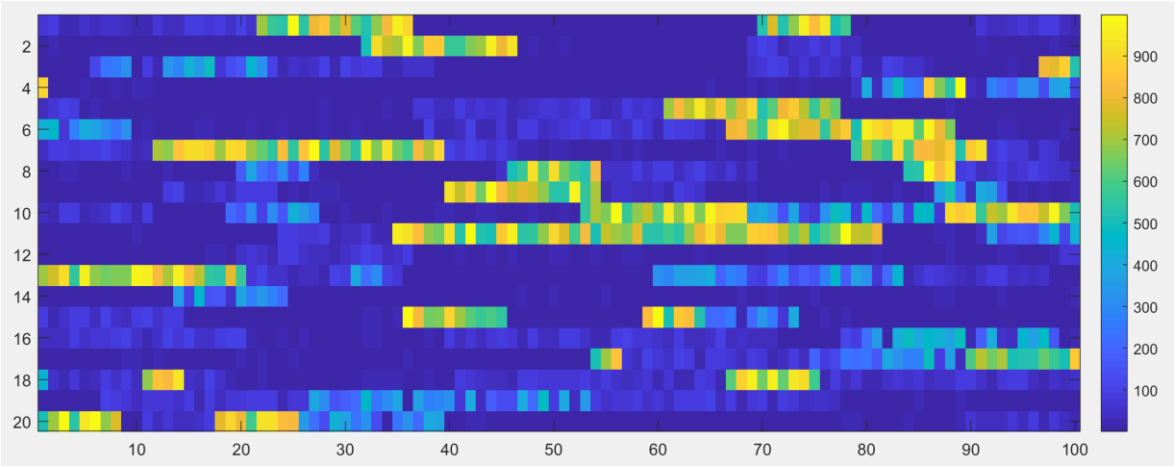

Figure 5.6. Distribution of permeability (kx) for the original 100×20 model. High permeabilities zone represents channel features. The data is from the SPE tenth comparative solution project.

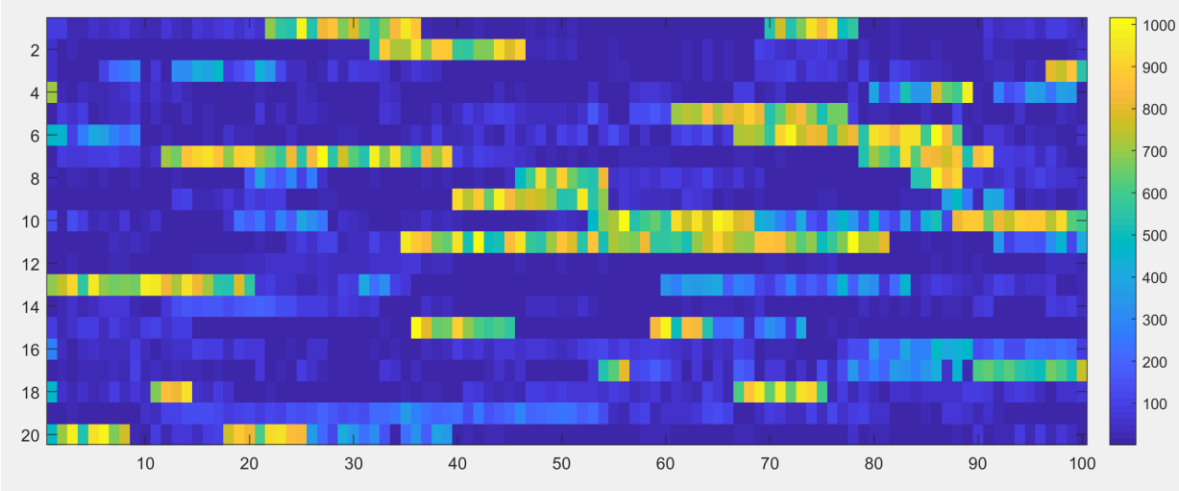

Figure 5.7. Distribution of the reconstructed permeability matrix using 15 singular values. The major channels are still visible using smaller amount of singular values.

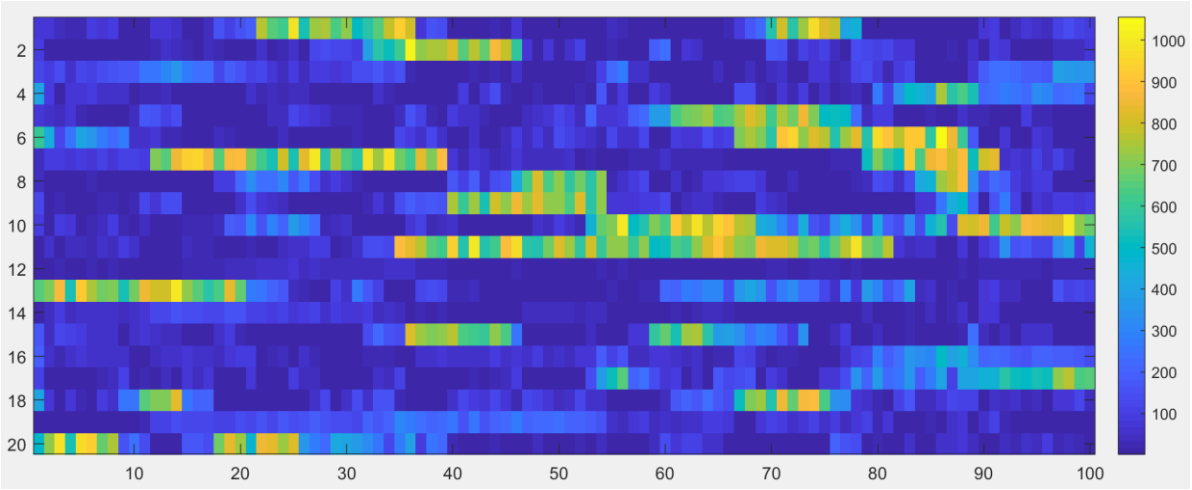

Figure 5.8. Distribution of the reconstructed permeability matrix using 10 singular values.

Figure 5.9 shows the plot for the singular values and the cumulative energy versus different numbers of singular values. As shown in the right plot of figure 5.9, using 15 singular values preserves 93% of the cumulative energy. Using 10 singular values preserves 80% of the cumulative energy.

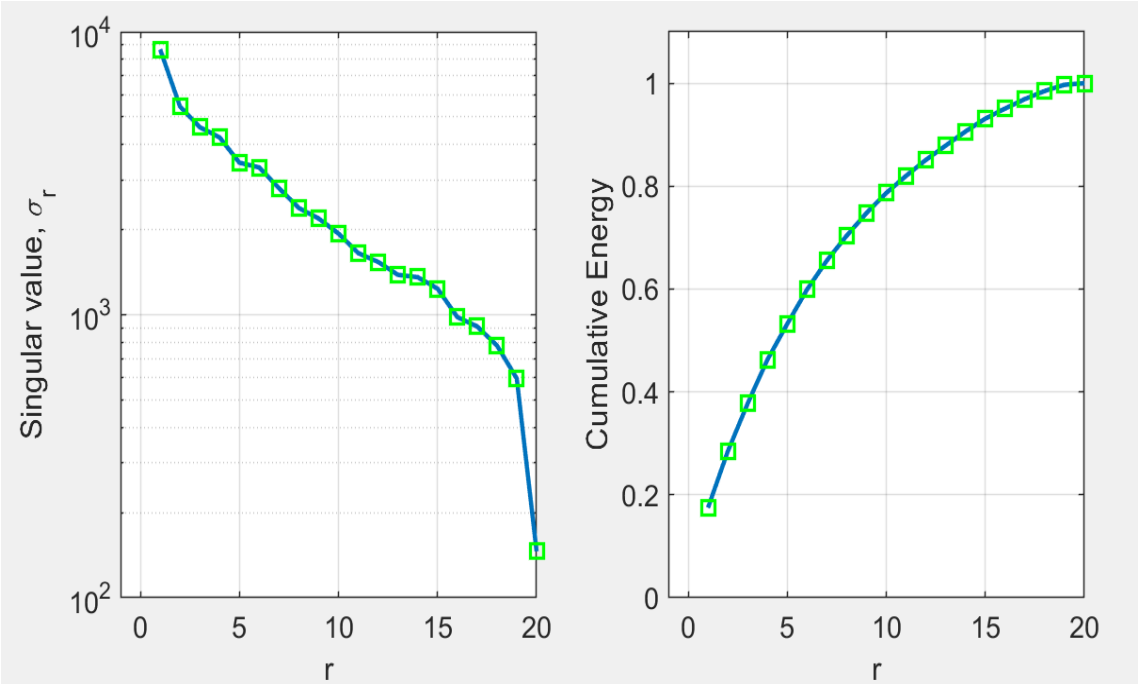

Figure 5.9. Plot of singular values and the cumulative energy. There are 20 total singular values.

Reservoir simulation is conducted to evaluate whether the reconstructed permeability matrices are good approximations of the original permeability matrix. In the reservoir model, an injection well locates at the left side of the model, and a production well locates at the right side of the model.

Figure 5.10 shows the permeability distribution in CMG. Figure 5.11 shows the distribution of oil flux. Red arrows indicate the magnitude and direction of oil flux vectors.

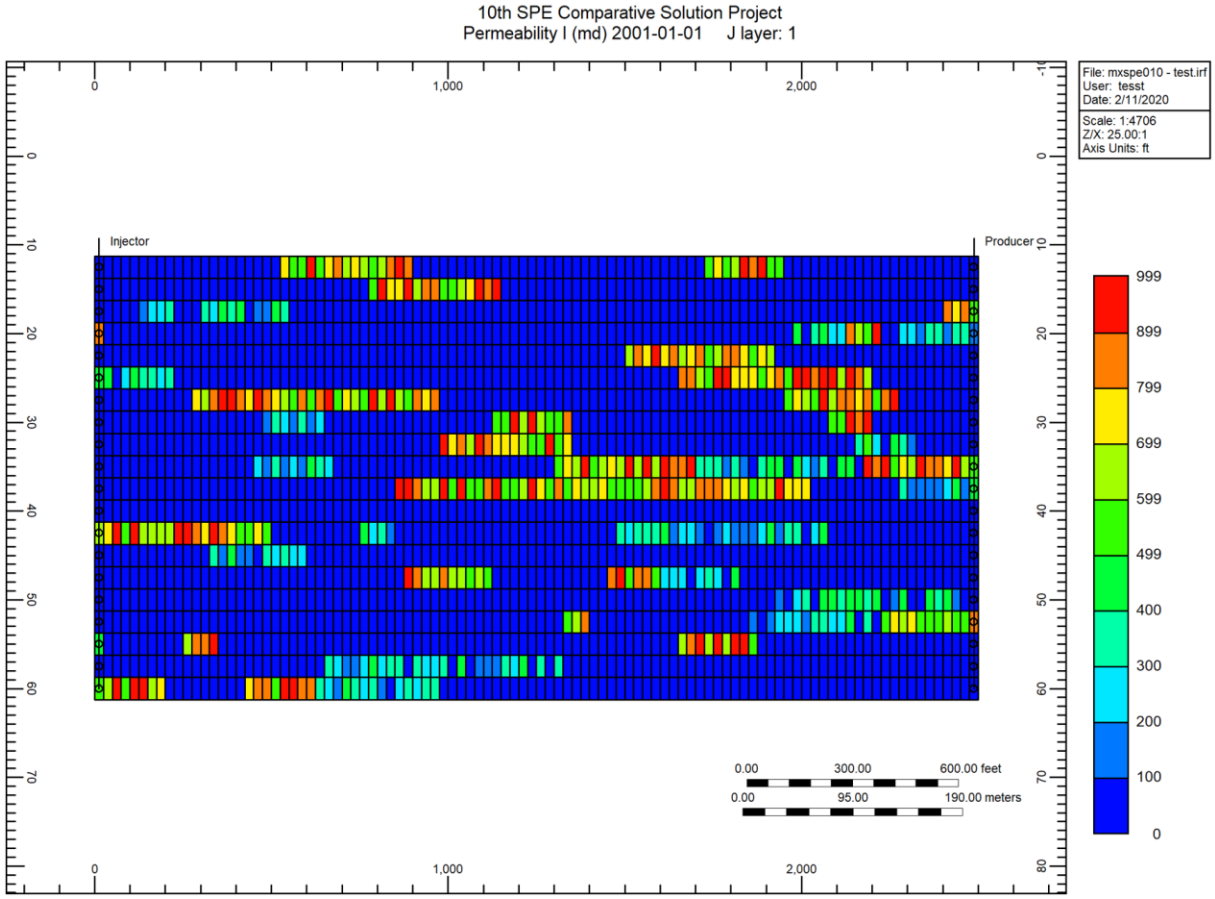

Figure 5.10. Distribution of permeability in CMG. Red color represents grid blocks with high permeability values.

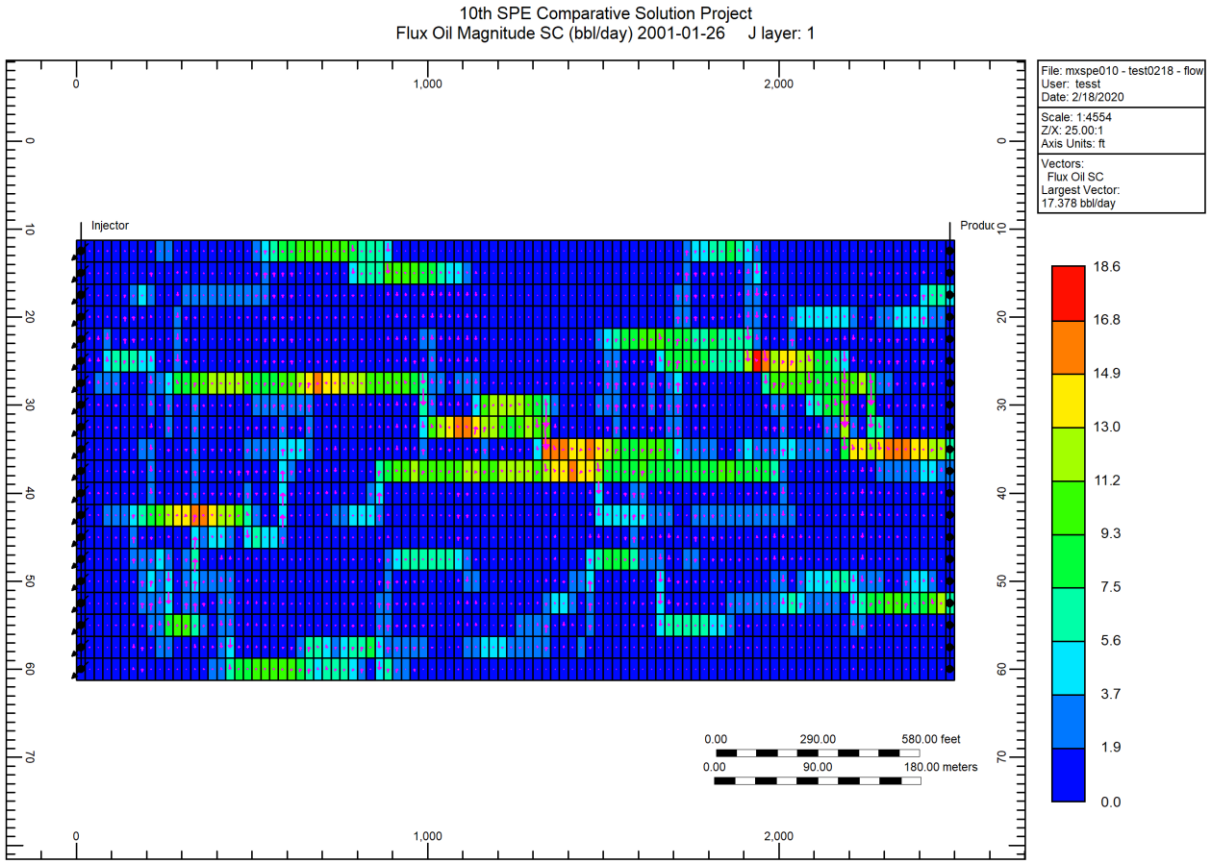

Figure 5.11. Distribution of oil flux in CMG. Red arrows indicate the magnitude and direction of oil flux vectors.

