UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

MACHINE LEARNING FOR THE SUBSURFACE CHARACTERIZATION AT CORE, WELL, AND RESERVOIR SCALES

A DISSERTATION

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

DOCTOR OF PHILOSOPHY

By

HAO LI
Norman, Oklahoma
2020

MACHINE LEARNING FOR THE SUBSURFACE CHARACTERIZATION AT CORE, WELL, AND RESERVOIR SCALES

A DISSERTATION APPROVED FOR THE
MEWBOURNE SCHOOL OF PETROLEUM AND GEOLOGICAL ENGINEERING

BY THE COMMITTEE CONSISTING OF

Dr. Deepak Devegowda, Chair

Dr. Siddharth Misra, Co-Chair

Dr. Amy McGovern

Dr. Xingru Wu

Dr. Rouzbeh G. Moghanloo

DEDICATION

I dedicate this dissertation to my beloved parents, Tao Li and Jianying Zhang. Thank you for your support and unconditional love.

Acknowledgements

I would like to express my gratitude to Dr. Siddharth Misra. His wisdom and knowledge guide me to overcome the obstacles in the research, give me the courage to explore new areas and learn new technologies, inspire me with new ideas that no one has ever tried. He introduced me to the area of machine learning, which will also be my goal of career after graduation. His optimism and encouragement will be my treasure of life. This research is not possible without his support and supervision.

I would like to thank Dr. Deepak Devegowda for his supervision during the last year of my Ph.D. research. Dr. Deepak Devegowda guided me on the improvement of the automatic history matching task using reinforcement learning. His knowledge in reservoir engineering guided me on improving the research by generating a high-quality reservoir simulation environment. His valuable feedback and supervision helped me finished the last phase of my Ph.D. research.

I would like to thank my committee members Dr. Amy McGovern, Dr. Xingru Wu, and Dr. Rouzbeh G. Moghanloo. Thank you for reviewing my dissertation and provide valuable feedback. I would like to thank Dr. Amy McGovern for her fantastic machine learning class, which helped me a lot in both research and job seeking.

I would also like to thank Mr. Jiabo He, Mr. Sangcheol Yoon, and Mr. Yifu Han for helping me improve my research.

Finally, I would like to thank my family for their unconditional support. I would like to thank all my friends who helped me, trusted me, and supported me during the last four years.

Few of the research findings in the dissertation are based upon work supported by the U.S. Department of Energy, Office of Science, Office of Basic Energy Sciences, Chemical Sciences Geosciences, and Biosciences Division, under Award Number DE-SC-00019266.

Table of Contents

List of Tables

List of Figures

Abstract

The development of machine learning techniques and the digitization of the subsurface geophysical/petrophysical measurements provides a new opportunity for the industries focusing on exploration and extraction of subsurface earth resources, such as oil, gas, coal, geothermal energy, mining, and sequestration. With more data and more computation power, the traditional methods for subsurface characterization and engineering that are adopted by these industries can be automized and improved. New phenomenon can be discovered, and new understandings may be acquired from the analysis of big data. The studies conducted in this dissertation explore the possibility of applying machine learning to improve the characterization of geological materials and geomaterials.

Accurate characterization of subsurface hydrocarbon reservoirs is essential for economical oil and gas reservoir development. The characterization of reservoir formation requires the integration interpretation of data from different sources. Large-scale seismic measurements, intermediate-scale well logging measurements, and small-scale core sample measurements help engineers understand the characteristics of the hydrocarbon reservoirs. Seismic data acquisition is expensive and core samples are sparse and have limited volume. Consequently, well log acquisition provides essential information that improves seismic analysis and core analysis. However, the well logging data may be missing due to financial or operational challenges or may be contaminated due to complex downhole environment. At the near-wellbore scale, I solve the data constraint problem in the reservoir characterization by applying machine learning models to generate synthetic sonic traveltime and NMR logs that are crucial for geomechanical

and pore-scale characterization, respectively. At the core scale, I solve the problems in fracture characterization by processing the multipoint sonic wave propagation measurements using machine learning to characterize the dispersion, orientation, and distribution of cracks embedded in material. At reservoir scale, I utilize reinforcement learning models to achieve automatic history matching by using a fast-marching-based reservoir simulator to estimate reservoir permeability that controls pressure transient response of the well.

The application of machine learning provides new insights into traditional subsurface characterization techniques. First, by applying shallow and deep machine learning models, sonic logs and NMR T2 logs can be acquired from other easy-to-acquire well logs with high accuracy. Second, the development of the sonic wave propagation simulator enables the characterization of crack-bearing materials with the simple wavefront arrival times. Third, the combination of reinforcement learning algorithms and encapsulated reservoir simulation provides a possible solution for automatic history matching.

Chapter 1 Introduction

1.1 Research Motivations

Machine learning has shown great potential in the area of image processing, text processing, and digital marketing. Machine learning and deep learning have been successfully applied to solve problems like image recognition, language translation, self-driving cars, and content recommendation. The research performed in this dissertation applied different machine learning techniques to improve the subsurface characterization at different scales.

Different technologies and tools are developed for the purpose of reservoir characterization. Well logs are utilized to acquire the subsurface near-wellbore reservoir information, laboratory core sample analysis is used to measure the mechanical or petrophysical properties, and history matching techniques are developed to estimate the reservoir-scale properties. Exploratory studies are conducted in this dissertation to improve the synthesis of pseudo-well logs, core-scale fracture characterization, and automatic history matching by using machine learning algorithms.

With the success of machine learning in the computer science area, different studies have tried to apply machine learning models in the oil and gas industry to help improve subsurface characterization, generate pseudo petrophysical data, improve the characterization of crack-bearing materials, and improve the results of history matching. The following sections give a brief introduction of existing machine learning applications in the oil and gas industry.

1.2 Machine Learning Applications in Subsurface Characterization

Machine learning models can be taken as a series of transforms of data from one space to another space from a geometric point of view. For the task of subsurface characterization, raw measurements (e.g. acoustic, electrical, nuclear, and magnetic) are transformed with physical or empirical equations to reservoir properties (e.g. porosity and permeability). Many studies have shown promising results of applying machine learning models to improve subsurface characterization. A lot of efforts have been made to use machine learning models to automatically perform tasks of well logs interpretation or synthetic logs generation. The application of machine learning models is facilitated by the digitization process in the oil and gas industry.

Some studies applied machine learning models to automatically interpret well logs (Shao et al., 2019; Wu et al., 2018). Well logs are interpreted using neural networks and data inversion techniques to obtain subsurface physical properties. For example, Wong et al. (Wong et al., 1995) classified well log data into different lithofacies followed by the estimation of porosity and permeability using genetic neural networks. Similarly, lithology determination from well logs was performed by Chang et al. (Chang et al., 1997) in Ordovician rock units in northern Kansas using fuzzy associative memory neural network.

Others use machine learning models to generate synthetic logs (Akinnikawe et al., 2018). These tasks usually involve a simple regression problem. The artificial neural network model is a popular selection for these tasks due to its flexibility. For example, Elkatatny et al. applied artificial neural networks, etc. to predict both compressional and shear sonic time from GR, bulk density, and neutron porosity (Elkatatny et al., 2016).

1.3 Machine Learning Applications in Characterization of Crack-bearing Materials

Machine learning models can be used to predict the propagation direction of fractures, classify rock fracture types, or reduce the computation time in crack-bearing material simulation tasks. Some studies have applied artificial neural networks to investigate the crack evolution process. Moore et al. (Moore et al., 2018) applied ANN to predict fracture growth from simulation data. The authors gathered data from the Finite-Discrete Element model. The study aims to predict whether two fractures will coalesce or not based on the parameters of fracture orientations, distances between two fractures, the minimum distance from one of the fractures to the nearest boundary, etc. Another popular method used is SVM. The study from Zhou et al. (Zhou et al., 2018) applied ANN and SVM to classify rock fracture and blast events. Farhidzadeh et al. (Farhidzadeh et al., 2014) used SVM to classify the acoustic emission (AE) signals acquired from experiments. They extracted different features like aptitude, duration, etc. from the AE signal, and used the extracted features to predict whether the fracture is a tensile fracture or shear fracture.