Figure 5.12 and figure 5.13 show the cumulative oil production and oil rate for different permeability matrices reconstructed using different numbers of singular values (k). They show that the overall trend for the oil production is preserved by using reconstructed permeability matrices, indicating that SVD is valid in the parameterization of permeability values. When using more than 15 singular values to reconstruct the permeability values, the simulated oil production profile is similar to the original case using the original permeability values.

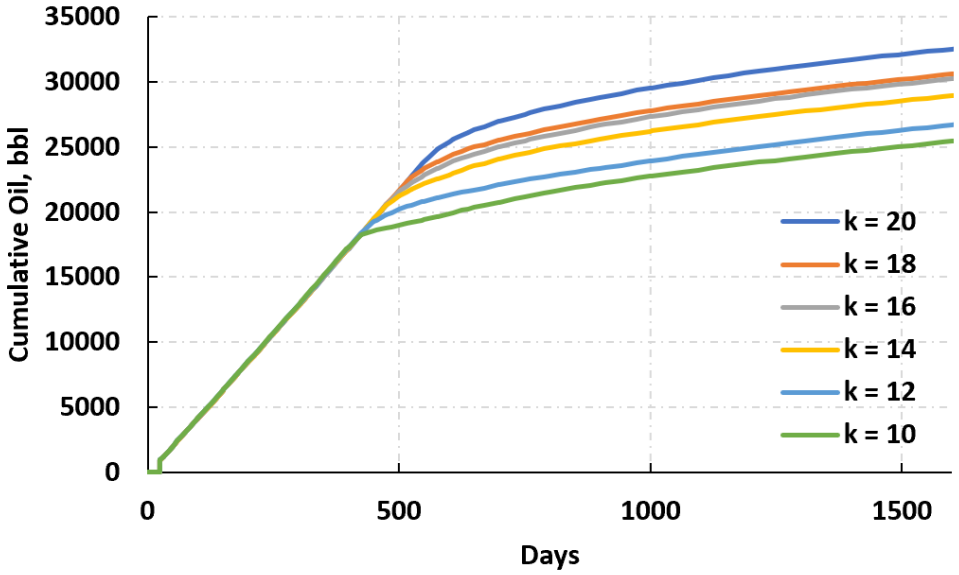

Figure 5.12. Cumulative oil production for different permeability matrices reconstructed by different numbers of singular values.

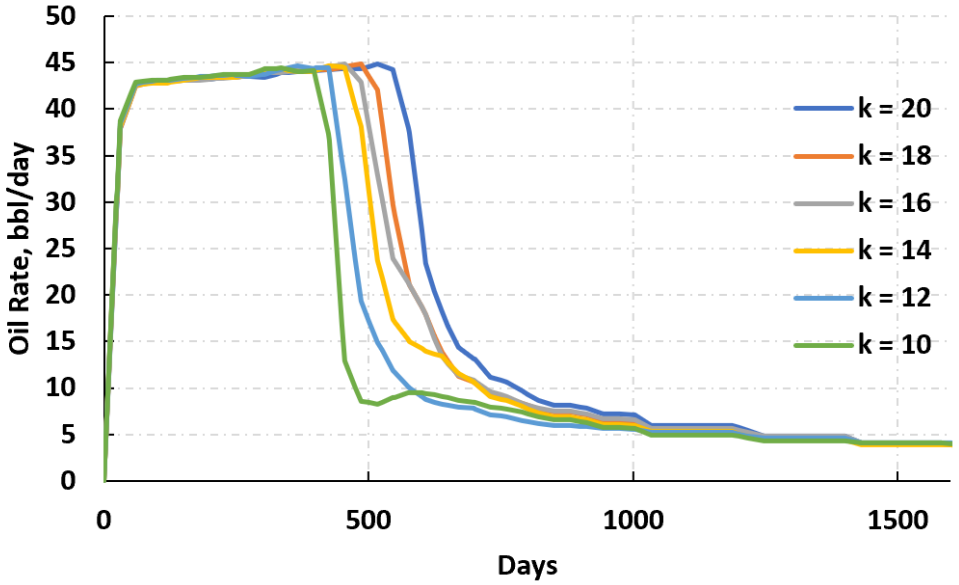

Figure 5.13. Oil rate for different permeability matrices reconstructed using different numbers of singular values.

Part three is to evaluate the application of SVD in upscaling. There are three sub cases created to evaluate the effects of SVD on upscaling. The first case is the base model, which has 20×10 grid blocks. The second case is to upscale permeability in the x direction, which has 10×10 grid blocks. The third case upscales in both x and y direction and it has 10×5 grid blocks.

Figure 5.14 shows the distribution of the permeability value for the original 20×10 model (case 1). Yellow blocks have higher permeability values, and blue indicates lower permeability values. It is shown from the distribution that there exist linear features that have relatively higher permeability values, which represent fluvial or channelized environment.

Figure 5.15 plots the distribution of the permeability value for the upscaled 10×10 model. Upscaling was performed by taking the average permeability value for two adjacent grid blocks in the x direction from case 1.

Figure 5.16 shows the distribution of the permeability value for the upscaled 10×5 model. Upscaling was performed by taking the average permeability value for adjacent grid blocks in both the x and y direction from case 1.

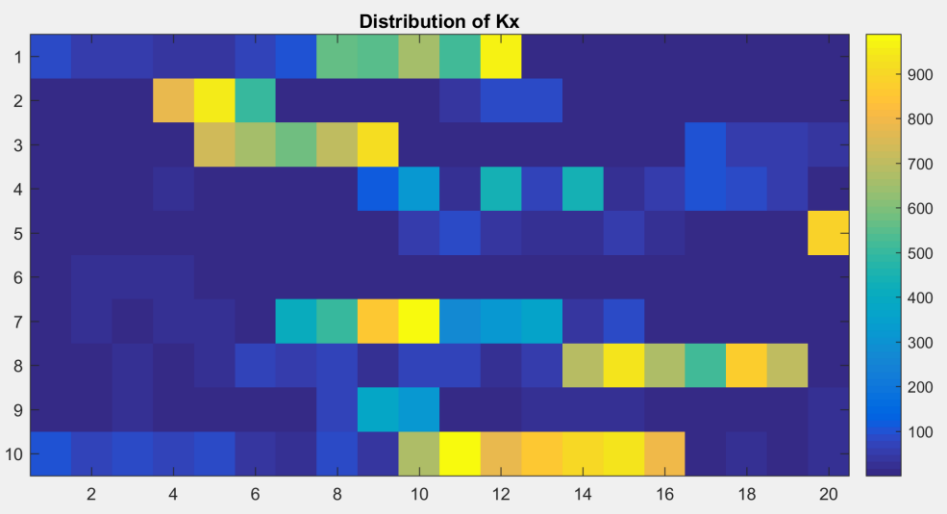

Figure 5.14. Distribution of permeability (kx) for the original 20×10 model (case 1). The data is from the SPE Tenth Comparative Solution Project.

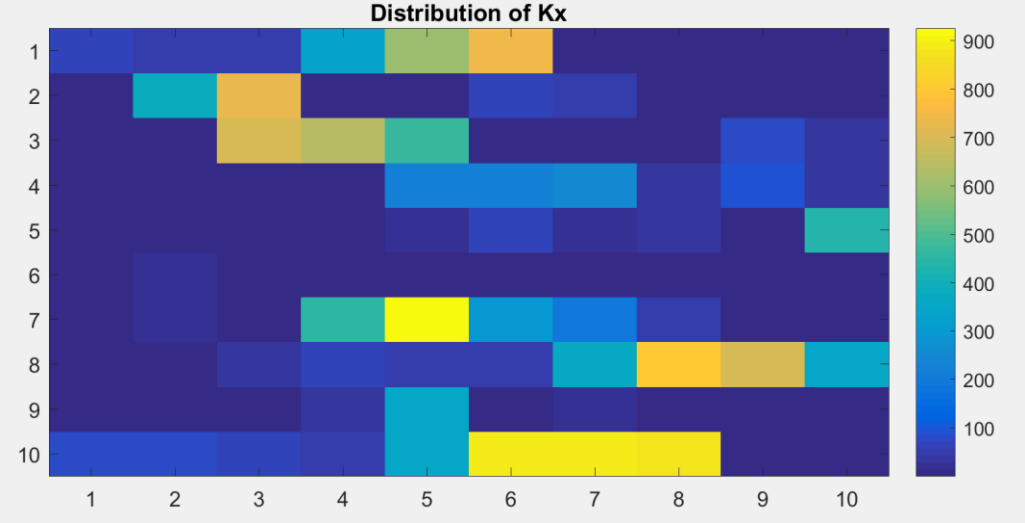

Figure 5.15. Distribution of permeability (kx) for the upscaled 10×10 model (case 2). Upscaling was performed by taking the average permeability value for two adjacent grid blocks in the x direction from the original model (case 1).

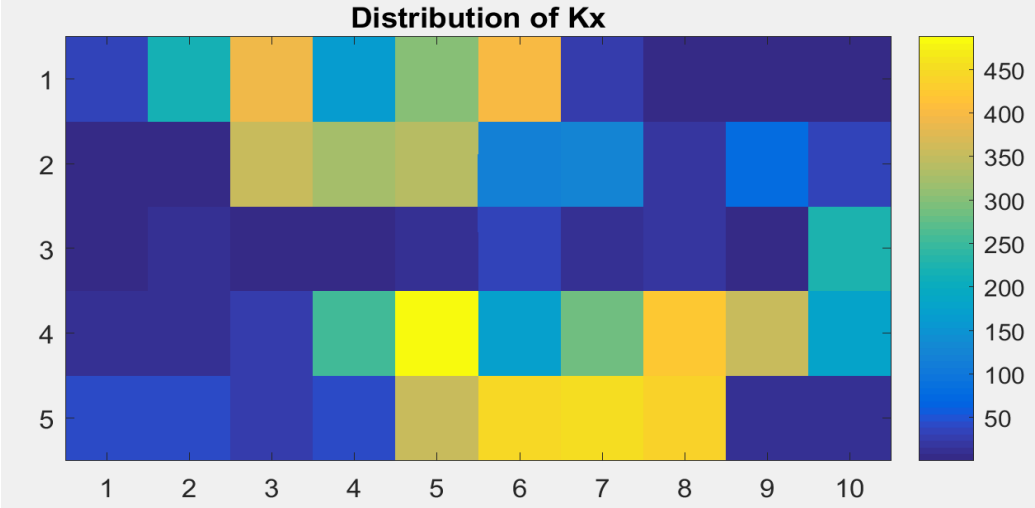

Figure 5.16. Distribution of permeability (kx) for the upscaled 10×5 model (case 3). Upscaling was performed by taking the average permeability value for two adjacent grid blocks in the x and y direction from the original model (case 1).

A reservoir simulator is developed using the IMPES method and is based on block centered grid system. There is one injection well at the (3,3) block and one production well at the (18,8) block.

Figure 5.17 shows the distribution of oil saturation after simulation for 10 years for the original model with the dimension of 20×10 (case 1). Oil saturation was lower around the injector

and higher around the producer. Figure 5.18 shows the distribution of pressure distribution after 10 years for the original model (case 1). Pressure is relatively higher near the injector and lower near the producer.

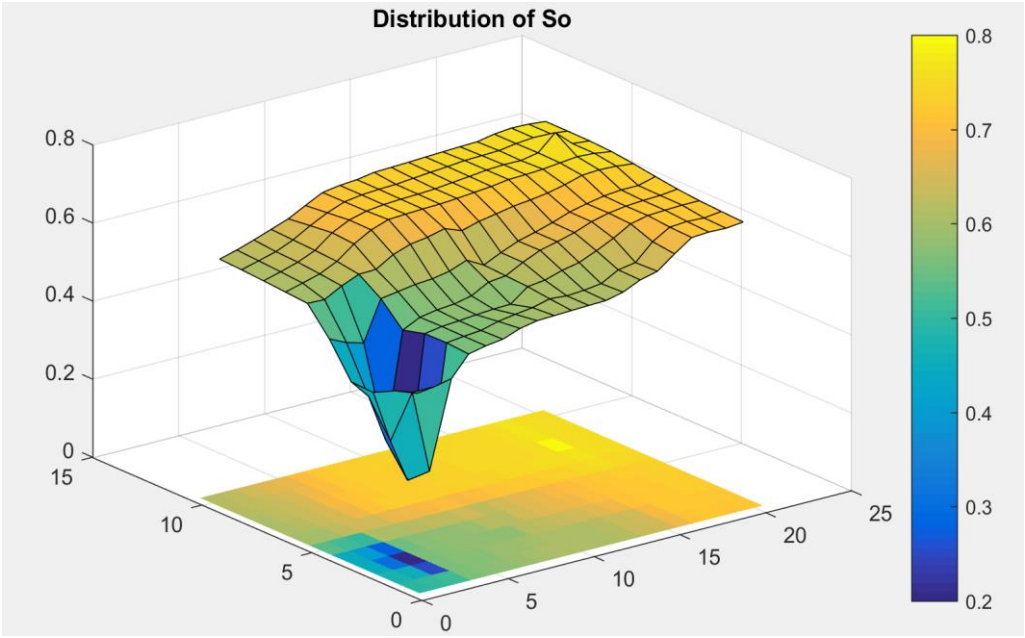

Figure 5.17. Distribution of oil saturation (So) at each grid block after simulation for 10 years for the original model with the dimension of 20×10 (case 1).

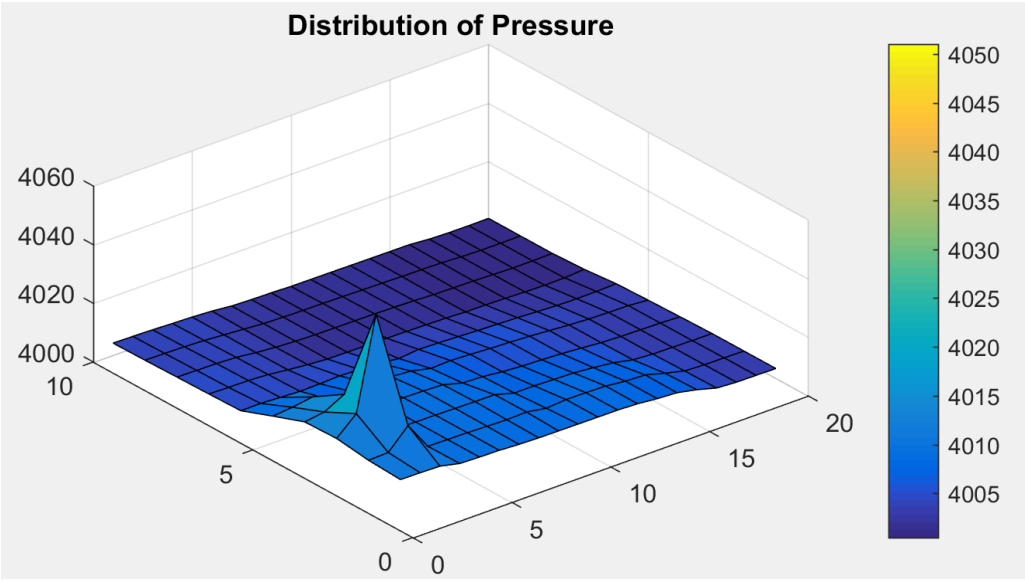

Figure 5.18. Distribution of pressure at each grid block after simulation for 10 years for the original model with the dimension of 20×10 (case 1).

Figure 5.19 and figure 5.20 show the distribution of oil saturation and pressure after 10 years for case 2. The overall pattern for oil saturation and pressure are similar to these in case 1 (figure 5.17 and figure 5.18). Oil saturation was relatively lower around the injector and higher around the producer. There exists relatively higher pressure around the injector, and lower pressure near the producer. It is also noticed that, since the resolution in case 2 was lower than the resolution in case 1, the distribution patterns are less smooth than these in case 1.

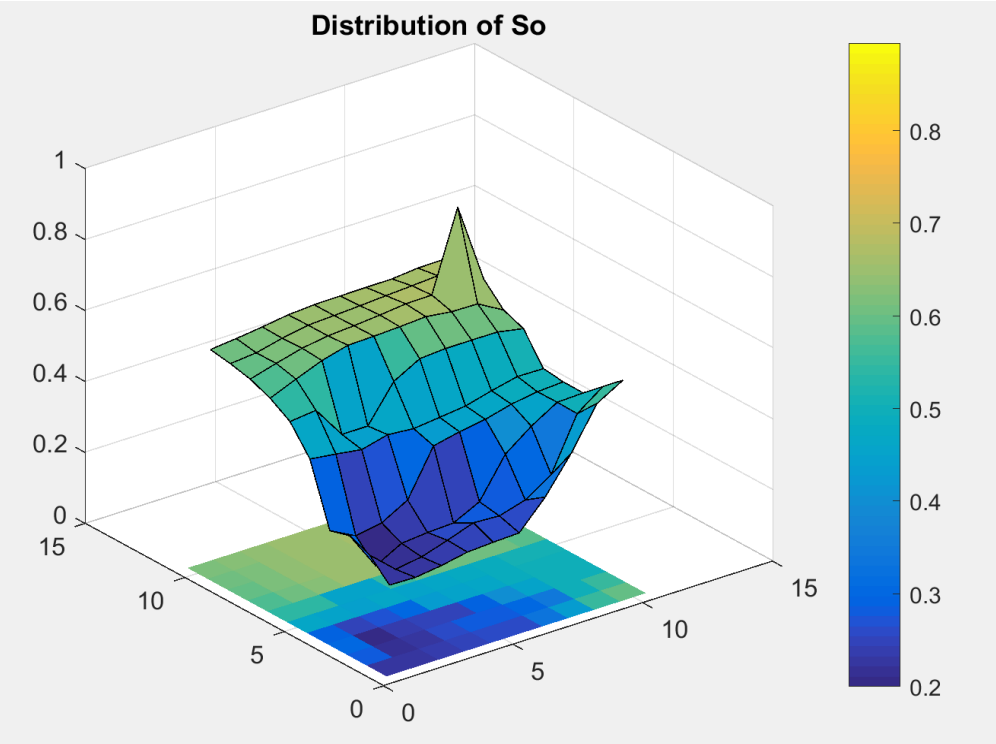

Figure 5.19. Distribution of oil saturation (So) at each grid block after simulation for 10 years for the original model with the dimension of 10×10 (case 2).

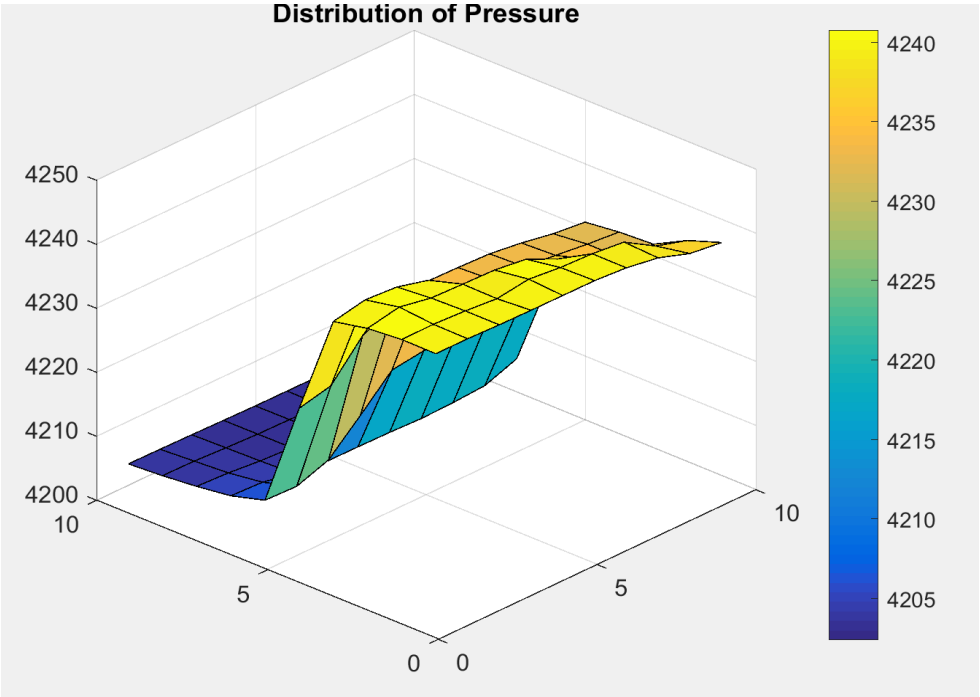

Figure 5.20. Distribution of pressure at each grid block after simulation for 10 years for the original model with the dimension of 10×10 (case 2).

Figure 5.21 shows the cumulative oil production for case 1 and case 2. Figure 5.22 shows a zoomed-in view for the change of cumulative oil production during the later time (2400 days to 3600 days) of the simulation. Figure 5.23 shows the oil rate for case 1 and case 2. In all three figures from 21 to 23, the orange line represents the case for the base 20×10 model (case 1). The light blue line represents the case for the upscaled case (case 2). The rest three lines represent the three cases where SVD were performed in case 2, among which different numbers of singular values were truncated. SVD 1, SVD 2, and SVD 3 represent the cases where the minimum one, two and three singular values were truncated respectively.

As shown in figure 21 to 23, the cumulative oil production and oil rate for the upscaled case are similar to the base model, which means that the upscaling scheme is a relatively good representation of the original model. However, there is not much difference for the upscaled case and the three cases where SVD is performed; the cumulative oil production profile for the SVD 1 and the upscaled case is similar. SVD 2 and 3 shows slight worse performance than SVD 1.

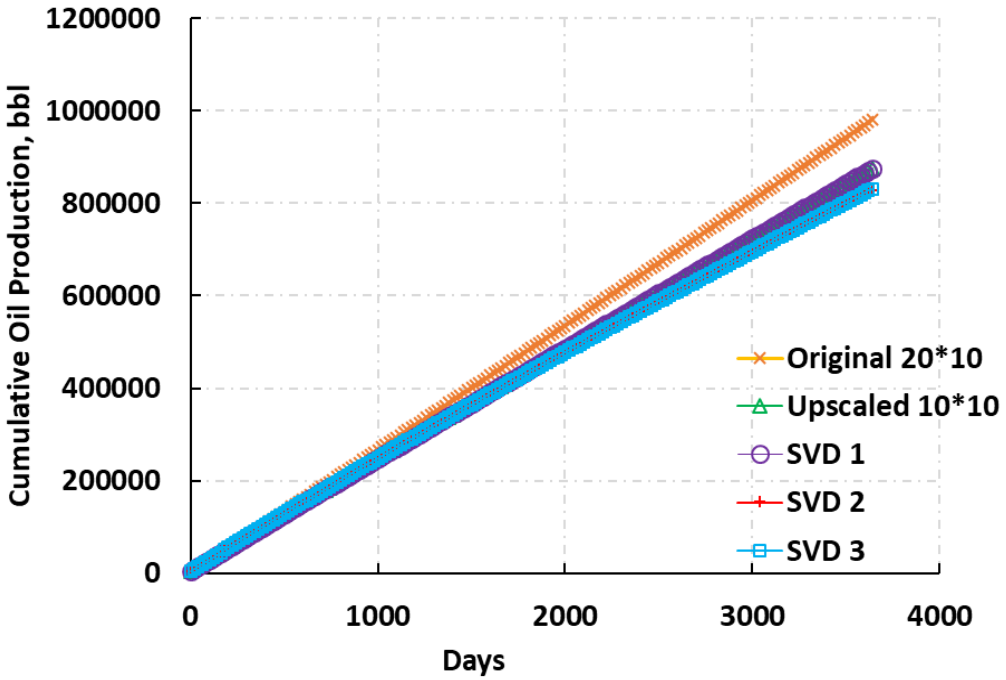

Figure 5.21. Cumulative oil production for 10 years for case 1 and 2. The orange line represents the base model. The green line represents the upscaled model. The rest three lines represents the SVD cases with 1, 2, and 3 singular values being truncated.

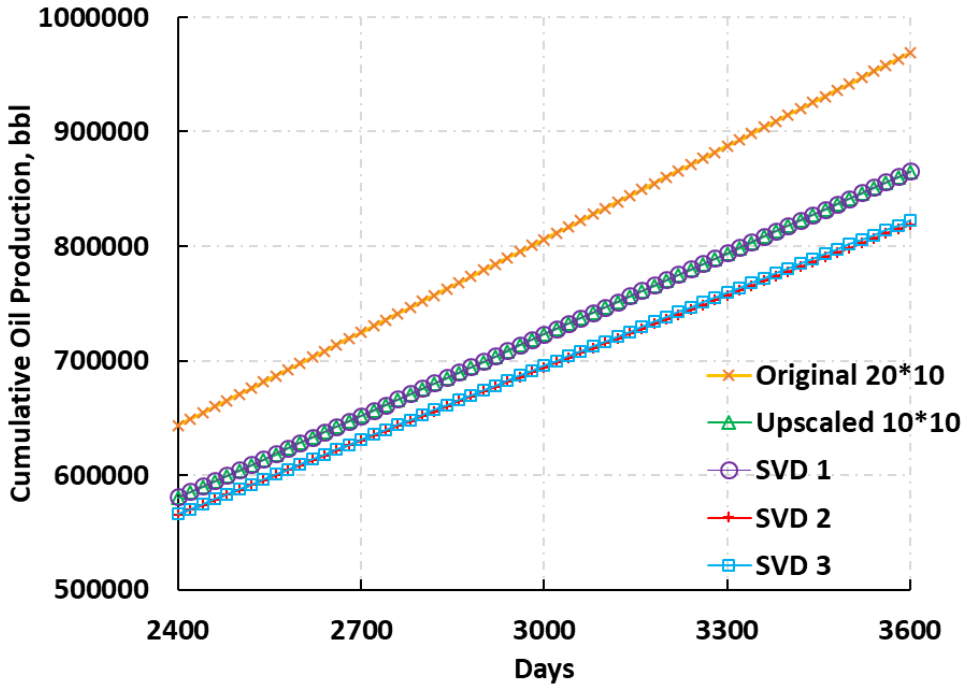

Figure 5.22. A zoomed in view for the cumulative oil production from 6 to 10 years for case 1 and 2.

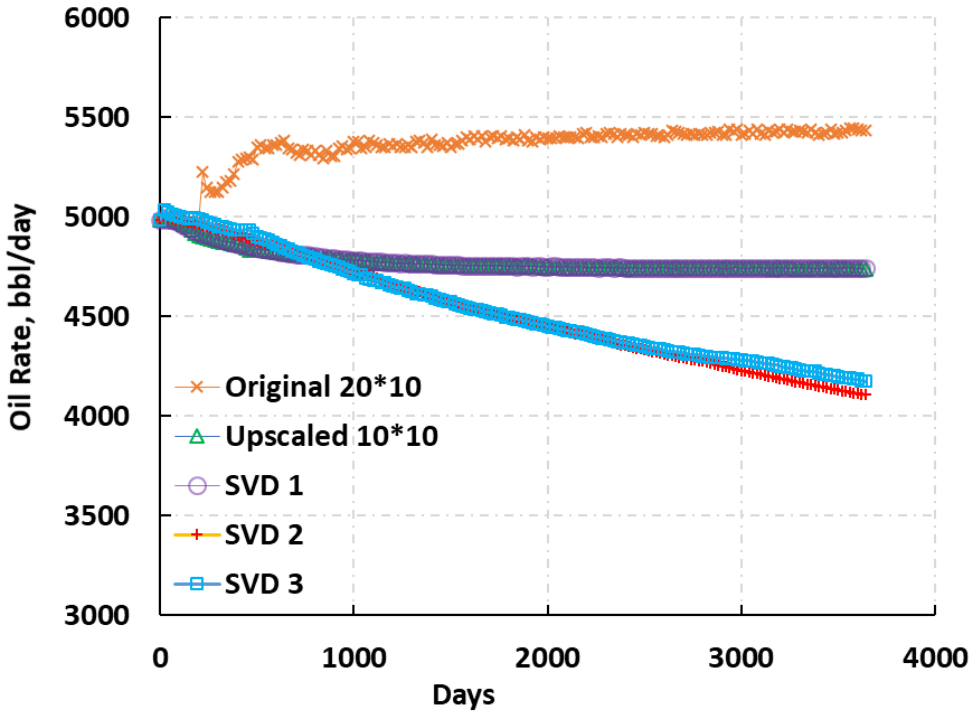

Figure 5.23. Oil rate for 10 years for case 1 and 2.

Figure 5.24 shows the comparison of the cumulative oil production for case 1 and case 3. Figure 5.25 shows a zoomed in view for change of cumulative oil change during the later time (2400 days to 3600 days) of the simulation. Figure 5.26 shows the oil rate for case 1 and case 3. Similar patterns are shown in figure 5.21 and 5.24. However, the upscaled model in case 3 (figure 5.24) is less accurate than that for case 2 (figure 5.21). This is within our expectation since case 3 has a much lower resolution.

There is also not much difference for the oil production and oil rate for the upscaled case and the three SVD cases from figure 5.24 to 5.26. In other words, the oil production and oil rate trend for the 10×5 upscaled model and the three SVD cases were similar.

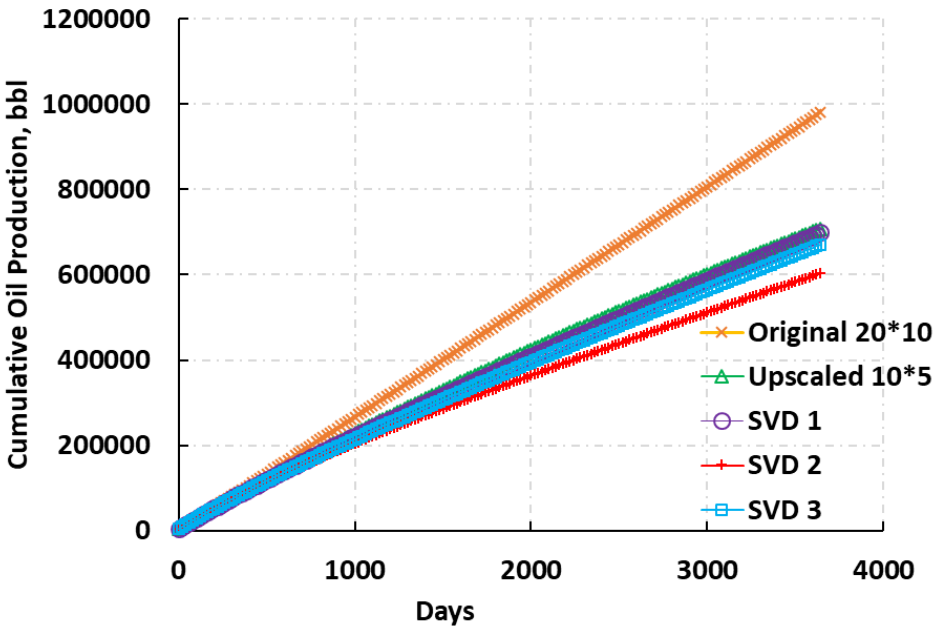

Figure 5.24. Cumulative oil production for 10 years for case 1 and 3. The orange line represents the base model. The green line represents the upscaled model. The rest three lines represents the SVD cases with 1, 2, and 3 singular values being truncated.