Other researchers applied convolutional neural networks (CNN) to process the measurements during fracture propagation. Miller et al. (Miller et al., 2017) applied CNN to process 2D images acquired from rock fracturing simulation. The 2D fracture image is processed as a weighted graph. CNN is used to learn the graph features that are most predictive of final fracture length distribution. Researchers from Harvard and MIT applied CNN to process seismic data (Perol et al., 2018).

1.4 Machine Learning Applications in History Matching and Reservoir Modeling

History matching integrates multiple types of data to build a reliable reservoir model capable of predicting the future performance of a reservoir. Spatial data of geoscience (seismic), geophysics (well log and core data), as well as dynamic data (production rate and pressure), are combined to build, calibrate, and test the validity of the reservoir model (Avansi and Schiozer, 2015).

A large number of techniques are developed to facilitate the matching procedure. Proxy models and the Fast Marching algorithm are applied to reduce the computational resources needed for simulation. Based on the matching algorithm applied, history matching can be identified as deterministic (gradient-based), stochastic or probabilistic (Kazemi and Stephen, 2012). Gradient-based optimization algorithms like Gauss-Newton (GN) and the Levenberg-Marquardt (LM) algorithms, follow the gradient calculated by taking derivative of reservoir parameters to find a local minimum in the objective function plane. The gradient-based algorithms have the advantage of high efficiency, but they are not suitable for problems where the number of model parameters is large (Shirangi and Emerick, 2016).

Compared to deterministic optimization algorithms, stochastic algorithms include randomness in the exploration of the parameter space and thus may enable the algorithm to escape a local minimum and search a larger area in the parameter space. Stochastic algorithms of the Genetic Algorithm (GA) and Particle Swarm Optimization (PSO) have been found efficient at dealing with history matching problems where large parameters are involved (Sanghyun and Stephen, 2018).

Probabilistic algorithms like Ensemble Kalman Filter (EnKF), Maximum Likelihood, etc. are within the Bayesian framework (Vink et al., 2015). Instead of taking history matching as a process of finding the best match, probabilistic algorithms aim to infer the set of models with the possible combination of parameters from existing information. EnKF is a gradient-free algorithm, which relies on the initial ensemble to generates multiple history-matched models (Ding et al., 2016). EnKF method is a Monte-Carlo implementation of the Bayesian update, where probability density function (PDF) is updated when new data is available (Mandel, 2009).

1.5 Organization of the Dissertation

Chapter 1 gives a brief introduction to existing machine learning applications in the oil and gas industry. The motivations of the research and different machine learning applications in subsurface characterization, in crack-bearing materials characterization, and in history matching are discussed.

Chapter 2 applies shallow learning machine learning models to generate synthetic sonic traveltime logs. Sonic traveltime logs are important to evaluate the reservoir geomechanical characteristics. 6 shallow learning models are applied and compared. Well log data from two oil wells are used. Model testing is performed using cross-well data to estimate real-world performance.

Chapter 3 aims to apply both shallow and deep learning models to generate synthetic NMR T2 logs. The NMR T2 log is important for pore-related parameter characterization. In this chapter, two types of input logs are used and compared as the input of the models. R^2, mean squared error, and bending energy are used as metrics to

evaluate the models from different aspects. The study shows promising results to generate synthetic NMR T2 log from other well logs.

Chapter 4 developes a noninvasive material characterization method by measuring the sonic wavefront arrival times using simulation. The experiments performed in chapter 4 generated 2D crack-bearing rock samples with different types of fracture systems. The 2D rock samples and the experiment configurations are inspired by real-world experiments. Fast marching algorithms are used to simulate the sonic wave propagation process inside the rock samples. 28 sensors are embedded around the samples to measure the sonic wavefront arrival times. 9 selected machine learning models are trained to identify the fracture system types with the sonic wavefront arrival times.

In chapter 5, two reinforcement learning algorithms are applied to achieve automatic history matching. A reservoir simulator using the fast marching algorithm is encapsulated in an interactable environment. The reinforcement learning algorithms use the sequential decision-making process to adjust the initial reservoir permeability to gain higher reward. The learning is based on interaction history between the environment and reinforcement models. Principles of existing history matching methods and the proposed history matching method using reinforcement learning algorithms are explained and compared in detail.

Chapter 6 presents the conclusions and recommendations for future work.

Chapter 2 Data-Driven Synthesis of Compressional and Shear Travel Times for Near-Wellbore Geomechanical Characterization

2.1 Background

Compressional and shear travel time logs (DTC and DTS) acquired using sonic logging tools are crucial for subsurface geomechanical characterization. In this study, 8 'easy-to-acquire' conventional logs were processed using 6 shallow regression-type supervised-learning models, namely ordinary least squares (OLS), partial least squares (PLS), ElasticNet (EN), LASSO, multivariate adaptive regression splines (MARS), and artificial neural network (ANN), to synthesize DTC and DTS logs. Among the 6 models, MARS outperforms other models with R2 of 0.63 and 0.59 for DTC and DTS logs when deployed on a new oil well in blind test, respectively. The 6 shallow learning models are trained and tested with 8481 data points acquired from a 4240-feet depth interval of a shale reservoir in Well 1, and the trained models are deployed in Well 2 for purposes of blind testing against 2920 data points from 1460-feet depth interval. Apart from the 6 shallow learning models, two widely used empirical models are implemented and compared with the shallow learning models. To investigate the depth dependency phenomenon in well logs, ANN models are built to predict DTC and DTS logs with input logs from three consecutive depths. The results indicate that ANN models are not as stable as other shallow learning models. Including depth dependencies in the synthesis of sonic logs does not improve the prediction accuracy.

Sonic logging tools transmit compressional and shear waves through the reservoir to capture the mechanical properties of reservoir matrix and fluid. Compressional waves travel through both the rock matrix and fluid, while shear waves only travel through the

matrix. The wave travel time depends on the elastic properties and moduli of the rock as well as the composition and microstructure of the reservoir. Compressional and shear travel time logs (DTC and DTS) can be computed from the waveforms recorded at the receiver. The difference and variation in the DTC and DTS contain information about the reservoir porosity, rock brittleness, and Young's modulus, to name a few. However, sonic logs may not be always available due to financial or operational constraints. Many empirical models have been developed in the oil and gas industry to predict the sonic logs from other well logs. This dissertation aims to develop a workflow to synthesize both DTC and DTS logs from 'easy-to-acquire' conventional well logs using simple machine learning models.

Well logging is essential for the oil and gas industry to understand the in-situ subsurface petrophysical and geomechanical properties (Alexeyev et al., 2017; Wang et al., 2019). Some researchers have tried to generate synthetic logs to improve reservoir characterization in case of missing or erroneous well logs. Certain well logs, like gamma-ray (GR), resistivity, density, and neutron, are considered as 'easy-to-acquire' conventional well logs and deployed in most of the wells. Other well logs, like nuclear magnetic resonance (NMR), dielectric dispersion, elemental spectroscopy, and sonic, are deployed in a limited number of wells. Easy-to-acquire well logs can be processed using statistical and machine learning methods to synthesize the well logs that are not frequently acquired in each well. Researchers have explored the possibility of synthesizing certain 'hard-to-acquire' well logs under data constraint (Li and Misra, 2017a; Li and Misra, 2017b; Tariq et al., 2016).