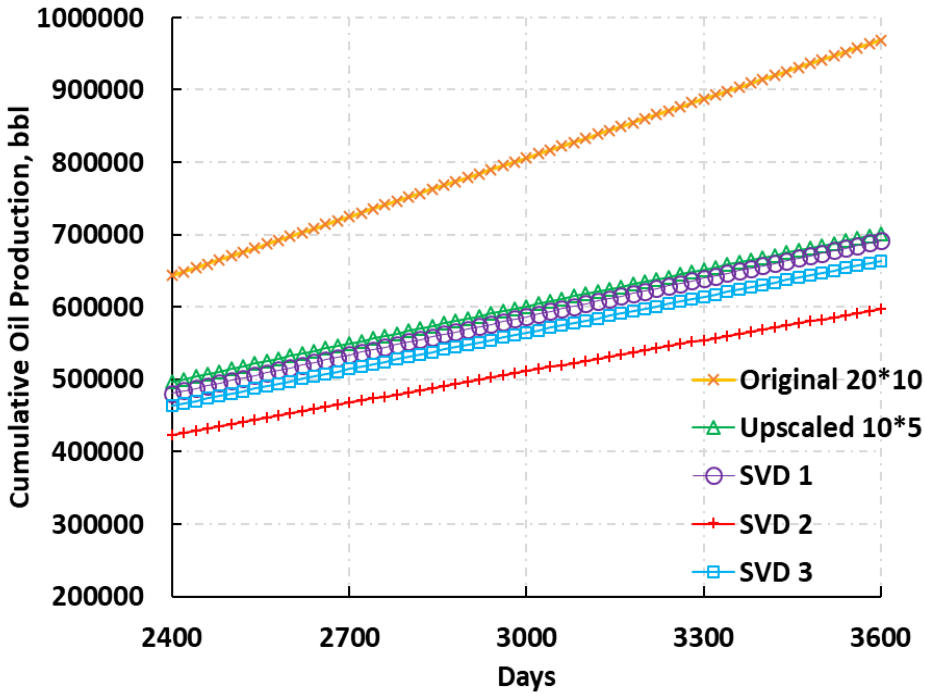

Figure 5.25. A zoomed in view for the cumulative oil production from 6 to 10 years for case 1 and 3.

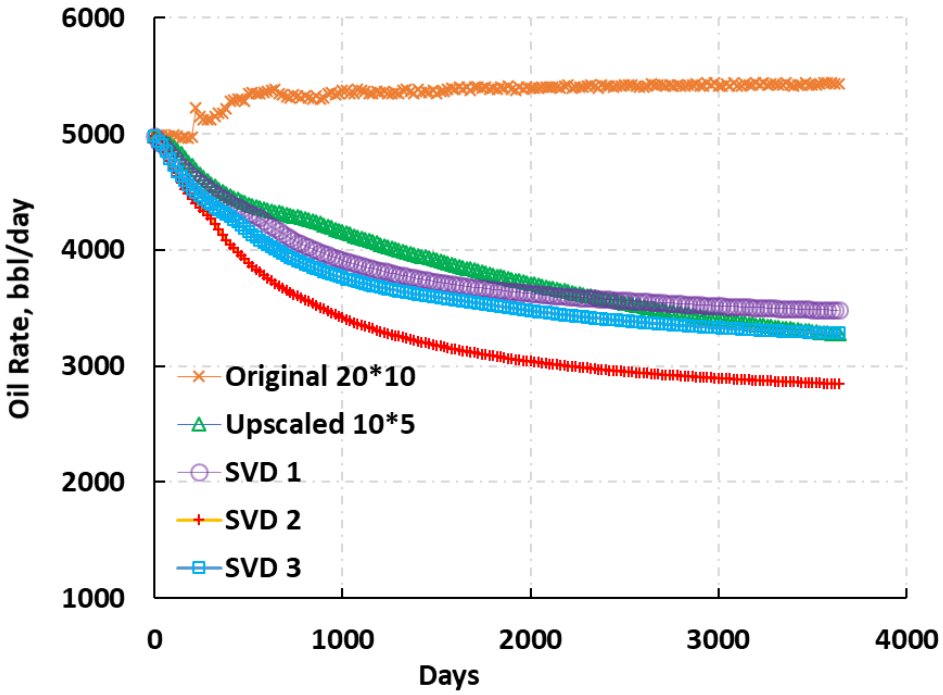

Figure 5.26. Oil rate for 10 years for case 1 and 3.

The reason that SVD does not show significant improvement in the accuracy of the simulation could be SVD processing does not reduce the size of the permeability matrix. Even though some singular values are truncated, and less total amount of singular values are used, the reconstructed permeability matrix still have the same size as the original permeability matrix. Thus, SVD did not significantly change the model accuracy.

Simulation time is collected for all cases to compare the model efficiency (Table 1). The base model (case 1) with 20×10 grid bocks runs for 676.3s to simulate production for 10 years. In case 2, upscaling in the x direction reduced the simulation time to be about 20% of case 1. Continuously upscaling in the y direction (case 3) also reduced the simulation time to be about 20% of case 2. Within case 2 and case 3, there is not much difference in the simulation time for applying SVD on the upscaled permeability.

In the future work, more complex models will be tested in to further test the effect of SVD on upscaling. The number of grid bocks can be increased. Additional layers can be added to the reservoir model as well.

Table 5.1. Summary of simulation time for each case.

	Grid Size	SVD Processing	Time (s)
Case 1	20*10	NA	676.3
Case 2	10*10	NA	132.2
	10*10	Drop 1 singular value	121.8
	10*10	Drop 2 singular values	125.1
	10*10	Drop 3 singular values	131.2
Case 3	10*5	NA	21.7
	10*5	Drop 1 singular value	21.22
	10*5	Drop 2 singular values	18.34
	10*5	Drop 3 singular values	20.31

5.5 Discussion

Here we compared SVD processing on the log scale. Three additional cases were compared to evaluate whether performing SVD in the log scale helps save the major features from the original input. The results are show in below figures. Figure 5.27 shows the distribution of permeability after using 60 singular values to reconstruct the original permeability field. Figure 5.28 shows the distribution of reconstructed permeability values after SVD processing on log scale of the permeability field with k = 60.

Comparison of 5.27 and 5.28 shows that the range of permeability values in figure 5.28 is significantly increased. This is due to the amplification of values during the reconstruction of permeability values after SVD processing.

Figure 5.29 and 5.30 shows the plot of singular values and the cumulative energy for different numbers of singular values for SVD on the original scale versus the log scale. The trend for regular SVD processing on the original permeability field is similar to that on the log scale.

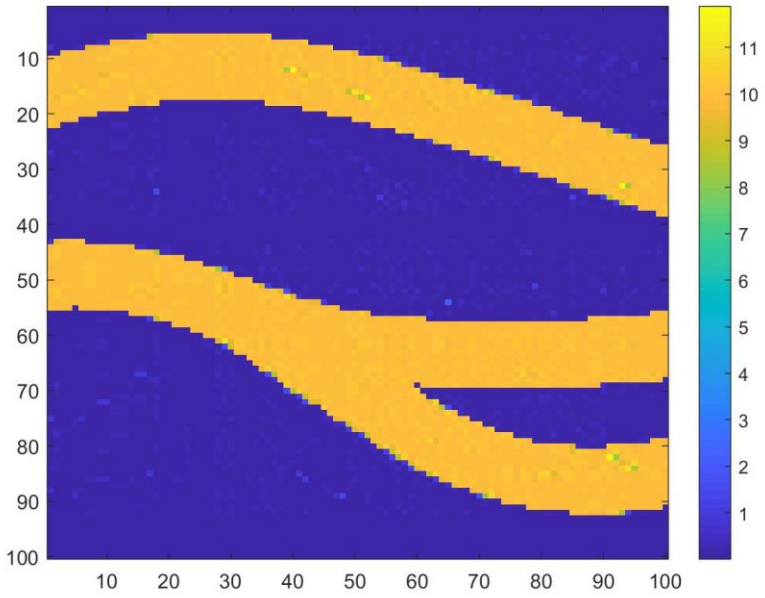

Figure 5.27. Distribution of permeability values after SVD with k = 60.

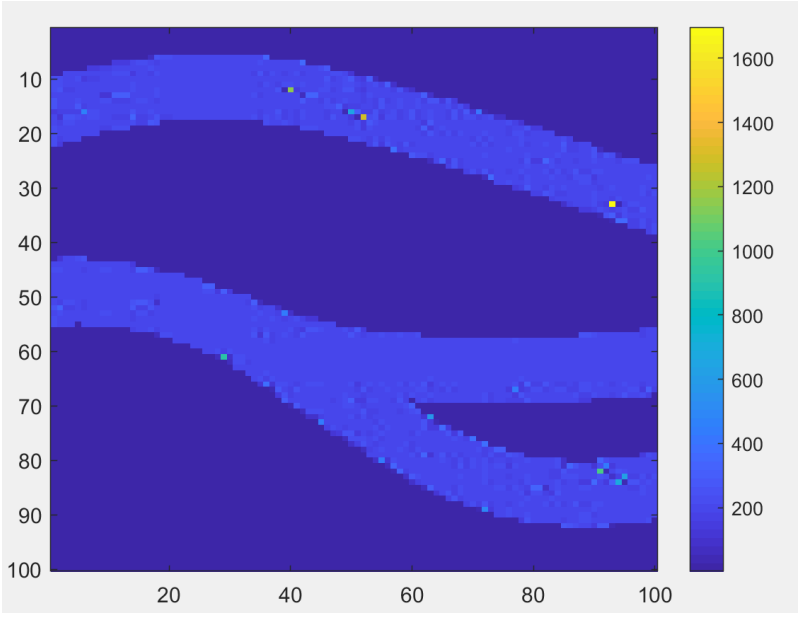

Figure 5.28. Distribution of permeability values after SVD on log scale of permeability, k = 60.

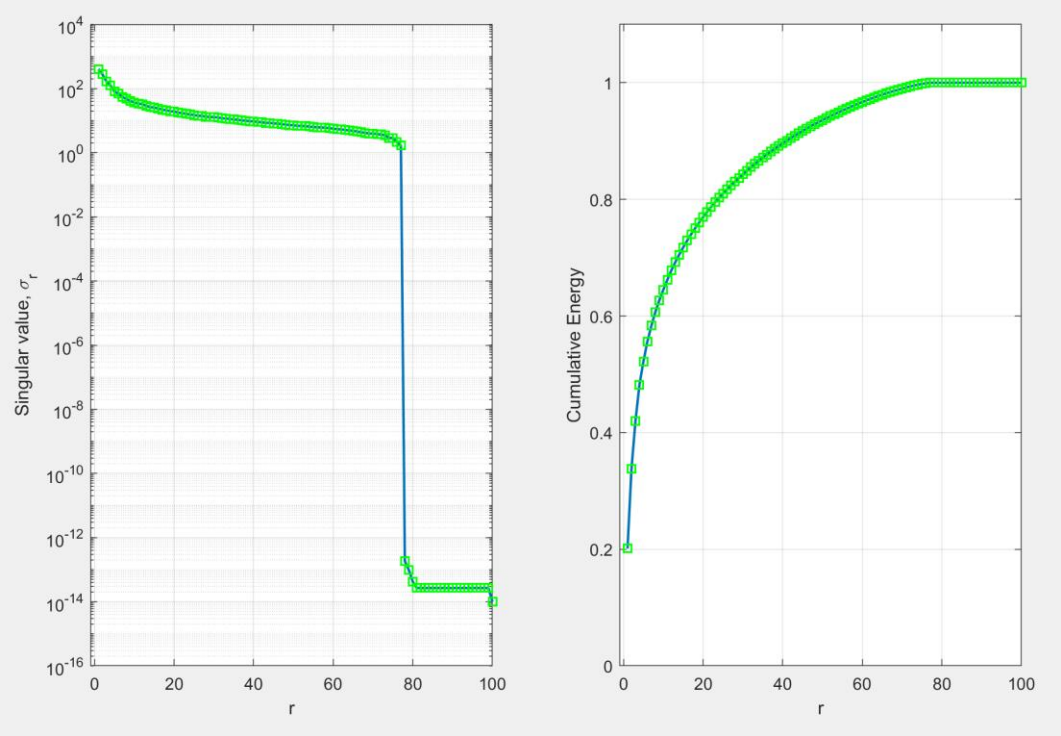

Figure 5.29. Plot of singular value and cumulative energy after SVD.

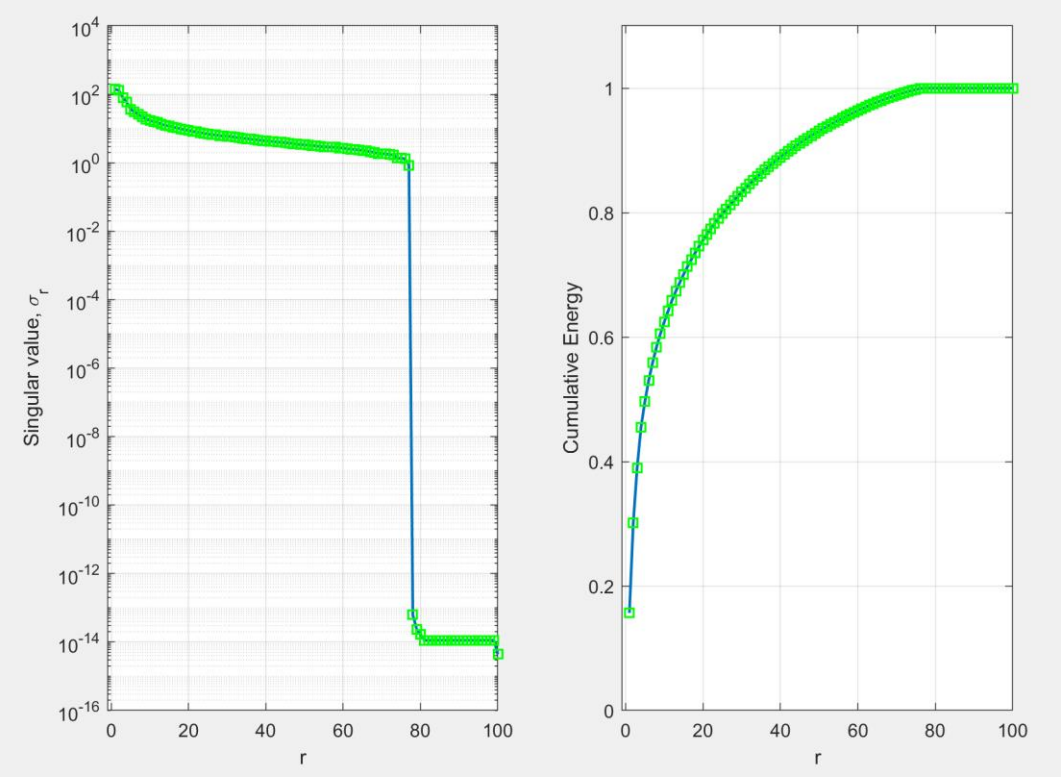

Figure 5.30. Plot of singular value and cumulative energy after SVD on the log scale of permeability.

Figure 5.31 to figure 34 shows the results for evaluating SVD on the log scale using the SPE tenth data. Figure 5.31 shows the distribution of permeability after using 15 singular values to reconstruct the original permeability field. Figure 5.32 shows the distribution of reconstructed permeability values after SVD processing on log scale of permeability values with k = 15.

Comparison of 5.31 and 5.32 shows that the range of permeability values in figure 5.32 is increased. This is also because of the amplification of values during the reconstruction of permeability values after SVD processing.

Figure 5.33 and 5.34 shows the plot of singular values and the cumulative energy for different numbers of singular values for SVD on the original scale versus the log scale. The trend for regular SVD processing on the original permeability field is similar to that on the log scale, meaning that SVD on the log scale does not improve the effectiveness of SVD for parameterization of permeability fields.

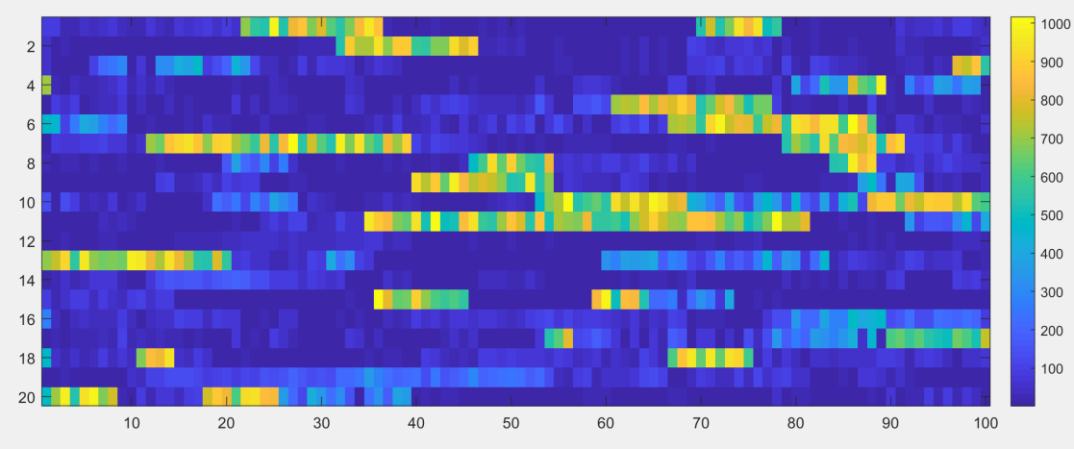

Figure 5.31. Distribution of permeability values after SVD with k = 15.

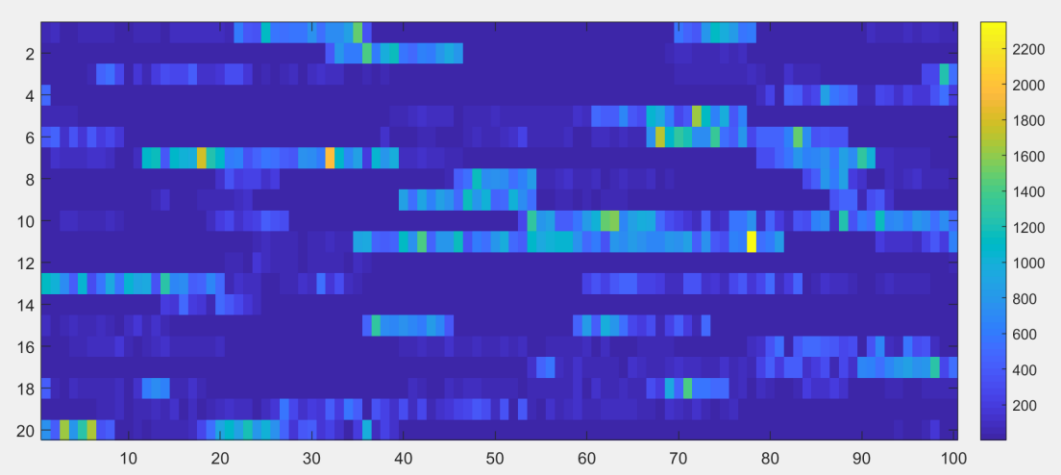

Figure 5.32. Distribution of permeability values after SVD on the log scale with k = 15.

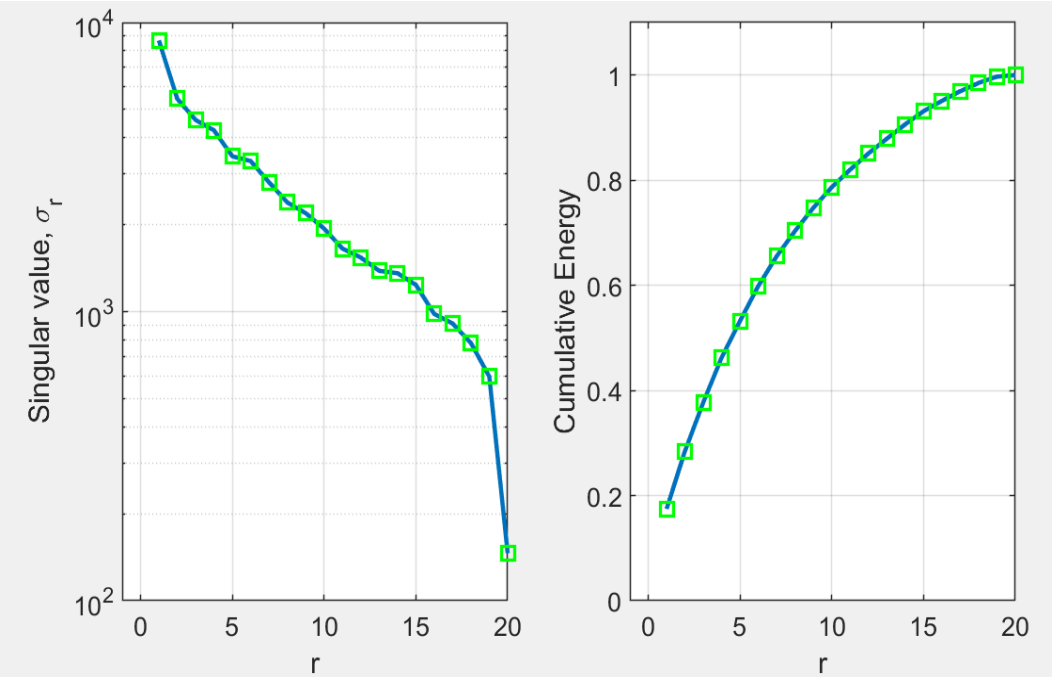

Figure 5.33. Plot of singular value and cumulative energy after SVD.

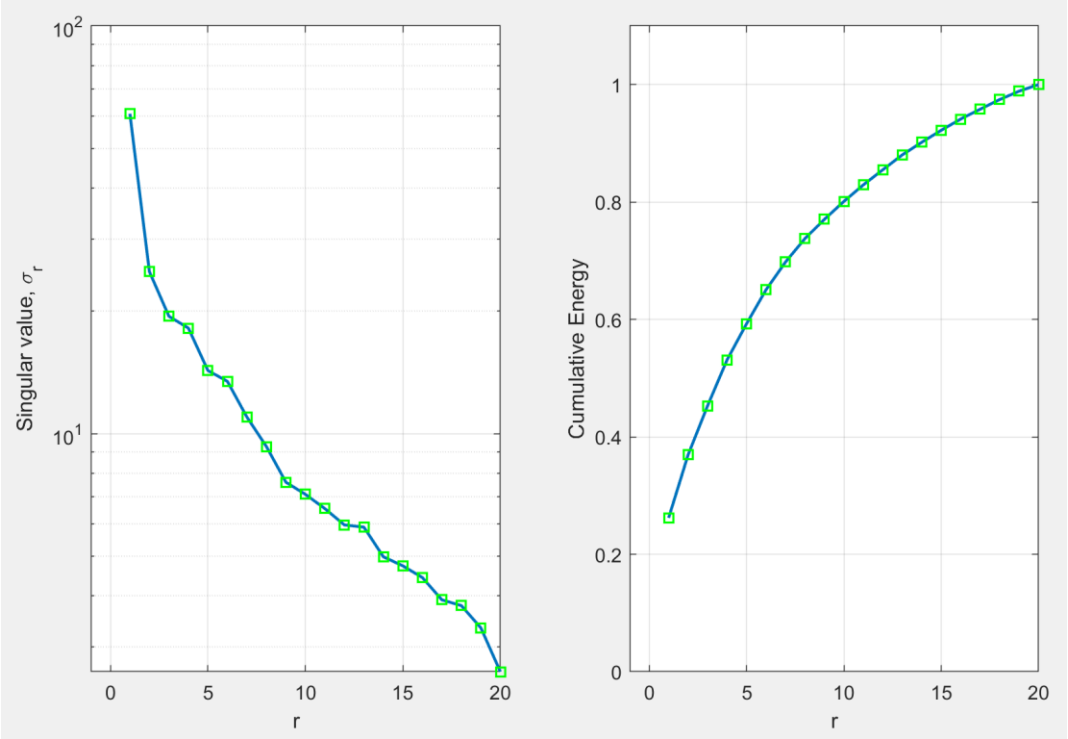

Figure 5.34. Plot of singular value and cumulative energy after SVD on the log scale of permeability.

5.6 Conclusion

This study evaluates the effect of singular value decomposition (SVD) in parameterization and upscaling on the SPE tenth comparative solution project. Simulation results show that SVD is valid in the parameterization of permeabilities. The reconstructed permeability matrices using certain amount of singular values are good approximations of the original permeability values. Simulation results using the reconstructed permeability matrices are similar to these using the original permeability values.

SVD is then applied on the upscaled permeability value. Reservoir simulation is performed for different cases with different numbers of singular values being used. Simulation results were compared between the base case, upscaled case, and SVD upscaled case. The effectiveness and efficiency of SVD on upscaling in reservoir simulation is evaluated.

The reservoir simulation results did not show a significant improvement in the accuracy of predicting oil production by applying SVD on the upscaled permeability values. Comparing to the results from the upscaled cases, the SVD processed cases also did not show improvement of model efficiency. It is speculated that the reconstructed permeability matrix has the same size before and after the SVD processing, thus the model accuracy and efficiency are not significantly improved.

Future work may increase number of blocks, or layers to see whether it works with reservoir simulation models in larger scales. Different upscaling techniques may also be explored to select the best upscaling scheme, and the effect of SVD on various upscaling schemes can be further evaluated.

Chapter 6. Conclusions and Recommendations

In this research study, a comprehensive workflow is developed to utilize data-driven modeling for reservoir characterization and reservoir simulation. Several different data-driven studies explore the possibility to create the relationships among different datatypes and datasets in this research work. Well logs are used to predict rock facies. Petrophysical and reservoir properties in the reservoir can be estimated using the built relationship between seismic attribute values and reservoir properties. Seismic images can also be used to predict seismic facies. Lastly, Singular Value Decomposition (SVD) is explored for the reservoir permeability field with reduced dimensional parameterization to evaluate its effectiveness in understanding reservoir connectivity.

The first study explores the application of using petrophysical well logs and different data analytics models to automatically classify rock facies. Four different types of models, which include decision tree, random forest, support vector machine and neural network, are created to look for the best model to predict rock facies from the corresponding well log values.

Among the four models, random forest (RF) method showed the overall best performance. It has the highest testing accuracy (0.691) for classifying the facies. The testing accuracy is 0.921 for the adjacent facies, which is the second highest among all the methods studied. It was comparable to the predictive accuracy of adjacent facies from the support vector machine (SVM) method (0.926). However, the accuracy for classification of facies from the support vector machine model is 0.585, which is lower than the random forest predictive accuracy.

Future work may include adding different types of well logs to further increase the prediction accuracy. More samples and more data analytics methods may also be tested to see whether they can help increase the model performances.

In the second study, deep learning neural network models are created to build the relationships between the input seismic attribute values from the seismic survey and petrophysical properties from well logs. Four different cases with different types of seismic attributes are created to compare the influence of each seismic attribute on the model predictive performance. The results show that a deep learning neural network model with multi-hidden layers can be used to predict porosity values using extracted seismic attribute values from 3D seismic volumes. The model has higher accuracy in predicting porosity values as more seismic attribute values are added. In future studies, more attributes can be added to evaluate their impacts on the predictive accuracy.

Third study explores machine learning (ML) method in automatically identifying salt bodies from seismic images. A novel wavelet convolutional neural network (Wavelet CNN) method that combines wavelet transformation with CNN is introduced and applied for the task of identifying salt bodies from seismic images. The Wavelet CNN model uses all four components from the output of wavelet transformation to construct the pooling layers. It saves important information from the high-pass and low-pass filters during pooling operations.

The results show that the Wavelet CNN model can improve classification performance as compared to conventional CNN models that use max and mean pooling layers. The Wavelet CNN model has 0.949 testing accuracy after 35 epochs, which performs consistently although only marginally better than the other models. By utilizing multi-level wavelet transformation as the pooling layers, the input image size is reduced without losing the key features, resulting in improved computational efficiency and prediction accuracy.

This novel technique of integrating wavelet transformation with conventional CNN, can be expanded for other geophysical interpretations to automatically identify other geologic features in

the subsurface. Different wavelet transformation methods, and different combinations of pooling layers can also be tested.

The last study evaluates the effectiveness of singular value decomposition (SVD) in parameterization and upscaling for reservoir properties using the data from the SPE tenth comparative solution project. Reservoir simulation results show that SVD is valid in the reduced parameterization of permeability fields. The reconstructed permeability fields using only a select number of singular values are good approximations of the original permeability values. Predicted oil production profiles from reservoir simulation results using the reconstructed permeability matrices are also similar to those obtained using the original permeability values. However, current simulation results do not show any significant improvement in the predictive accuracy of oil production rates by applying SVD on the upscaled permeability values. It is speculated that the reconstructed permeability matrix has similar dimensionality before and after the SVD processing, thus the model accuracy and efficiency are not significantly improved.

Future research work with SVD could increase number of blocks, or layers in reservoir description to identify if it is justified for reservoir simulation models at larger scales. Different upscaling strategies could also be explored to select the best upscaling scheme, and the effectiveness of SVD on various upscaling schemes can be evaluated.

Appendix A. Supplemental Equations for Chapter 5

This appendix list equations that are used in the methodology part of Chapter 5. The reservoir simulator is based on block centered grid system using IMPES method (Chen, 2007). The IMPES pressure equation is shown in Equation 5.5, and saturation equation is shown in Equation 5.6.

$$-\frac{\partial}{\partial x_1}\left(k_{11}\lambda\frac{\partial p}{\partial x_1}\right)-\frac{\partial}{\partial x_2}\left(k_{22}\lambda\frac{\partial p}{\partial x_2}\right)-\frac{\partial}{\partial x_3}\left(k_{33}\lambda\frac{\partial p}{\partial x_3}\right)=\hat{q}-\frac{\partial}{\partial x_1}\left(k_{11(}\lambda_w\frac{\partial p_c}{\partial x_1}+\bar{\gamma}\frac{\partial z}{\partial x_1}\right))-$$

$$\frac{\partial}{\partial x_2}\left(k_{22(}\lambda_w\frac{\partial p_c}{\partial x_2}+\bar{\gamma}\frac{\partial z}{\partial x_2}\right))-\frac{\partial}{\partial x_3}\left(k_{33(}\lambda_w\frac{\partial p_c}{\partial x_3}+\bar{\gamma}\frac{\partial z}{\partial x_3}\right)) \tag{5.5}$$

$$\varphi=\frac{\partial}{\partial x_1}\left(\lambda_w k_{11}\left(\frac{\partial p_w}{\partial x_1}-\gamma_w\frac{\partial z}{\partial x_1}\right)\right)+\frac{\partial}{\partial x_2}\left(\lambda_w k_{22}\left(\frac{\partial p_w}{\partial x_2}-\gamma_w\frac{\partial z}{\partial x_2}\right)\right)+\frac{\partial}{\partial x_3}\left(\lambda_w k_{33}\left(\frac{\partial p_w}{\partial x_3}-\right.\right.$$

$$\left.\left.\gamma_w\frac{\partial z}{\partial x_3}\right)\right)+q_w \tag{5.6}$$

The pressure and saturation have spatial discretization form shown in Equation 5.7 and Equation 5.8. The pressure is solved fully implicitly using Gauss-elimination in Equation 5.7. The saturation for two-phase reservoir simulator is solved explicitly by using Equation 5.8.