2.2 Motivation for this Work

Machine learning algorithms, like ANN and fuzzy logic, are widely applied in prediction tasks in the oil and gas industry. The performance of these models can be easily adjusted according to the complexity of the problem. Several studies have implemented machine learning techniques to determine the sonic log from other well logs. ANNs, Adaptive data-driven Inference System (ANFIS), and Support Vector Machines (SVM) were used to predict both compressional and shear sonic travel time from GR, bulk density, and neutron porosity (Elkatatny et al., 2016). The study achieved a correlation coefficient of 0.99 when tested on field data. In another study, shear wave velocity (reciprocal of DTS) is predicted using the intelligent system, which combined the algorithms of fuzzy logic, neuro-fuzzy, and ANNs. The mean squared error during the testing stage was around 0.05 (Rezaee et al., 2007). The models show stable performance, but the model can only predict the shear wave. A similar study applied a committee machine with intelligent systems to predict sonic travel time from conventional well logs (Asoodeh and Bagheripour, 2012). Onalo et. al. applied data-driven models for the purpose of sonic logs prediction (Onalo et al., 2018; Onalo et al., 2019). Apart from machine learning algorithms, other studies predicted DTS or DTC using empirical equations (Greenberg and Castagna, 1992a; Iverson and Walker, 1988), empirical correlations (Maleki et al., 2014), or self-developed model (Keys and Xu, 2002). Other researchers tried to predict DTS and DTC in thin beds using petrophysical properties instead of raw conventional logs (Baines et al., 2008).

Sonic logs are important to evaluate the reservoir mechanical properties. However, sonic logs may be not accessible, or the sonic logs data is contaminated for

some oil wells. Under this circumstance, synthesizing sonic logs from other well logs may greatly help engineers to characterize the reservoir mechanical properties. Different machine learning models and simple empirical models have been developed for the purpose of sonic logs prediction. This dissertation focuses on the possibility of predicting sonic logs with simple machine learning models and 'easy-to-acquire' well logs. Strict data preprocessing and model training and testing procedures are implemented. The models are evaluated with different metrics on two different testing datasets with different data distributions. Moreover, the stability of the models is also investigated. A good model should not only with good prediction accuracy but also with good stability when exposed to a new dataset. Lastly, the depth dependency problem is also investigated. The results show that simple models may outperform complex models for the sonic logs prediction tasks.

2.2.1 Brief Introduction of Workflow

For the purposes of synthesizing DTC and DTS logs from conventional logs, this dissertation aims to test the performances of shallow learning models. A complex predictive model with many parameters is not necessary for the regression problem we are investigating in this study. In this study, the inputs and outputs of the problem are relatively simple and have low dimensionality. Moreover, shallow learning models need fewer data and tend not to overfit. Shallow learning models have the following two advantages: (1) they need less computational resources and hyperparameter-tuning time, and (2) they are easy to deploy and interpret.

The 8 'easy-to-acquire' conventional logs together with DTC and DTS logs acquired in two wells are used for this study. The dataset is acquired from two oil wells

within the same reservoir. The dataset in well #1 is used for the purpose of model training, validation, and testing. The dataset in well #2 is used to test the models' performance when exposed to a new dataset in blind test. The 6 shallow learning models implemented in this dissertation are Ordinary Least Squares (OLS), Partial Least Squares (PLS), Least Absolute Shrinkage and Selection Operator (LASSO), ElasticNet model, Multivariate Adaptive Regression Splines (MARS), and ANN. The first four models are considered as linear models and the last two are nonlinear models. The regression capacity of the last model, ANN, can be adjusted based on the complexity of the problem. By comparing the performance of the 6 models with increasing regression capacity, we can identify if shallow models are adequate for the sonic log synthesis.

Apart from the 6 shallow learning models, two empirical models are implemented as a baseline to compare with the 6 shallow learning models. The results indicate that the shallow learning models outperform empirical models in both DTC and DTS prediction. To investigate the depth dependency in well log synthesis, a new ANN model is implemented to predict a DTC or DTS from three consecutive depths. The training and testing procedure are the same as the other 6 shallow learning models. The train and test dataset of the new ANN model is different.

Prediction performance of the six regression-type models can be easily evaluated on the train/test dataset by using metrics such as mean squared error and correlation coefficient. Four evaluation metrics are used to evaluate the models' prediction performance. Apart from the prediction performance, the models' stability is also investigated using the bootstrap resampling method. The models' stability measures the stability of the models' prediction performance when exposed to a new dataset.

2.3 Key Fundamental Questions to Be Answered

- Can simple models developed using machine learning techniques generate sonic logs by processing easy-to-acquire well logs?
- Do simple machine-learning models outperform empirical models in predicting/synthesizing the sonic logs?
- Which model is most suitable for predicting/synthesizing the sonic logs prediction task? Why is the model most suitable?
- How to investigate the stability of a machine learning model for log synthesis? Which model has the best prediction accuracy as well as model stability?
- Does including depth dependencies and data from multiple depths improve the prediction of sonic logs?
- How well does a machine learning model for log synthesis perform on cross-well dataset? A cross-well dataset is acquired from another oil well in the same reservoir.
- What are the characteristics of the shallow learning models compared with complex machine learning models? What is the advantage of shallow learning models?

2.4 Data Preparation and Preprocessing

2.4.1 Data Description and Preparation

Well log data used in this dissertation was acquired from two wells. In Well #1, 8481 well log data points were acquired from a 4240-feet depth interval. In Well #2, 2920 data points were acquired from 1460-feet depth intervals. Gamma Ray log (GR), caliper log (DCAL), density porosity log (DPHZ), neutron porosity log (NPOR),

photoelectric factor log (PEFZ), bulk density log (RHOZ), lithology type, and laterolog resistivity logs at 6 depths of investigation (RLA0, RLA1, RLA2, RLA3, RLA4, RLA5) are selected as the 13 easy-to-acquire conventional logs (as shown in Figure 2-1, Tracks 2-6) that are preprocessed and fed into the six predictive models.

Correlation between different well logs is a common phenomenon in the well logging measurements. Correlated features may affect the machine learning models' performance. For a specific problem, a simpler model is always preferred. Including correlated features may result in more complex models that overfit. In this study, I preprocessed the raw data to remove outliers and highly correlated features. The isolation forest algorithm is used to remove samples that are considered outliers. Pearson correlation and spearman correlation indices are used to remove highly correlated features. The well logs in well #1 are used for model training, validation, and testing. The well logs in well #2 are used to cross-well testing.

Single-arm caliper indicates the hole size, which can also be attributed to drilling practices, mud parameters (density, composition) and need not be always linked to rock mechanics. For formations susceptible to breakouts such as shale, variations in DCAL can serve as an indicator of the formation type, which can then be related to the geomechanical properties that govern the DTC and DTS logs. Sonic logs are shallow sensing logs as compared to resistivity and density logs. Consequently, DCAL can serve as an indicator of the effects of borehole size on the sonic logs. Machine learning models are suitable to integrate disparate datatypes to generate outputs that are beyond simple physics-based models and human perceptions. Consequently, I use the commonly available DCAL as an input in the proposed log synthesis.

DPHZ, the density porosity is derived from RHOB with assumptions for matrix density and fluid density. RHOB is a direct measurement and DPHZ is an estimation. I use both DPHZ and RHOB logs despite their correlation because their interplay indicates fluid and matrix properties and their variations along the well length. Having both DPHZ and NPOR assists the shallow learning models used in this study. Shallow learning models need feature engineering, and DPHZ can be considered as an engineered feature to assist the shallow-learning models.

Formations with different lithologies exhibit different petrophysical and mechanical properties, which in turn affect the sonic wave propagation. A geology expert identified 13 broad lithology types in the 4240-feet depth interval and assigned specific sections of the depth interval with discrete-valued lithology types ranging from 1 to 13 to indicate the formation lithology. DTC and DTS logs are the outputs of the models (as shown in Figure 2-1, Track 7).