$$-T^n_{1,i+\frac{1}{2},j,k}\left(p^n_{i+1,j,k}-p^n_{i,j,k}\right)+T^n_{1,i-\frac{1}{2},j,k}\left(p^n_{i+1,j,k}-p^n_{i,j,k}\right)-T^n_{1,i,j+\frac{1}{2},k}\left(p^n_{i,j+1,k}-\right.$$

$$\left.p^n_{i,j,k}\right)+T^n_{1,i,j-\frac{1}{2},k}\left(p^n_{i,j-1,k}-p^n_{i,j,k}\right)-T^n_{1,i,j,k+\frac{1}{2}}\left(p^n_{i,j,k+1}-p^n_{i,j,k}\right)+T^n_{1,i,j,k-\frac{1}{2}}\left(p^n_{i,j,k-1}-p^n_{i,j,k}\right)=$$

$$-T^n_{w,i+\frac{1}{2},j,k}\left(p^n_{c\,i+1,j,k}-p^n_{c\,i,j,k}\right)+T^n_{w,i+\frac{1}{2},j,k}\left(p^n_{c\,i,j,k}-p^n_{i-1,j,k}\right)-$$

$$T^n_{w,i,j+\frac{1}{2},k}\left(p^n_{c\,i,j+1,k}-p^n_{ci,j,k}\right)+T^n_{w,i+\frac{1}{2},j,k}\left(p^n_{c\,i,j,k}-p^n_{c\,i,j-1,k}\right)-$$

$$T^n_{w,i,j,k+\frac{1}{2}}\left(p^n_{c\,i,j,k}-p^n_{c\,i,j,k+1}\right)+T^n_{w,i,j,k-\frac{1}{2}}\left(p^n_{c\,i,j,k}-p^n_{c\,i,j,k-1}\right)-$$

$$(T\bar{\gamma})^n_{1,i+\frac{1}{2},j,k}\left(z^n_{i+1,j,k}-z^n_{i,j,k}\right)+(T\bar{\gamma})^n_{1,i-\frac{1}{2},j,k}\left(z^n_{i,j,k}-z^n_{i-1,j,k}\right)-$$

$$(T\bar{\gamma})^n_{1,i,j+\frac{1}{2},k}\left(z^n_{i,j+1,k}-z^n_{i,j,k}\right)+(T\bar{\gamma})^n_{1,i,j\pm,k}\left(z^n_{i,j,k}-z^n_{i,j-1,k}\right)-$$

$$(T\bar{\gamma})^n_{1,i,j,k+\frac{1}{2}}\left(z^n_{i,j,k+1}-z^n_{i,j,k}\right)+(T\bar{\gamma})^n_{1,i+\frac{1}{2},j,k}\left(z^n_{i,j,k}-z^n_{i,j,k-1}\right)+Q^n_{i,j,k} \tag{5.7}$$

$$\left(V\varphi\frac{S^{n+1}+S^{n}}{\Delta t}\right)=-T^{n}_{w,i+\frac{1}{2},j,k}\left(p^{n}_{c\,i+1,j,k}-p^{n}_{c\,i,j,k}\right)+T^{n}_{w,i+\frac{1}{2},j,k}\left(p^{n}_{c\,i,j,k}-p^{n}_{i-1,j,k}\right)-$$

$$T^{n}_{w,i,j+\frac{1}{2},k}\left(p^{n}_{c\,i,j+1,k}-p^{n}_{ci,j,k}\right)+T^{n}_{w,i+\frac{1}{2},j,k}\left(p^{n}_{c\,i,j,k}-p^{n}_{c\,i,j-1,k}\right)-$$

$$T^{n}_{w,i,j,k+\frac{1}{2}}\left(p^{n}_{c\,i,j,k}-p^{n}_{c\,i,j,k+1}\right)+^{n}_{w,i,j,k-\frac{1}{2}}\left(p^{n}_{c\,i,j,k}-p^{n}_{c\,i,j,k-1}\right)-$$

$$(T_w\bar{\gamma}_w)^{n}_{1,i+\frac{1}{2},j,k}\left(z^{n}_{i+1,j,k}-z^{n}_{i,j,k}\right)+(T_w\bar{\gamma}_w)^{n}_{1,i-\frac{1}{2},j,k}\left(z^{n}_{i,j,k}-z^{n}_{i-1,j,k}\right)-$$

$$(T_w\bar{\gamma}_w)^{n}_{1,i,j+\frac{1}{2},k}\left(z^{n}_{i,j+1,k}-z^{n}_{i,j,k}\right)+(T_w\bar{\gamma}_w)^{n}_{1,i,j\pm,k}\left(z^{n}_{i,j,k}-z^{n}_{i,j-1,k}\right)-$$

$$(T_w\bar{\gamma}_w)^{n}_{1,i,j,k+\frac{1}{2}}\left(z^{n}_{i,j,k+1}-z^{n}_{i,j,k}\right)+(T_w\bar{\gamma}_w)^{n}_{1,i+\frac{1}{2},j,k}\left(z^{n}_{i,j,k}-z^{n}_{i,j,k-1}\right)+Q^{n}_{wi,j,k} \qquad (5.8)$$

Appendix B. Description of Python Script for Chapter 2

The python script of this chapter is uploaded into a GitHub repository. The link to the GitHub repository can be found at https://github.com/mwzhouxu/Data-Driven-Modeling-and-Prediction-Using-Seismic-and-Petrophysical-Data-Analyses. Several python libraries were used in this chapter including scikit-learn, pandas, numpy, matplotlib, etc. The python scikit-learn library is used to create the data-driven models for this study. The script for creating the models are shown as follows.

```
from sklearn.ensemble import RandomForestClassifier
clf_rf = RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                        criterion='gini', max_depth=None, max_features='auto',
                        max_leaf_nodes=None, max_samples=None,
                        min_impurity_decrease=0.0, min_impurity_split=None,
                        min_samples_leaf=1, min_samples_split=2,
                        min_weight_fraction_leaf=0.0, n_estimators=10,
                        n_jobs=None, oob_score=False, random_state=None,
                        verbose=0, warm_start=False)

clf_svm = svm.SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)

clf_tree = tree.DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                        max_depth=None, max_features=None, max_leaf_nodes=None,
                        min_impurity_decrease=0.0, min_impurity_split=None,
                        min_samples_leaf=1, min_samples_split=2,
                        min_weight_fraction_leaf=0.0, presort='deprecated',
                        random_state=None, splitter='best')

from sklearn.neural_network import MLPClassifier
clf_nn = MLPClassifier(hidden_layer_sizes=[20, 15,10],
                    alpha=1,
                    max_iter=10000, tol=0.00005,
                    learning_rate_init=0.0005)

clf_rf.fit(X_train,y_train)
predicted_labels = clf_rf.predict(X_test)
```

Since this chapter is a classification task, a confusion matrix is calculated to visualize the relationship of the prediction results versus the actual results. The python scikit-learn library is used to create the confusion matrix. The normalized confusion matrix is created by normalizing each row of the confusion matrix by the total number of each class. After creating the confusion

matrix, the evaluation metrics including the precision, recall, and F-1 score are calculated using the following script.

```
from sklearn.metrics import plot_confusion_matrix
titles_options = [("Confusion matrix, without normalization", None),
                  ("Normalized confusion matrix", 'true')]

for title, normalize in titles_options:
    plt.grid(False)
    plt.figure(figsize=(10,15))
    disp = plot_confusion_matrix(clf, X_test, y_test,
                                 display_labels=("1","2","3","4","5","6","7","8","9"),
                                 cmap=plt.cm.Blues,
                                 normalize=normalize, values_format = '.1f')
    disp.ax_.set_title(title)
    print(title)
    print(disp.confusion_matrix)
plt.grid(False)

plt.show()
```

```
conf_matrix = confusion_matrix(y_test, clf_rf.predict(X_test))
display_cm(conf_matrix, facies_labels,
           display_metrics=True, hide_zeros=True)
```

Appendix C. Description of Python Script for Chapter 3

The python script for chapter 3 is uploaded to the same GitHub repository as Appendix B. The python TensorFlow library is used to create the artificial neural network model. Both the training and the testing data are normalized by their mean and standard deviation before being imported to the model. The dataset is split into a training and a testing set with 80% and 20% of the total sample points.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
data = pd.read_csv("seismic - well log data.csv")

# data split
train_dataset = data.sample(frac=0.8,random_state=0)
test_dataset = data.drop(train_dataset.index)
print(test_dataset)

# create training and teset dataset
train_stats = train_dataset.describe()
train_stats.pop("NPHI")
train_stats = train_stats.transpose()
train_stats

train_labels = train_dataset.pop('NPHI')
test_labels = test_dataset.pop('NPHI')
print("test labels are")
print(test_labels)

# define a normalization function
# normalize the training and testing data by the mean and std
def norm(x):
  return (x - train_stats['mean']) / train_stats['std']
normed_train_data = norm(train_dataset)
normed_test_data = norm(test_dataset)
```

The python script for the model is shown as follows. The neural network model includes four levels of hidden layers with additional three drop out layers for regularization. The loss function of the model is mean squared error and the evaluation metrics includes mean absolute error and mean squared error.

```
# create the model
def build_model():
  model = keras.Sequential([
    layers.Dense(20, activation=tf.nn.relu, input_shape=[len(train_dataset.keys())]),
    layers.Dense(15, activation=tf.nn.relu),
    layers.Dense(10, activation=tf.nn.relu),
    layers.Dense(5, activation=tf.nn.relu),
    layers.Dense(1)
  ])
  optimizer = tf.keras.optimizers.RMSprop(0.001)
  model.compile(loss='mean_squared_error',
                optimizer=optimizer,
                metrics=['mean_absolute_error', 'mean_squared_error'])
  return model

model = build_model()
model.summary()

example_batch = normed_train_data[:10]
example_result = model.predict(example_batch)
print(example_result)

# Display training progress by printing a single dot for each completed epoch
class PrintDot(keras.callbacks.Callback):
  def on_epoch_end(self, epoch, logs):
    if epoch % 100 == 0: print('')
    print('.', end='')
EPOCHS = 100
history = model.fit(
  normed_train_data, train_labels,
  epochs=EPOCHS, validation_split = 0.2, verbose=0,
  callbacks=[PrintDot()])
hist = pd.DataFrame(history.history)
hist['epoch'] = history.epoch
hist.tail()
```

During the training process, the training curves that show the change of mean absolute error and mean squared error at each epoch can be created. Since this is a regression model, the model predicts a numerical value for each sample. After the training is complete, the mean absolute error, mean squared error, and the coefficient of determination (R^2) can used to evaluate the model performance. The scripts are shown as follows.

```
def plot_history(history):
  hist = pd.DataFrame(history.history)
  hist['epoch'] = history.epoch

  plt.figure()
  plt.xlabel('Epoch')
  plt.ylabel('Mean Abs Error [NPHI]')
  plt.plot(hist['epoch'], hist['mean_absolute_error'],
           label='Train Error')
  plt.plot(hist['epoch'], hist['val_mean_absolute_error'],
           label='Val Error')
  plt.ylim([0, 0.1])
  plt.legend()

  plt.figure()
  plt.xlabel('Epoch')
  plt.ylabel('Mean Square Error [$NPHI^2$]')
  plt.plot(hist['epoch'], hist['mean_squared_error'],
           label='Train Error')
  plt.plot(hist['epoch'], hist['val_mean_squared_error'],
           label='Val Error')
  plt.ylim([0, 0.05])
  plt.legend()
  plt.show()
plot_history(history)
test_predictions = model.predict(normed_test_data).flatten()
plt.scatter(test_labels, test_predictions)
plt.xlabel('True Values [NPHI]')
plt.ylabel('Predictions [NPHI]')
plt.axis('equal')
plt.axis('square')
plt.xlim([0,plt.xlim()[1]])
plt.ylim([0,plt.ylim()[1]])
_ = plt.plot([-100, 100], [-100, 100])
plt.show()


loss, mae, mse = model.evaluate(normed_test_data, test_labels, verbose=0)
print("Testing set loss: {:5.4f} NPHI".format(loss))
print("Testing set Mean Absolute Error: {:5.4f} NPHI".format(mae))
print("Testing set Mean Squared Error: {:5.4f} NPHI".format(mse))
```

Appendix D. Description of Python Script for Chapter 4

The python script for chapter 4 is uploaded to the same GitHub repository as Appendix B. The python TensorFlow library is used to create the wavelet convolutional neural network model. Part of the script is accessed and modified from GitHub repositories of Patole (2019) and Menon (2019). The python scripts for importing the libraries and tools are shown as follows.

```
import numpy as np
from matplotlib import pyplot as plt
from keras import backend as K
from keras.models import Model
from keras.layers import Input, Dense, Conv2D, Lambda, Flatten,
from keras.layers import Reshape, Dropout, Activation
from keras.layers import MaximumPooling2D, AveragePooling2D, BatchNormalization
from keras.layers.merge import add, concatenate
from keras.preprocessing.image import ImageDataGenerator
from keras.utils import plot_model
from keras.layers import Conv2DTranspose
from keras.callbacks import EarlyStopping, ModelCheckpoint, ReduceLROnPlateau
import os
import sys
import random
import numpy as np
import cv2
import matplotlib.pyplot as plt
from glob import glob
from sklearn.model_selection import train_test_split
```

The script for the Wavelet CNN model is shown below. A U-net model is utilized as the base model. There are four levels of wavelet transformation that are performed to reduce the dimension of the output from the previous layer and save the boundary information from both high-pass and low-pass filters. For each level, there are four output components, which contains the approximation, horizontal, vertical, and diagonal component. The approximation component is similar to the original image. The other three components show edge features of the original image. For each 2D input image, the wavelet transformation is performed on both the x axis and y axis of the input data to perform a 2D wavelet transformation.

```
def wavelet_cnn():

    input_shape = 128, 128, 3

    modelinput = Input(input_shape, name='the_input')

#-----------------------------------------------------------------------------------------------------------------
    # Begin down sampling process

    c_0 = Conv2D(64, kernel_size=(3, 3), padding='same', name='c_0')(modelinput)
    n_0 = BatchNormalization(name='n_0')(c_0)
    r_0 = Activation('relu', name='r_0')(n_0)

    wavelet = Lambda(Wavelet, Wavelet_out_shape, name='wavelet')
    wavelet_l1, wavelet_l2, wavelet_l3, wavelet_l4 = wavelet(r_0)

#-----------------------------------------------------------------------------------------------------------------
    # first level wavelet transformation
    c_1 = Conv2D(64, kernel_size=(3, 3), padding='same', name='c_1')(wavelet_l1)
    n_1 = BatchNormalization(name='n_1')(c_1)
    r_1 = Activation('relu', name='r_1')(n_1)

    c_1_2 = Conv2D(64, kernel_size=(3, 3), strides=(2, 2), padding='same', name='c_1_2')(r_1)
    n_1_2 = BatchNormalization(name='n_1_2')(c_1_2)
    r_1_2 = Activation('relu', name='r_1_2')(n_1_2)

#-----------------------------------------------------------------------------------------------------------------
    # second level wavelet transformation
    c_a = Conv2D(filters=64, kernel_size=(3, 3), padding='same', name='c_a')(wavelet_l2)
    n_a = BatchNormalization(name='n_a')(c_a)
    r_a = Activation('relu', name='r_a')(n_a)
    concate_level_2 = concatenate([r_1_2, r_a])
    c_2 = Conv2D(128, kernel_size=(3, 3), padding='same', name='c_2')(concate_level_2)
    n_2 = BatchNormalization(name='n_2')(c_2)
    r_2 = Activation('relu', name='r_2')(n_2)

    c_2_2 = Conv2D(128, kernel_size=(3, 3), strides=(2, 2), padding='same', name='c_2_2')(r_2)
    n_2_2 = BatchNormalization(name='n_2_2')(c_2_2)
    r_2_2 = Activation('relu', name='r_2_2')(n_2_2)