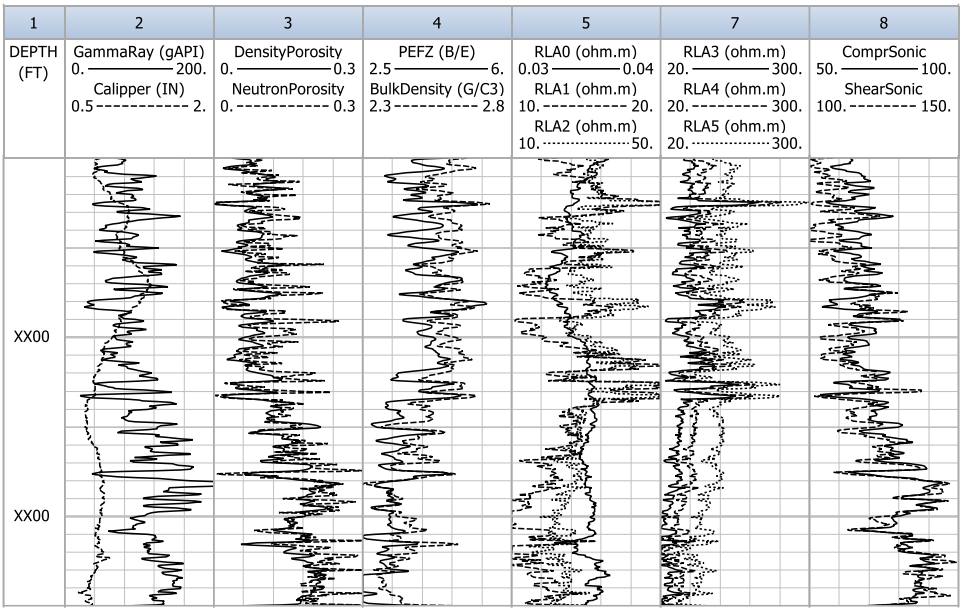


Figure 2-1. Track 1 is depth, Track 2 contains gamma-ray and caliper logs, Track 3 contains density porosity and neutron porosity logs, Track 4 contains formation photoelectric factor and bulk density logs, Track 5 is laterolog resistivity logs at shallow

depths of investigation (RLA0, RLA1, RLA2), Track 6 is laterolog resistivity logs at deep depths of investigation (RLA3, RLA4, RLA5), and Track 7 contains DTC and DTS logs for a 200-feet section of the formation.

2.4.2 Data Preprocessing

Data preprocessing aims to facilitate the training process by appropriately transforming the entire dataset to identify outliers, reduce the effects of outliers, and normalize different features to an equivalent range. The following sections described the data processing procedure. A proper data preprocessing facilitates the model training process and ensures the models' performances are representative of real-world performances.

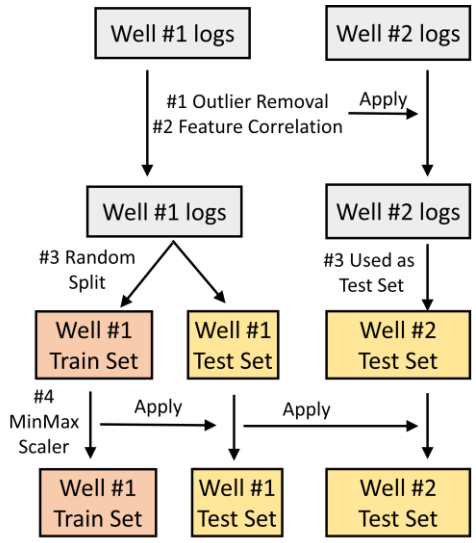


Figure 2-2. Data preprocess flow chart for machine learning models.

The data preprocessing workflow is shown in Figure 2-2. The most obvious difference of the preprocessing workflow compared with a classic data preprocess workflow in machine learning is that the well logs from well #1 and well #2 are preprocessed separately. For a classic machine learning problem, the training and testing datasets are usually randomly generated from a dataset that mixing data from all different sources. By mixing all the datasets together, we can make sure the models learn the

distributions of data from different data sources. However, for the well log data in the oil and gas industry, it is impractical to gather well log data from all wells. New oil wells are drilled every day, and the model should able to perform stably when applied on a new oil well. This highlights the importance of cross-well testing. Data points in well #2 are reserved as the cross-well test set. The models' performance on well #2 should represent the models' real-world performance.

In Figure 2-2, the data preprocessing procedure for well #1 is similar to a standard data preprocessing procedure. The outliers in the dataset are first removed using the isolation forest model. Feature correlations are investigated with Pearson and Spearman correlation coefficients to remove correlated features. The outlier removal model and the feature removal process developed on well #1 is then applied on well #2. Then the well logs in well #1 are randomly split into train and test dataset with a split ratio of 7:3. Lastly, a Minmax scaler is applied on the well #1 train set to scale the train set to a range of [-1, 1]. The trained MinMax scaler is then applied to the well #1 test set and all well logs in well #2. The details of the outlier detection, train test split etc. will be discussed in the following sections.

2.4.2.1 Outlier Detection

The well logs from both well #1 and well #2 are first preprocessed using isolation forest algorithm to remove the outliers. The isolation forest algorithm detects the outliers by randomly selecting a feature and a split value to build a tree-like structure. The normality of a data sample is measured by the average length of the sample to the root of the tree.

In this section, an isolation forest model is fit on the well logs data from well #1. I assume that 1% of the data points are outliers. 85 samples are removed from the well logs in well #1. The same isolation forest model trained on well #1 is then applied on well #2. 11 samples are identified as outliers in well #2. After removing the outliers, 8396 samples are remained in well #1, and 2901 samples are remained in well #2.

2.4.2.2 Feature Correlations

After outliers are removed, the correlations between input well logs are calculated to remove highly correlated input features. Features with a high correlation index have similar information. Including highly correlated features in the model may result in a more complex model. A smaller and simpler model is always preferred in machine learning. By removing the correlated features, a simpler model with fewer parameters can be built with less training time.

Two types of correlation coefficients are used to measure the correlation between features. Pearson correlation coefficient is one of the most widely used correlation coefficients to measure the linear correlation between two variables. It is defined as the covariance of the two variables divided by the multiplication of the standard deviations, which is expressed as

$$\boldsymbol{\rho_{X,Y}} = \frac{\sum_{i=1}^{n}(X_i-\bar{X})(Y_i-\bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i-\bar{X})^2(Y_i-\bar{Y})^2}} \tag{2-1}$$

where *X*, *Y* represent the two variables, *n* is the number of samples, *i* is the sample index.

Spearman correlation coefficient is used to measure the monotonic relationship between two variables. Compared to the Pearson correlation coefficient, the Spearman correlation coefficient measures both linear and nonlinear relationships. The Spearman

correlation coefficient is defined as the Pearson correlation coefficient of two rank variables, which is expressed as

$$r_s = \rho_{rg_X, rg_Y} = \frac{cov(rg_X, rg_Y)}{\sigma_{rg_X} \sigma_{rg_Y}} \quad (2\text{-}2)$$

where *cov* represents the covariance, *rgx* and *rgy* represent the rank of the two variables, and σ is the standard deviation.

The Pearson and Spearman correlation of the input logs are shown in Table 2-1 and Table 2-2. For the Pearson correlation coefficient, two variables are considered highly correlated if the coefficient is higher than 0.9. By examining the correlation coefficient, we can see that the 6 resistivity logs are correlated. Also, we can see the DPHZ and NPOR logs are also highly correlated.

Table 2-1. Pearson correlation coefficient of the 13 input logs (colored by value).

	Lithology	GR	DPHZ	NPOR	PEFZ	RHOZ	RLA0	RLA1	RLA2	RLA3	RLA4	RLA5
Lithology	1.00	0.05	0.30	0.04	0.16	0.30	0.32	0.41	0.24	0.16	0.06	0.01
GR	0.05	1.00	0.44	0.66	0.34	0.44	0.43	0.02	0.15	0.19	0.21	0.22
DPHZ	0.30	0.44	1.00	0.74	0.58	1.00	0.10	0.41	0.44	0.41	0.36	0.34
NPOR	0.04	0.66	0.74	1.00	0.44	0.74	0.12	0.22	0.34	0.35	0.33	0.33
PEFZ	0.16	0.34	0.58	0.44	1.00	0.58	0.19	0.47	0.58	0.59	0.55	0.56
RHOZ	0.30	0.44	1.00	0.74	0.58	1.00	0.10	0.41	0.44	0.41	0.36	0.34
RLA0	0.32	0.43	0.10	0.12	0.19	0.10	1.00	0.37	0.29	0.26	0.16	0.08
RLA1	0.41	0.02	0.41	0.22	0.47	0.41	0.37	1.00	0.76	0.61	0.41	0.31
RLA2	0.24	0.15	0.44	0.34	0.58	0.44	0.29	0.76	1.00	0.94	0.77	0.66
RLA3	0.16	0.19	0.41	0.35	0.59	0.41	0.26	0.61	0.94	1.00	0.92	0.83
RLA4	0.06	0.21	0.36	0.33	0.55	0.36	0.16	0.41	0.77	0.92	1.00	0.96
RLA5	0.01	0.22	0.34	0.33	0.56	0.34	0.08	0.31	0.66	0.83	0.96	1.00

Table 2-2. Spearman correlation coefficient of the 13 input logs (colored by value).