#-----------------------------------------------------------------------------------------------------------------
    # third level wavelet transformation
    c_b = Conv2D(filters=64, kernel_size=(3, 3), padding='same', name='c_b')(wavelet_l3)
    n_b = BatchNormalization(name='n_b')(c_b)
    r_b = Activation('relu', name='r_b')(n_b)

    c_b_2 = Conv2D(128, kernel_size=(3, 3), padding='same', name='c_b_2')(r_b)
    n_b_2 = BatchNormalization(name='n_b_2')(c_b_2)
    r_b_2 = Activation('relu', name='r_b_2')(n_b_2)

    concate_level_3 = concatenate([r_2_2, r_b_2])
    c_3 = Conv2D(256, kernel_size=(3, 3), padding='same', name='c_3')(concate_level_3)
    n_3 = BatchNormalization(name='n_3')(c_3)
    r_3 = Activation('relu', name='r_3')(n_3)

    c_3_2 = Conv2D(256, kernel_size=(3, 3), strides=(2, 2), padding='same', name='c_3_2')(r_3)
    n_3_2 = BatchNormalization(name='n_3_2')(c_3_2)
    r_3_2 = Activation('relu', name='r_3_2')(n_3_2)
```

```
#-----------------------------------------------------------------------------------------------------------------
    # forth level wavelet transformation
    c_c = Conv2D(64, kernel_size=(3, 3), padding='same', name='c_c')(wavelet_l4)
    n_c = BatchNormalization(name='n_c')(c_c)
    r_c = Activation('relu', name='r_c')(n_c)

    c_c_2 = Conv2D(256, kernel_size=(3, 3), padding='same', name='c_c_2')(r_c)
    n_c_2 = BatchNormalization(name='n_c_2')(c_c_2)
    r_c_2 = Activation('relu', name='r_c_2')(n_c_2)

    c_c_3 = Conv2D(256, kernel_size=(3, 3), padding='same', name='c_c_3')(r_c_2)
    n_c_3 = BatchNormalization(name='n_c_3')(c_c_3)
    r_c_3 = Activation('relu', name='r_c_3')(n_c_3)

    concate_level_4 = concatenate([r_3_2, r_c_3])
    c_4 = Conv2D(256, kernel_size=(3, 3), padding='same', name='c_4')(concate_level_4)
    n_4 = BatchNormalization(name='n_4')(c_4)
    r_4 = Activation('relu', name='r_4')(n_4)

#-----------------------------------------------------------------------------------------------------------------

    c_5 = Conv2D(256, kernel_size = (3,3), kernel_initializer = 'he_normal', padding = 'same')(r_4)
    c_5 = BatchNormalization()(c_5)
    c_5 = Activation('relu')(c_5)

    c_5 = Conv2D(256, kernel_size = (3,3), kernel_initializer = 'he_normal', padding = 'same')(c_5)
    c_5 = BatchNormalization()(c_5)
    c_5 = Activation('relu')(c_5)

#-----------------------------------------------------------------------------------------------------------------
# Begin up sampling process

    u_6 = Conv2DTranspose(128, (3, 3), strides = (2, 2), padding = 'same')(c_5)
    u_6 = concatenate([u_6, c_3])
    u_6 = Dropout(0.1)(u_6)

    c_6 = Conv2D(128, kernel_size = (3,3), kernel_initializer = 'he_normal', padding = 'same')(u_6)
    c_6 = BatchNormalization()(c_6)
    c_6 = Activation('relu')(c_6)

    c_6 = Conv2D(128, kernel_size = (3,3), kernel_initializer = 'he_normal', padding = 'same')(c_6)
    c_6 = BatchNormalization()(c_6)
    c_6 = Activation('relu')(c_6)

#-----------------------------------------------------------------------------------------------------------------

    u_7 = Conv2DTranspose(64, (3, 3), strides = (2, 2), padding = 'same')(c_6)
    u_7 = concatenate([u_7, c_2])
    u_7 = Dropout(0.1)(u_7)

    c_7 = Conv2D(64, kernel_size = (3,3), kernel_initializer = 'he_normal', padding = 'same')(u_7)
    c_7 = BatchNormalization()(c_7)
    c_7 = Activation('relu')(c_7)

    c_7 = Conv2D(64, kernel_size = (3,3), kernel_initializer = 'he_normal', padding = 'same')(c_7)
    c_7 = BatchNormalization()(c_7)
    c_7 = Activation('relu')(c_7)
```

```
#-----------------------------------------------------------------------------------------------------

    u_8 = Conv2DTranspose(32, (3, 3), strides = (2, 2), padding = 'same')(c_7)
    u_8 = concatenate([u_8, c_1])
    u_8 = Dropout(0.1)(u_8)

    c_8 = Conv2D(32, kernel_size = (3,3), kernel_initializer = 'he_normal', padding = 'same')(u_8)
    c_8 = BatchNormalization()(c_8)
    c_8 = Activation('relu')(c_8)

    c_8 = Conv2D(32, kernel_size = (3,3), kernel_initializer = 'he_normal', padding = 'same')(c_8)
    c_8 = BatchNormalization()(c_8)
    c_8 = Activation('relu')(c_8)

#-----------------------------------------------------------------------------------------------------

    u_9 = Conv2DTranspose(16, (3, 3), strides = (2, 2), padding = 'same')(c_8)
    u_9 = concatenate([u_9, c_0])
    u_9 = Dropout(0.1)(u_9)

    c_9 = Conv2D(16, kernel_size = (3,3), kernel_initializer = 'he_normal', padding = 'same')(u_9)
    c_9 = BatchNormalization()(c_9)
    c_9 = Activation('relu')(c_9)

    c_9 = Conv2D(16, kernel_size = (3,3), kernel_initializer = 'he_normal', padding = 'same')(c_9)
    c_9 = BatchNormalization()(c_9)
    c_9 = Activation('relu')(c_9)

    outputs = Conv2D(1, (1, 1), activation='sigmoid')(c_9)

    model = Model(inputs=modelinput, outputs=outputs)
    model.summary()

    return model
```

The loading of the input and output data are shown in the following scripts. The dataset contains 4000 images and masks. The inputs of the model are 2D seismic images, which are displayed as grayscale images showing the subsurface features. The outputs are pre-interpreted salt body masks. In each mask, white color represents pre-interpreted salt bodies, and black color represents non-salt areas.

The original seismic images provided in the dataset are randomly cropped 101×101 small size seismic images from the original seismic survey. These images and masks are reshaped to be 128×128 before training the model.

```
# create model and set up model parameters
model = wavelet_cnn()
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# print number of images
ids = next(os.walk(r"./Dataset/Train/images"))[2]
print("No. of images = ", len(ids))

labels_folder = "./Dataset/Train/masks/"
images_folder = "./Dataset/train/images/"

# print location of data folder
print(labels_folder)
print(images_folder)

# load data and reshape
import glob
X = []
Y = []
for filename in glob.glob(images_folder + "*.PNG"):

    img = cv2.imread(filename)
    img = cv2.resize(img,(128,128))

    mask = cv2.imread(labels_folder + os.path.split(filename)[1])
    mask = cv2.cvtColor(mask, cv2.COLOR_BGR2GRAY)
    mask = cv2.resize(mask,(128,128))

    X.append(img)
    Y.append(mask)

X = np.array(X)
X = X/255
Y = np.array(Y)
Y = Y/255

# display shape of input and output data
Y = np.reshape(Y,(Y.shape[0], Y.shape[1], Y.shape[2],1))

print(X.shape)
print(Y.shape)
```

After loading the 4000 images and masks, the whole dataset is split into a training and testing set using 90% and 10% of the total image data. Parameters for early stopping is pre-set to stop the training process once the accuracy becomes stable. A dynamic learning rate is used to expedite the training process. The scripts for data splitting and training the model are shown in the below section.

After the model is trained, it is used to make predictions for salt bodies from new input seismic images. The original seismic images, original interpreted masks, and predicted masks can be compared to visualize the predicted results of the model.

```
# data spliting
X_train, X_valid, y_train, y_valid = train_test_split(X, Y, test_size=0.1, random_state=50)

# check size of the training and testing set
print(X_train.shape,y_train.shape)
print(X_valid.shape,y_valid.shape)

# set parameters for each stopping and dynamic learning reate
callbacks = [
    EarlyStopping(patience=10, verbose=1),
    ReduceLROnPlateau(factor=0.1, patience=5, min_lr=0.00001, verbose=1),
]

# set epoch numbers and batch size number, train the model
model.fit(X_train, y_train, batch_size=12, epochs=50, callbacks=callbacks,\
                    validation_data=(X_valid, y_valid))

# make redictions on the training and testing data
preds_train = model.predict(X_train, verbose=1)
preds_val = model.predict(X_valid, verbose=1)

# make threshold predictions
preds_train_t = (preds_train > 0.5).astype(np.uint8)
preds_val_t = (preds_val > 0.5).astype(np.uint8)
```

References

Abokhodair, A., 2008, Networks (Geophysical Inverse problem Example), accessed on 05/10/2019 from http://faculty.kfupm.edu.sa/ES/akwahab/.

Austin D., 2013, We Recommend a Singular Value Decomposition, accessed on Jan 10, 2020 from http://www.ams.org/publicoutreach/feature-column/fcarc-svd

Bangaru, S.S., Wang, C., Zhou, X., Jeon, H.W., Li, Y., 2020, Gesture Recognition based Smart Training Assistant System for Construction Worker Earplugs Wearing Training, Journal of Construction Engineering and Management. Manuscript Submitted.

Bekara, M., and Baan, M., (2007). "Local singular value decomposition for signal enhancement of seismic data." Geophysics, 72(2), V59-V65. https://doi.org/10.1190/1.2435967

Cao, L., 2006, Singular value decomposition applied to digital image processing, Division of Computing Studies, Arizona State University.

Chen, Y., 2018, Automatic micro seismic event picking via unsupervised machine learning, Geophysical Journal International, Volume 212, Issue 1, January 2018, Pages 88–102. https://doi.org/10.1093/gji/ggx420.

Chen, Y., He, C., Zhou, X., Yu, H., 2019, Analysis of Factors Affecting Drilling Friction and Investigation of Friction Reduction Tool in Horizontal Wells in Sichuan, Advances in Mechanical Engineering. Volume 11, Issue 7.

Chen, Z., 2007, Reservoir Simulation: Mathematical Techniques in Oil Recovery, Society for Industrial and Applied Mathematics.

Chopra, S., And Marfurt, K., 2007, Volumetric Curvature Attributes adding value to 3D seismic data interpretation: The Leading Edge, v. 26, p. 856-867.

Chris, L., 2019, Evaluating ML Models: Precision, Recall, F1 and Accuracy, accessed on 03/20/2020 from https://medium.com/analytics-vidhya/evaluating-ml-models-precision-recall-f1-and-accuracy-f734e9fcc0d3.

Christie, M. A. (1996, November 1). Upscaling for Reservoir Simulation. Society of Petroleum Engineers. doi:10.2118/37324-JPT

Cracknell, M.J., Reading, A.M., 2014. Geological mapping using remote sensing data: a comparison of five machine learning algorithms, their response to variations in the spatial distribution of training data and the use of explicit spatial information. Comput. Geosci. 63, 22–33.

CVPR workshop on autonomous driving, 2018, CVPR 2018 Workshop on Autonomous Driving (WAD) Video Segmentation Challenge, accessed on 04/21/2019 from https://www.kaggle.com/c/cvpr-2018-autonomous-driving.

Di, H., and D. Gao, 2017, 3D seismic flexure analysis for subsurface fault detection and fracture characterization: Pure and Applied Geophysics, 174, 747-761.

Dubois M., Bohling, G., Chakrabarti, S., 2007, Comparison of four approaches to a rock facies classification problem, Computers & Geosciences, Volume 33, Issue 5, May 2007, Pages 599-617.

Feng, Y., Shi, E., Luo, Y., Wang, B., Zhang, L. and Zhao, Y., 2019. Implementation of streamline simulation based on finite element method in FEniCS. Computational Geosciences, pp.1-15.

Figueiredo M., and Jain A. K., "Unsupervised learning of finite mixture models," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 3, pp. 381-396, March 2002.

Friedl, M.A., and Brodley, C.E., 1997, Decision tree classification of land cover from remotely sensed data, Remote Sensing of Environment, Volume 61, Issue 3, Pages 399-409, https://doi.org/10.1016/S0034-4257(97)00049-7.

Fujieda, S., Takayama, K., Hachisuka, T., 2017, Wavelet Convolutional Neural Networks for Texture Classification, arXiv:1707.07394.

Fujieda, S., Takayama, K., Hachisuka, T., 2018, Wavelet Convolutional Neural Networks, arXiv:1805.08620.

Fung, G. M., and Mangasarian O. L., 2005, Multicategory Proximal Support Vector Machine Classifiers, Machine Learning, 59, 77–97.

Gandhi, R., 2018, Support Vector Machine - Introduction to Machine Learning Algorithms, SVM model from scratch, accessed on 01/20/2020 from https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47.

Gibiansky, A., 2013, Cool Linear Algebra: Singular Value Decomposition, accessed on Jan 08, 2020 from http://andrew.gibiansky.com/blog/mathematics/cool-linear-algebra-singular-value-decomposition/.

Hampson, D. P., J. S. Schuelke, and J. A. Quirein (2001) Use of multiattribute transforms to predict log properties from seismic data, GEOPHYSICS 66(1):220.

Hegazy, T., AlRegib, G., 2014, Texture attributes for detecting salt bodies in seismic data. In: SEG Technical Program Expanded Abstracts, Society of Exploration Geophysicists. pp. 1455-1459.

Holdaway, K., 2014. Harness Oil and Gas Big Data with Analytics: Optimize Exploration and Production with Data Driven Models, Wiley Publishing.

Huang, C. and S. Chen (2019). "Stress Analysis of an Inclined Borehole Subjected to Fluid Discharge in Saturated Transversely Isotropic Rocks." International Journal of Geomechanics 19(11): 04019118.

Huang, C., Akbari, B., Chen, S.L. (2018). "Quick approximate elastoplastic solutions of wellbore stability problems based on numerical simulation and statistical analysis." Journal of Natural Gas Science and Engineering 51: 147-154.

Ildstad and Bormann, 2017, Complete tool for training & classifying facies on 3D SEGY seismic using deep neural networks, accessed from https://github.com/bolgebrygg/MalenoV.

Jain, A.K., Mao, K.M Mohiuddin, K.M., 1996, Artificial neural networks: a tutorial, Comput. IEEE, pp. 31-44.

Jafarpour B, McLaughlin DB, 2008, History-matching with an ensemble kalman filter and discrete cosine parameterization. Comput Geosci 12,2:227–244.

Jafarpour B., 2013, Sparsity-Promoting Solution of Subsurface Flow Model Calibration Inverse Problems. In: Mishra P., Kuhlman K. (eds) Advances in Hydrogeology. Springer, New York, NY

Jha S. K. and Yadava R. D. S., 2010, "Denoising by Singular Value Decomposition and Its Application to Electronic Nose Data Processing," in IEEE Sensors Journal, vol. 11, no. 1, pp. 35-44, Jan. 2011. doi: 10.1109/JSEN.2010.2049351

Jones, I.F., Davison, I., 2014, Seismic imaging in and around salt bodies. Interpretation, 2(4), SL1-SL20.

Kaggle: TGS salt identication challenge, 2018, https://www.kaggle.com/c/tgs-salt-identification-challenge, accessed: 2019-10-20.

Karchevskiy, M., Ashrapov, I., Kozinkin, L., 2018, Automatic salt deposits segmentation: A deep learning approach. arXiv Machine Learning

Khair, Abul, H., Cooke, D., Backe G., King R. Hand, M., Tingay, M., And Holford S., 2012, Subsurface Mapping of Natural Fracture Networks; A Major Challenge to be solved. Case Study from the Shale intervals in the Cooper Basin, South Australia: Proceedings, Thirty-Seventh Workshop on Geothermal Reservoir Engineering, Stanford, California.

Kim, Y., Hardisty, R., Torres, E., and Marfurt, K., 2018, Seismic-facies Classification Using Random Forest Algorithm, SEG Technical Program Expanded Abstracts : 2161-2165.

King, M. J. (2007, January 1). Upgridding and Upscaling: Current Trends and Future Directions. Society of Petroleum Engineers.

Kotsiantis, S.B., 2007. Supervised machine learning: a review of classification techniques. In: Proceedings of the 2007 Conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies. IOS Press, pp. 3–24.

Li, F., 2019, Convolutional Neural Networks for Visual Recognition, lecture notes, Stanford University, Accessed from http://cs231n.stanford.edu/ on 05/30/2019.

Li, P., and Zhang, Y., 2016, Facies Characterization of a Reservoir in the North Sea Using Machine Learning Techniques, Class Project Final Report for CS229, Stanford University.

Liaw, A., and Wiener, M., 2002, Classification and Regression by randomForest, R news, Vol. 2/3.

Liu, P., Zhang, H., Lian, W., and Zuo, W., 2019, Multi-level Wavelet Convolutional Neural Networks, arXiv:1907.03128.

Liu, X., J. Li, X. Chen, L. Zhou, and K. Guo, 2017, Bayesian discriminant analysis of lithofacies integrate the fisher transformation and the kernel function estimation: Interpretation, 5, no. 2. SE1-SE10.

Liu, X., Chen, X., Li, J., Zhou, X., Chen, Y., 2020. Facies identification based on multikernel relevance vector machine. IEEE Transactions on Geoscience and Remote Sensing. https://doi.org/10.1109/TGRS.2020.2981687.

Lo, S. B., Huai Li and M. T. Freedman, "Optimization of wavelet decomposition for image compression and feature preservation," in IEEE Transactions on Medical Imaging, vol. 22, no. 9, pp. 1141-1151, Sept. 2003, doi: 10.1109/TMI.2003.816953.

Lopez, D.L., Lorenzo, J.M., Zhou, X., 2020. Soil type Data Analytics Prediction Using Electrical Resistivity and S-wave Velocities for Shallow (<20 m) Unconsolidated Sediments. In: SEG Technical Program Expanded Abstracts 2020. Society of Exploration Geophysicists. Manuscript Submitted.

Luo, M., Gao, D., Zhao, X., Chen, Y., Yang, Y., Li, Z., Zhang, W., Zhou, X., 2020. Study on Influx Risk Evaluation of Horizontal Gas Wells with High-Pressure High-Temperature in the South China Sea. In ASME 2020 39th International Conference on Ocean, Offshore and Arctic Engineering, American Society of Mechanical Engineers.

Menon, M., H., 2019, https://github.com/menon92/WaveletCNN, accessed on 12/28/2019.

Mishra, S., & Lin, L. (2017, July 24). Application of Data Analytics for Production Optimization in Unconventional Reservoirs: A Critical Review. Unconventional Resources Technology Conference. doi:10.15530/URTEC-2017-2670157.

Pal, M., 2005, Random forest classifier for remote sensing classification, International Journal of Remote Sensing, volume 26, no. 1, page 217 to 222, DOI: 10.1080/01431160412331269698.

Patole, I., 2019, https://github.com/IsaacPatole/TGS-Salt-Identification-Challenge, accessed on 11/24/2019.

Pitas, I., Kotropoulos, C., 1992, A texture-based approach to the segmentation of seismic images, Pattern Recognition, Volume 25, Issue 9, September 1992, Pages 929-945, https://doi.org/10.1016/0031-3203(92)90059-R.

Porumb, M, Iadanza, E., Massaro, S., & Pecchia, L., 2020, A convolutional neural network approach to detect congestive heart failure. Biomedical Signal Processing and Control. 55. 101597. 10.1016/j.bspc.2019.101597.

Preux, C. (2016). About the Use of Quality Indicators to Reduce Information Loss When Performing Upscaling. Oil & Gas Science And Technology, Vol 71, Iss 1, P 7 (2016), (1), 7. doi:10.2516/ogst/2014023.

Qu, S., Z. Guan, E. Verschur, and Y. Chen, 2019, Automatic high-resolution micro-seismic event detection via supervised machine learning: Geophysical Journal International, 218, no. 3, 2106-2121.

Qureshi, M.A., Deriche, M., 2016, A new wavelet based efficient image compression algorithm using compressive sensing, Multimed Tools Appl, 75:6737–6754.

Robert, A., 2001, Curvature attributes and their application to 3D interpreted horizons, First Break, vol 19, issue .2.

Ronneberger, O., Fischer, P., Brox, T., 2015, U-Net: Convolutional Networks for Biomedical Image Segmentation, arXiv:1505.04597.

Roy, A., Jayaram, V., & Marfurt, K. J. (2013, January 1). Active Learning Algorithms in Seismic Facies Classification. Society of Exploration Geophysicists.

Roy, A.G., Navab, N., Wachinger, C., 2018, Concurrent spatial and channel squeeze & excitation in fully convolutional networks. In: International Conference on Medical Image Computing and Computer-Assisted Intervention.

Safavian, S. R., and Landgrebe, D., 1991, A survey of decision tree classifier methodology, in IEEE Transactions on Systems, Man, and Cybernetics, vol. 21, no. 3, pp. 660-674, May-June 1991.

Saha, S, 2018, A Comprehensive Guide to Convolutional Neural Networks - the ELI5 way, accessed on 01/10/2020 from https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

Salazar, M. O., & Villa Piamo, J. R. (2007, January 1). Permeability Upscaling Techniques for Reservoir Simulation. Society of Petroleum Engineers. doi:10.2118/106679-MS

Sebtosheikh, M.A., Salehi, A., 2015, Lithology prediction by support vector classifiers using inverted seismic attributes data and petrophysical logs as a new approach and investigation of training data set size effect on its performance in a heterogeneous carbonate reservoir. J. Petrol. Sci. Eng. 134, 143–149.

Schuetter, J., Mishra, S., Zhong, M., & LaFollette, R. (2015, July 20). Data Analytics for Production Optimization in Unconventional Reservoirs. Unconventional Resources Technology Conference. doi:10.15530/URTEC-2015-2167005

Schuetter, J., Mishra, S., Zhong, M., & LaFollette, R. (2018, August 1). A Data-Analytics Tutorial: Building Predictive Models for Oil Production in an Unconventional Shale Reservoir. Society of Petroleum Engineers. doi:10.2118/189969-PA.

SEG Tutorials, 2016, Society of Exploration Geophysicists, accessed on 11/30/2017 from https://github.com/seg.

Singh, A., 2012, Kashagan oil field, accessed on 01/15/2020, from https://www.slideshare.net/vikiitr/kashagan-oil-field.

SPE Tenth Comparative Solution Project, accessed on 01/15/2018, from https://www.spe.org/web/csp/datasets/set01.htm.

Suykens, J.A.K., and Vandewalle, J., 1999, Least Squares Support Vector Machine Classifiers, Neural Processing Letters 9: 293–300.

Teapot dome 3D survey, SEG Wiki, accessed on 04/01/2018 from https://wiki.seg.org/wiki/Teapot_dome_3D_survey

TensorFlow: accessed from https://www.tensorflow.org/ on 06/24/2019.

TensorFlow Core Tutorials, accessed from https://www.tensorflow.org/tutorials/keras/regression on 04/15/2019.

Tran, 2016, Precision, Recall, Sensitivity and Specificity, accessed on 04/01/2020 from https://newbiettn.github.io/2016/08/30/precision-recall-sensitivity-specificity/.

Udegbe, E., Morgan, E., & Srinivasan, S. (2018, September 24). Big Data Analytics for Seismic Fracture Identification, Using Amplitude-Based Statistics. Society of Petroleum Engineers. doi:10.2118/191668-MS

Waldeland, A.U., Jensen, A.C., Gelius, L.J., Solberg, A.H.S., 2018, Convolutional neural networks for automated seismic interpretation. The Leading Edge 37(7).

Wang, B., Feng, Y., Pieraccini, S., Scialò, S. and Fidelibus, C., 2019. Iterative coupling algorithms for large multidomain problems with the boundary element method. International Journal for Numerical Methods in Engineering, 117(1), pp.1-14.

Wang, G., Xie, X., Lai, J., Zhuo, J., 2018, Deep growing learning. In: 2017 IEEE International Conference on Computer Vision (ICCV).

Wang, B., Feng, Y., Du, J., Wang, Y., Wang, S. and Yang, R., 2018. An embedded grid-free approach for near-wellbore streamline simulation. SPE Journal, 23(02), pp.567-588.

Wang, S., and Chen, S. (2016, August 24). A Comprehensive Evaluation of Well Completion and Production Performance in Bakken Shale Using Data-Driven Approaches. Society of Petroleum Engineers. doi:10.2118/181803-MS.

Wang, W., Yang, F., Ma, J., 2018, Automatic salt detection with machine learning. In: 80th EAGE Conference and Exhibition 2018.

Wang, Z., Zhang, G., Zhou, X., Zhong, X., Li, X., Wang, X., Julaiti, S., 2012, Reservoir configuration analysis and its significance in oilfield development, Xinjiang Petroleum Geology, Volume 33, Issue 1, Page 61-64.

Wang, Ze, Mingzheng Yang, and Yuanhang Chen. 2019. "Numerical Modeling and Analysis of Induced Thermal Stress for a Non-Isothermal Wellbore Strengthening Process." Journal of Petroleum Science and Engineering 175 (September 2018): 173–83. https://doi.org/10.1016/j.petrol.2018.12.019.

Wei, Panfeng, Lihui Zheng, Mingzheng Yang, Chao Wang, Qifan Chang, Wang Zhang. 2020. "Fuzzy-ball Fluid Self-selective Profile Control for Enhanced Oil Recovery in Heterogeneous Reservoirs: The Techniques and the Mechanisms." Fuel 275 (February), https://doi.org/10.1016/j.fuel.2020.117959.

Williams, T., Li, R., 2018, Wavelet pooling for convolutional neural networks. ICLR 2018.

Wrona, T., Pan, I., Gawthorpe, R.L., Fossen, H., 2018, Seismic facies analysis using machine learning. Geophysics 83(5).

Wu, X., 2016, Methods to compute salt likelihoods and extract salt boundaries from 3d seismic images. Geophysics 81(6).

Xie, S., Girshick, R., Dollar, P., Tu, Z., He, K., 2017, Aggregated residual transformations for deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

Xie, Y., Zhu, C., Zhou, W., Li, Z., Liu, X., Tu, M., 2018, Evaluation of machine learning methods for formation lithology identification: A comparison of tuning processes and model performances, Journal of Petroleum Science and Engineering, Volume 160, 2018, Pages 182-193, https://doi.org/10.1016/j.petrol.2017.10.028.

Xiong, W., Z. Wan, M. Chen, and H. Zhang, 2010, Semi-automatic determination of the number of seismic facies in waveform classification: 72nd Annual International Meeting, EAGE, Expanded Abstracts, 3829.

Yang, Mingzheng, Mei-chun Li, Qinglin Wu, Frederick B. Growcock, and Yuanhang Chen. 2020. "Experimental Study of the Impact of Filter Cakes on the Evaluation of LCMs for Improved Lost Circulation Preventive Treatments." Journal of Petroleum Science and Engineering 191 (August), https://doi.org/10.1016/j.petrol.2020.107152.

Yang, Mingzheng, and Yuanhang Chen. 2020. "An Experimental Evaluation of the Effects of Filtercake in Wellbore Strengthening: Filtercake Rupture Resistance and Fracture Sealing Time." Journal of Energy Resources Technology 142 (4): 1–44. https://doi.org/10.1115/1.4044802.

Yang, Mingzheng, Yuanhang Chen, “Lost Circulation Material Implementation Strategies on Fluid Loss Remediation and Formation Damage Control”. SPE International Conference and Exhibition on Formation Damage Control, Lafayette, Louisiana, USA, 19-21 Feb 2020, https://doi:10.2118/199249-MS.

Yin, Q., Yang, J., Hou, X., Tyagi, M., Zhou, X., Cao, B., Sun, T., Li, L., Xu, D., 2020, Drilling Performance Improvement in Offshore Batch Wells Based on Rig State Classification Using Machine Learning, Journal of Petroleum Science and Engineering. Volume 192. https://doi.org/10.1016/j.petrol.2020.107306.

Yiu, T., 2019, Understanding Random Forest, Towards Data Science, accessed on 01/21/2020 from https://towardsdatascience.com/understanding-random-forest-58381e0602d2.

Yu, Z., Lu, Y., Zhang, J., You, J., Wong, H.S., Wang, Y., Han, G., 2018, Progressive semi supervised learning of multiple classifiers. IEEE transactions on cybernetics 48(2). DOI: 10.1109/TCYB.2017.2651114.

Zhang, C., Frogner, C., Araya-Polo M., and Hohl, D., 2014, Machine-learning Based Automated Fault Detection in Seismic Traces, 76th EAGE Conference and Exhibition 2014.

Zhang, G., Wang, Z., Chen, Y., 2018, Deep learning for seismic lithology prediction, Geophysical Journal International, Volume 215, Issue 2. Pages 1368–1387.

Zhao, J., Jiang, Y., Li, Y., Zhou, X., Wang, R., 2018, Modeling Fractures and Barriers as Interfaces for Porous Flow with Extended Finite-Element Method, Journal of Hydrologic Engineering. Volume 23, Issue 7.

Zhao, T., Jayaram, V., Marfurt, K. J., & Zhou, H. (2014, October 29). Lithofacies Classification in Barnett Shale Using Proximal Support Vector Machines. Society of Exploration Geophysicists.

Zheng, Z., P. Kavousi, and H. Di, 2014, Multi-attribute and Neural Network-based fault detection in 3D seismic interpretation: Advanced Materials Research, 838-841, 1497-1502

Zhou, X., Dawers, N.H., Amer, R.M., 2014, Fault Population Analyses in the Eastern California Shear Zone: Insights into the Development of Young, Actively Evolving Plate Boundary Structures, AGU Fall Meeting Abstracts 2014.

Zhou, X., Taleghani, A.D., Choi, J.W., 2017, Imaging Three-Dimensional Hydraulic Fractures in Horizontal Wells Using Functionally-Graded Electromagnetic Contrasting Proppants. Presented at the Unconventional Resources Technology Conference, Austin, Texas.

Zhou, X., Tyagi, M., Zhang, G., Yu, H., Chen, Y., 2019, September 23. Data Driven Modeling and Prediction for Reservoir Characterization Using Seismic Attribute Analyses and Big Data Analytics. Society of Petroleum Engineers. https://doi.org/10.2118/195856-MS.

Zhou, X., Tyagi, M., 2020. Evaluation of Singular Value Decomposition (SVD) Enhanced Upscaling in Reservoir Simulation. In ASME 2020 39th International Conference on Ocean, Offshore and Arctic Engineering, American Society of Mechanical Engineers.

Zhou, X., Tyagi, M., Sharma, J., 2020, Enhanced Automatic Segmentation of Salt Bodies from Seismic Images Using Wavelet Convolutional Neural Networks, in 82nd EAGE Conference and Exhibition 2020.

Zhu, X.J., 2005, Semi-supervised learning literature survey. Tech. rep., University of Wisconsin-Madison Department of Computer Sciences.

Zu, S., H. Zhou, R. Wu, W. Mao, and Y. Chen, 2018, Hybrid-sparsity constrained dictionary learning for iterative deblending of extremely noisy simultaneous-source data: IEEE Transaction on Geosciences and Remote Sensing, 57, no. 4, 2249.

Zupan, J., Gasteiger, J., 1993, Neural Networks For Chemists: An Introduction, VCH, New York.

Vita

Xu Zhou was born in Songzi, Hubei, China. He received a bachelor's degree in Geological Engineering from China University of Petroleum (Beijing) in 2012. After that, he earned a master's degree in Earth and Environmental Science from Tulane University in New Orleans, Louisiana in 2015. As his interest in Petroleum Engineering grew, he decided to enter the Department of Petroleum Engineering at Louisiana State University. His research interests include reservoir modeling, seismic and petrophysical data analyses and machine learning. He intends to work as a reservoir engineer after completion of his Ph.D. degree.