	Lithology	GR	DPHZ	NPOR	PEFZ	RHOZ	RLA0	RLA1	RLA2	RLA3	RLA4	RLA5
Lithology	1.00	0.10	0.29	0.01	0.25	0.29	0.14	0.36	0.22	0.18	0.15	0.13
GR	0.10	1.00	0.39	0.68	0.17	0.39	0.42	0.08	0.02	0.00	0.01	0.01
DPHZ	0.29	0.39	1.00	0.70	0.60	1.00	0.03	0.47	0.51	0.50	0.49	0.48
NPOR	0.01	0.68	0.70	1.00	0.36	0.70	0.12	0.23	0.29	0.30	0.31	0.30
PEFZ	0.25	0.17	0.60	0.36	1.00	0.60	0.33	0.67	0.75	0.78	0.79	0.81
RHOZ	0.29	0.39	1.00	0.70	0.60	1.00	0.03	0.47	0.51	0.50	0.49	0.48
RLA0	0.14	0.42	0.03	0.12	0.33	0.03	1.00	0.38	0.45	0.47	0.47	0.47
RLA1	0.36	0.08	0.47	0.23	0.67	0.47	0.38	1.00	0.92	0.87	0.83	0.80
RLA2	0.22	0.02	0.51	0.29	0.75	0.51	0.45	0.92	1.00	0.99	0.97	0.95
RLA3	0.18	0.00	0.50	0.30	0.78	0.50	0.47	0.87	0.99	1.00	1.00	0.99
RLA4	0.15	0.01	0.49	0.31	0.79	0.49	0.47	0.83	0.97	1.00	1.00	1.00
RLA5	0.13	0.01	0.48	0.30	0.81	0.48	0.47	0.80	0.95	0.99	1.00	1.00

By comparing the Pearson and Spearman correlation coefficient, DPHZ is removed because the Spearman correlation between DPHZ and RHOZ is 1, which means they are monotonically correlated. RLA3, RLA5 are removed because they are highly correlated with RLA2 and RLA4 (Spearman > 0.9). By removing the three logs, only 9 logs are used for the following sections.

2.4.2.3 Feature Correlation with Target

The correlations between input logs and output logs are also investigated. To predict the DTC and DTS logs, we hope the input logs contain information that is related to the output targets. By calculating the correlation index, we can estimate the contribution of each input logs in the prediction of the output targets. If an input log is highly correlated with the output logs, the models can predict the output by a linear or non-linear transformation of the input log.

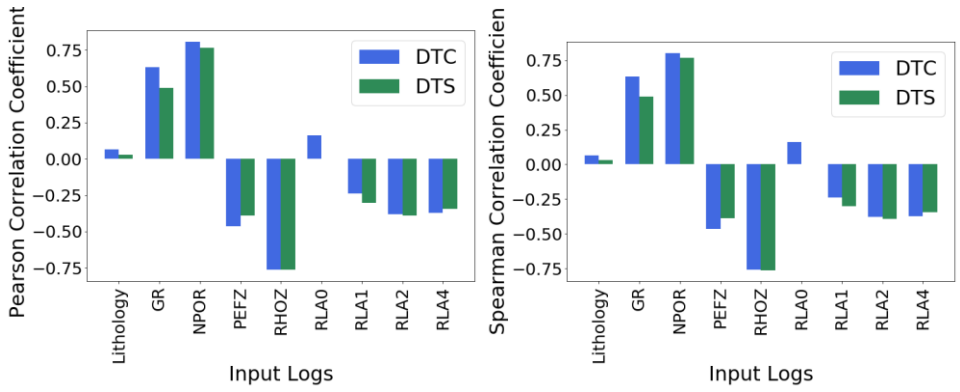


Figure 2-3. Pearson and Spearman correlations between input logs and output sonic logs in well #1.

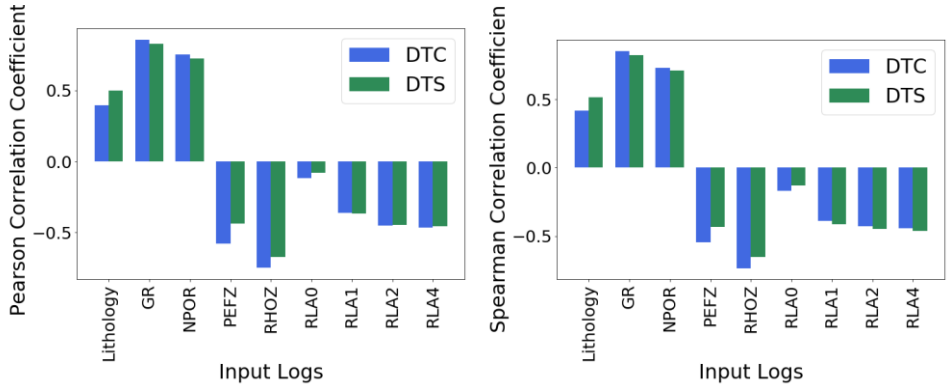


Figure 2-4. Pearson and Spearman correlations between input logs and output sonic logs in well #2.

Figure 2-3 and Figure 2-4 plot the Pearson and Spearman correlation between the 9 input logs and the output DTC and DTS logs in well #1 and well #2. We can see that most logs have a high correlation with the output logs. In well #1, the lithology log does not have a high correlation with the output logs. However, the correlation coefficient between the lithology log and sonic logs in well #2 is considerably high. RLA0 log has a low Pearson and Spearman correlation coefficient in both well #1 and well #2. So, the RLA0 log is removed. After removing the RLA0 log, only 8 input logs are left as input logs for the machine learning models.

2.4.2.4 Train Test Split

In this section, the train and test dataset split methods are introduced. For a classic machine learning model training process, a common way for train and test dataset split is to randomly split the whole dataset into train and test dataset by a ratio. In our study, different models are built to test different hypotheses. Two different train test split methods are implemented based on the characteristic of our dataset and our hypothesis.

Three types of models are implemented in this chapter: empirical models, simple machine learning models without considering depth dependencies (6 shallow learning models), and ANN model considering depth dependencies (multi-depth ANN model). For empirical models and simple machine learning models that do not consider depth dependencies, the train test split method is the commonly used random split method. 70% of the logs in well #1 is randomly selected and used as training and validation dataset. The rest 30% of the logs are used as testing. All the logs from well #2 are used only as cross-well testing dataset.

Depth dependencies are a common phenomenon in well logs. The properties of formation at a specific depth can be estimated by the neighboring depth. To consider the depth dependencies, I used a simple ANN that simultaneously processes the input features/logs from three consecutive depths to predict DTC and DTS logs corresponding to the input logs from the centermost depth among the multiple depths. A conventional application of machine learning models does not take well logs at multiple depths as input. To build an ANN model that takes multiple depths as input, I split the dataset by depth. 70% of well logs on the top of well #1 is used as the train set and validation

dataset. The lower 30% of the well logs in well #1 is used as the test set. All the well logs in Well #2 are used as a cross-well testing dataset.

2.4.2.5 Feature Scaling

Normalization is performed in this study. Normalization ensures fast convergence of the learning process, especially when using neural networks. Normalization is performed using Minmax scaler which is expressed by Equation (2-3):

$$y'_i = 2\frac{y_i - y_{min}}{y_{max} - y_{min}} - 1 \quad (2\text{-}3)$$

where y_i is the original value of a log response and y'_i is the normalized value of the log response at each depth *i*. Equation (2-3) normalizes all the logs (features) within a range of [-1, 1]; consequently, the models will not be biased towards the logs having large values and range. In this study, we didn't shape the input log distribution or tried other standardization/normalization techniques.

2.5 Model Introduction

2.5.1 Literature Survey on The Machine Learning Models

The study of synthesizing sonic logs (DTC or DTS) from other logs has been studied by many researchers with different approaches. Empirical models have been developed to describe the relationship between sonic logs and other formation properties. In recent researches, some researchers applied different kinds of machine learning models in their studies to generate synthetic sonic logs.

It is a common phenomenon that different well logs exhibit some level of correlations. The current acquisition of reservoir properties relies on the interpretation of dozens of well logs. Each kind of well log measures one aspect of reservoir properties based on the formation responses of mechanical or electrical signals. The assumption of

synthesizing sonic logs from other logs is that it is not necessary to use dozens of well logs to acquire the information of a reservoir. Some well logs can be generated by processing the integrated information contained in other well logs.

Sonic logs measure the sonic wave propagation slowness of the subsurface formation. Sonic logs are critical for the estimation of the formation mechanical properties. The lack of sonic logs or the untrustworthy sonic logs readings may pose challenges for the analysis of the physic of the formation. To deal with such circumstances, researchers have proposed empirical models to describe the relationship of the DTC and DTS logs with respect to other formation measurements.

Correlations between well logs are a very common phenomenon (Castagna et al., 1985; Onalo et al., 2018). Several empirical models were developed based on the fact that there is a high correlation between sonic logs and other formation measurements (Castagna et al., 1985; Eberhart‐Phillips et al., 1989; Greenberg and Castagna, 1992b; Han et al., 1986; Hossain et al., 2012; Johnston and Christensen, 1993).

Machine learning models have shown the great ability to find hidden relationships in a large dataset and have been applied in many areas. Some studies have applied different models for the purpose of sonic log generation. With the emergence of different machine learning models, many researchers applied machine learning in the prediction of petrophysical properties. Among all of the machine learning models, ANN is preferred due to its flexibility and simplicity. Elkatatny et al., applied artificial neural networks and support vector machine etc. to generate compressional and shear sonic time from logs of gamma ray, build density and neutron porosity (Elkatatny et al., 2016). The study conducted by Elkatatny et al. achieved an accuracy of 0.99. The results show that it is

possible to apply machine learning models for the sonic logs synthesis tasks. In another study by Rezaee et al. (2017), an intelligent system that combines ANN, fuzzy logic etc., is applied to predict shear wave velocity (Rezaee et al., 2007). Both studies show that the models applied achieved stable and high synthesis accuracy in testing data. A similar study applied a committee machine with intelligent systems to predict sonic travel time from conventional well logs (Asoodeh and Bagheripour, 2012). Another study performed by Onalo et al. (Onalo et al., 2018) applied a three-layer feedforward multilayer perception ANN. The model is trained to predict compression and shear wave transit time using real gamma ray and formation density log.

The hidden relationships in different well logs are subtle for human engineers to find out. Machine learning models can find hidden relationships from the training dataset. By applying machine learning models, we hope to reduce the number of well logs that are necessary for reservoir characterization, and thus reduce the cost of well logging.

2.5.2 Introduction About Models Applied

Six simple prediction models are used to synthesize DTS and DTC logs by processing 8 ‘easy-to-acquire’ logs. I assume that the 6 shallow learning models can capture the hidden relationships between the 8 input logs and the 2 output sonic logs. Predicting DTC and DTS logs has been a research topic for a long time. Before the emergence of various machine learning models, researchers have applied empirical models for the purpose of DTC and DTS logs synthesis. I will also briefly introduce 2 empirical models that applied in our study. The results of classic empirical models and machine learning models are compared.

2.5.2.1 Empirical Models

In our study, I applied two widely used empirical models as a comparison to our machine learning models. The first model was developed by Castagna et al. (Castagna et al., 1985). In the following sections, the model developed by Castagna et al. will be referred to as the Castagna model. The model used commonly measured porosity and lay content in shaly-sand formation to predict the P and S wave velocities, which are the reciprocal of the DTC and DTS. The Castagna model is expressed as

$$\boldsymbol{v_p = 5.81 - 9.42(\emptyset_T) - 2.21(V_{cl})} \quad (2\text{-}4)$$

$$\boldsymbol{v_s = 0.862(v_p) - 1.172} \quad (2\text{-}5)$$

where $\emptyset_T$ is the total porosity, V_{cl} is the clay volume fraction, and v_p and v_s are the P wave and S wave velocities in km/s, respectively.

Han et al. developed a similar empirical model for clay-bearing sandstone (Han, 1987):

$$\boldsymbol{v_p = A - B(\emptyset_T) - D(V_{cl})} \quad (2\text{-}6)$$

$$\boldsymbol{v_s = 0.79(v_p) - 0.79} \quad (2\text{-}7)$$

where A, B, D are constraints. I will refer to the model developed by Han et al. as the Han model in the following sections.

Both models use simple regression on the sonic experimental dataset. The P wave velocity is calculated as a linear combination of total porosity and clay volume fraction. The S wave velocity is linearly transformed from the P wave velocity. From the empirical models, we can see that the relationship between the sonic wave velocities (which can be transformed to slowness) and other formation measurements are simple. The P wave and S wave velocities can be estimated from total porosity and clay volume fraction with a

linear relationship. Both of the empirical models are widely used before machine learning models became popular.

The DTS log predicted with the empirical models is also included in this chapter. The results are used as the baseline for our machine learning model. I assume that the machine learning models implemented should outperform empirical models. This is because empirical models are basically linear regression of sonic logs, total porosity, and clay volume fraction. Other researchers have proposed more complicated models. But most of the empirical models are based on the linear relationships like the Castagna model and Han model. Machine learning models can build large models with more parameters given a good training dataset. In this study, the machine learning models are trained to find the relationship between 8 input logs and two sonic logs. With more input variables and more complex models, machine learning models should outperform empirical models.

2.5.2.2 Ordinary Least Squares (OLS) Model

OLS model is one of the simplest statistical regression models that fit the data by reducing the Sum of Squared Errors (SSE) between the modeled and measured data (Chumney, 2006). The OLS model assumes the output y_i is a linear combination of input values x_{ip} and error ε_i formulated as

$$\boldsymbol{y_i = \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} + \varepsilon_i} \quad (2\text{-}8)$$

where i represents a specific depth from the total depth samples available for model training and p represents the number of input logs available for the training. The model fits the data set by reducing the sum of squared errors (SSE) expressed as

$$\mathbf{SSE} = \sum_{i=1}^{n} (\hat{\boldsymbol{y}}_i - \boldsymbol{y}_i)^2 \quad (2\text{-}9)$$

where $\hat{y}_i$ is the predicted output of the model. In this study, the input data x_{ip} are the 8 raw logs at depth i and the output data y_i are sonic logs. OLS predictions are adversely affected by outliers, noise in data, and correlations among the inputs.

2.5.2.3 Partial Least Squares (PLS) Model

PLS model aims to find the correlations between the input data and output data by constructing latent structures. Input and output logs are decomposed into their latent structures(Abdi, 2010; Lingjaerde and Christophersen, 2000; Wold, 2004). The latent structure corresponding to the most variation in output is extracted and then explained using a latent structure in the input space. New latent structures are combined with the original variables to form components. The number of components *m* (no more than inputs) are chosen to maximally summarize the covariance with outputs. The determination of *m* builds the PLS model with fewer inputs than the OLS model, suited when there are high correlations among inputs. The removal of redundant inputs can save computational time for training the model while simplifying the model's structure.

2.5.2.4 Least Absolute Shrinkage and Selection Operator (LASSO) Model

LASSO (Tibshirani, 1996) is a linear regression model that combines the SSE and constrains the sparsity of the coefficient matrix with L_1 norm. The objective function of LASSO is expressed as

$$\min_{\boldsymbol{w}} \frac{\mathbf{1}}{\mathbf{2}\boldsymbol{n}} \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{Y}\|_{\mathbf{2}}^{\mathbf{2}} + \boldsymbol{\lambda}\|\boldsymbol{w}\|_{\mathbf{1}} \tag{2-10}$$

where *w* is the coefficient vector, *X* is the measured input vector, *Y* is the measured output vector, n is the number of depth samples in the training dataset, and λ is the penalty parameter that balances the importance between the SSE term and the regularization term, which is the L1 norm of the coefficient vector. The SSE is represented as the

squared L2 Norm. If λ increases, the regularization term punishes the coefficient matrix to be sparser. The λ term is optimized by testing a range of values. Different values of the λ are tested and the prediction performance of the model is compared.

2.5.2.5 ElasticNet Model

ElasticNet model is a linear regression model suitable for high-dimensional data that combines both L_1 norm and L_2 norm as the penalty term (Kim et al., 2007; Zou and Hastie, 2005). Unlike the LASSO model, the ElasticNet model preserves certain groups of correlated input logs and does not ignore highly correlated inputs. The objective function of the ElasticNet model is defined as

$$\min_{w} \frac{1}{2n} \|Xw - y\|_2^2 + \alpha_1 \|w\|_1 + \alpha_2 \|w\|_2^2 \quad (2\text{-}11)$$

In comparison to Equation (2-10) relevant to LASSO, Equation (2-11) uses the L2 norm. ElasticNet is a combination of ridge regression and LASSO regression. Ridge regression uses the L2 norm, which generates less sparse solution as compared to LASSO regression. ElasticNet balances L1 and L2 norms.

2.5.2.6 Multivariate Adaptive Regression Splines (MARS) Model

The MARS model uses multiple linear regression models in the input space (Friedman, 1991). By fitting the data using multiple linear regressions, the model can capture the non-linearity of the dataset. The model is a weighted sum of basis functions. MARS model is formulated as

$$\hat{y}_i = \sum_{q=1}^{p} \alpha_q B_q(x_{qi}) \quad (2\text{-}12)$$

where x_{qi} is the value of q-th input log x_q at the i-th depth point, $B_q(x_{qi})$ is a basis function and α_q is the coefficient of B_q. The basis function can have many different

forms, generally, it is a hinge function. A hinge function is a linear function within a range expressed as:

$$\mathbf{max}(\mathbf{0}, \boldsymbol{x_{qi}} - \boldsymbol{C_q}) \text{ or } \mathbf{max}(\mathbf{0}, \boldsymbol{C_q} - \boldsymbol{x_{qi}}) \quad (2\text{-}13)$$

The shape of a hinge function can be visualized in Figure 2-5. Hinge function partitions the input into different sections by using different C_q. MARS algorithm approximates nonlinear relationship with multiple hinge functions. As expressed in equation (2-13), each hinge function is zero for some portion of the input range based on the selection of C_q. By combining hinge functions with different C_q, the MARS algorithm splits the input range into small sections and approximate each section separately.

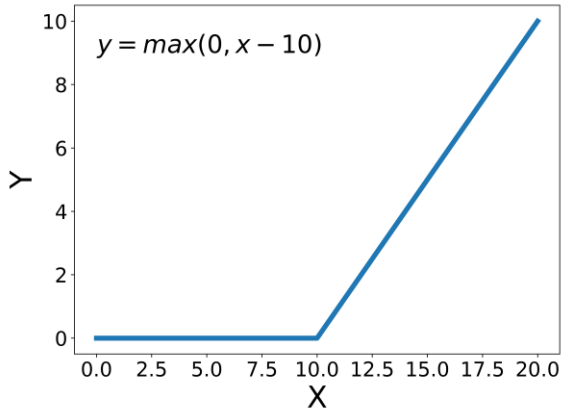


Figure 2-5. Shape of a hinge function with $C_q = 10$.

2.5.2.7 Artificial Neural Network (ANN) Model

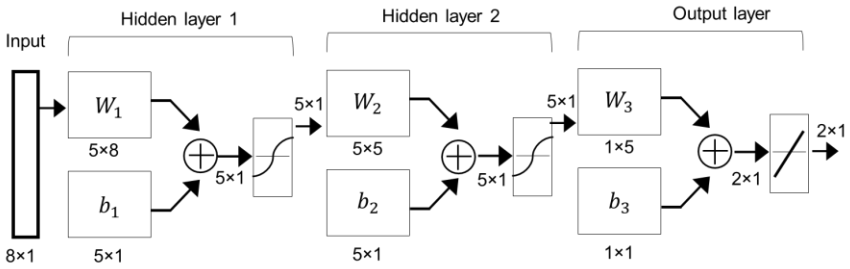


Figure 2-6. Neural network architecture example (Beale et al., 2012).

ANN is a widely used machine learning model suitable for both linear and nonlinear regression (Yegnanarayana, 2009). A neural network comprises an input layer, an output layer, and a few hidden layers. The capacity of the neural network model to fit data can be adjusted by adding or decreasing the number of hidden layers and the number of neurons in each hidden layer. Each hidden layer contains neurons, made of parameters (weights and biases), to perform matrix computations on signals computed in the previous layer. The activation function in each layer adds nonlinearity to the computation. All the models in this dissertation are tuned with the exhaustive grid search. For the ANN model, different learning rates, number of the hidden layers, number of neurons, activation functions are tested using exhaustive grid search. Figure 2-6 shows an example of ANN architecture. There are 8 input logs and 1 output logs (DTC or DTS) to be synthesized. The dimension of the input and output layers are 8 and 1 respectively. There are two hidden layers in the ANN model with 5 neurons in the first and second hidden layers, respectively.

2.5.3 Machine Learning Workflows for Modeling of Compressional and Shear Travel times

2.5.3.1 Model Training and Evaluation

After preprocessing, three datasets are generated. Well #1 train dataset is used for model training and hyperparameters tuning. Grid search and cross-validation are applied when a machine learning model is trained on this dataset. Well #1 testing dataset is to test the performance of the trained model on well #1. We can expect machine learning models should have similar performance on well #1 test dataset to the well #1 train dataset. This is because the two datasets follow the same distribution. The well logs from

well #2 are used to cross-well testing. This procedure is to test the model's performance when applied to a new well. We can expect the model's performance should drop since the distributions of the well logs in different wells are different.

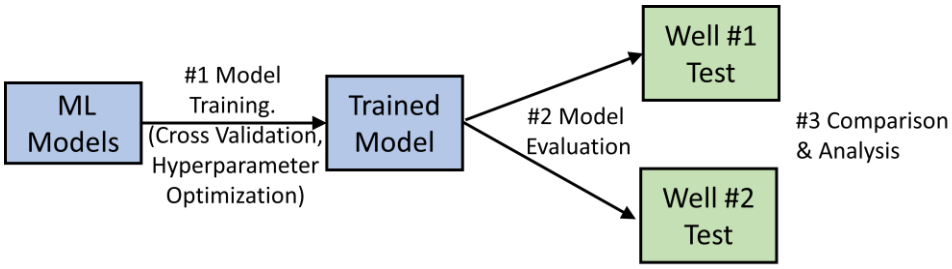


Figure 2-7. Machine learning training, cross-validation, and testing workflow.

After the train and test dataset is generated, the machine learning models are trained on the well #1 train dataset as shown in Figure 2-7. 5-fold cross-validation is used in combination with grid search to select the best hyperparameters of different machine learning models. After the models are trained, the models are tested on two different test datasets. One is the well #1 test set, the other is the well #2 test set. By comparing the testing results, we can evaluate the feasibility of applying trained machine learning models on a new oil well in the real world. The workflow is only used on machine learning models, empirical models are applied directly.

2.5.3.2 Model Stability Investigation

To investigate the models' stability, the workflow shown in Figure 2-7 is repeated 100 times. Each time, the training dataset is resampled with replacement using the bootstrap method. The bootstrap method is basically a resampling method used to evaluate the stability of machine learning models. When the training dataset is resampled using the bootstrap method, a new dataset is generated by randomly choosing the samples from the original train set. Each time we repeat the training process, a new train set is

generated (with duplicated samples) by the bootstrap resampling method. The bootstrap resampling changes the distribution of the dataset, and thus we can get 100 different models. The performance of the 100 models is evaluated on the test set. We can expect that the performance of the 100 trained models is different. A good model should have a high prediction accuracy and low prediction variance. The variance represents the stability of the model when exposed to new samples. We hope the trained model has a low variance.

2.6 Data-Driven Modeling

2.6.1 Metrics Used for Model Evaluation

4 different metrics are used to evaluate the performance of the models on the test dataset. They are R^2 coefficient (coefficient of determination), mean squared error (MSE), Akaike information criterion (AIC), and the Bayes Information criterion (BIC). Each metric has its own advantage and disadvantage. By comparing the value of different metrics, we can better evaluate the performance of the models. In this chapter, we observe that the values of the four metrics are correlated for the models.

The R^2 coefficient is an evaluation of the variance explained by the model. The R^2 coefficient is a popular metric for regression task. The value of the R^2 coefficient is limited to 0-1, which enable users compare different models on the same scale. The MSE metric measure the average squared error of the predictions. The MSE is always positive and there is no upper limit. The Akaike information criterion (AIC), and the Bayes Information criterion (BIC) are two metrics based on the information theory. The two metrics take the number of parameters of machine learning models into consideration. For two models that have similar performance, the model with fewer parameters

(weights) is always preferred. This is because a simpler model is easier to train and maintain. Increase the model's complexity without an obvious gain in performance should always be avoided.

R^2 coefficient is a widely used metric for the goodness of fit. R^2 coefficient measures the variance explained by the model, which is expressed as

$$\boldsymbol{R^2(y,\hat{y}) = 1 - \frac{\sum_{i=1}^{n}(y_i-\hat{y}_i)^2}{\sum_{i=1}^{n}(y_i-\bar{y}_i)^2}} \quad (2\text{-}14)$$

where y, $\hat{y}$ represent the actual and predicted values.

Mean squared error (MSE) measure the error of the fitting results, which is expressed as

$$\boldsymbol{MSE = \frac{1}{N}\sum_{i-1}^{n}(y_i - \hat{y}_i)^2} \quad (2\text{-}15)$$

The AIC model estimates the quality of the model by combining the prediction errors and the number of parameters in the model:

$$\boldsymbol{AIC = n \times log\left(\frac{\sum_{i-1}^{n}(y_i-\hat{y}_i)^2}{n}\right) + 2k} \quad (2\text{-}16)$$

where *n* is the number of samples, *k* is the number of parameters.

BIC is defined similarly to AIC, which is expressed as

$$\boldsymbol{BIC = n \times log\left(\frac{\sum_{i-1}^{n}(y_i-\hat{y}_i)^2}{n}\right) + k * log\,(n)} \quad (2\text{-}17)$$

The idea behind BIC and AIC metrics is similar to the regularization procedure of machine learning models. The AIC and BIC metrics measure the performance of models by both the prediction accuracy and the models' simplicity. The best model balances simplicity and accuracy.

2.6.2 Comparison of Prediction Performances of Machine Learning Models

The above mentioned 6 shallow learning models are trained and tested in a 4240-feet depth interval in Well 1 with 8396 data points and deployed for blind testing in 1460-

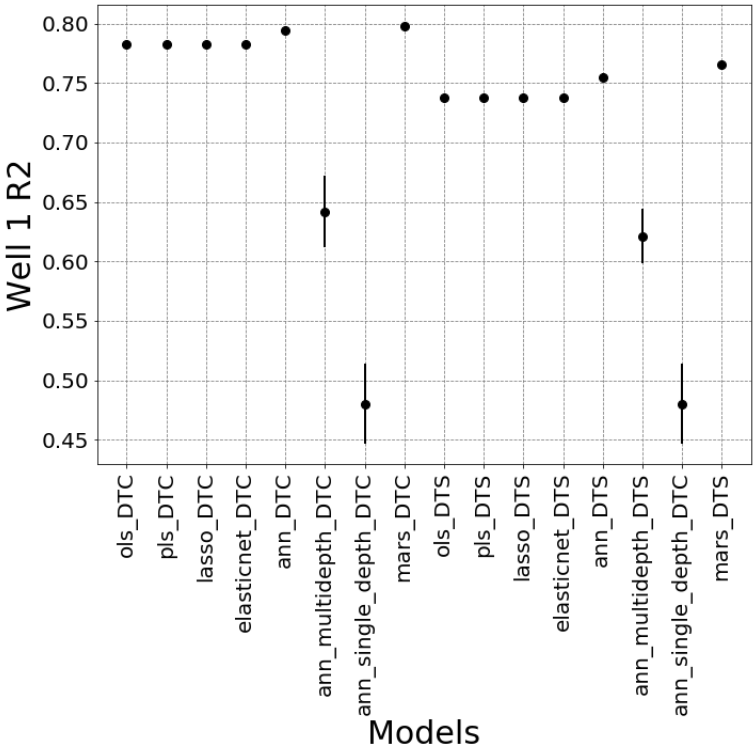
feet depth interval in Well 2 with 2901 data points. During the training and testing phases, the models were trained with 5-fold cross-validation using 70% of randomly selected data from Well 1 and tested with 30% of the data. 5-fold cross-validation was performed to ensure the robustness of the model predictions. The 5-fold cross-validation method split the training dataset into 5 equal parts. The model is trained 5 times, each time the model is trained with 4-fold and validated with the remaining fold of data. This cross-validation method ensures that all the model is trained to generalize over the entire available dataset.

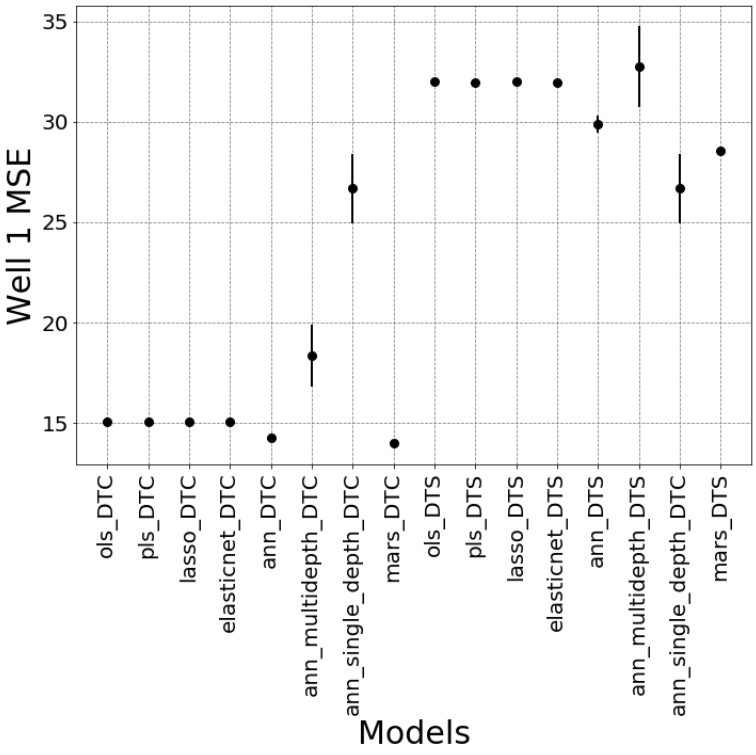
In this section, the performance of the 6 machine learning models is compared with the metrics introduced in the previous section. Models are built and trained separately for both DTC and DTS logs. The 6 shallow machine learning models are trained and tested with randomly sampled data in well #1. For the purpose of comparison, the performance of empirical models, a single-depth ANN model, and a multi-depth ANN model are also included in the following tables and figures. The details of the empirical models, single-depth ANN model, and multi-depth ANN model will be introduced in the following sections. This section focusses on the performance of the 6 shallow learning models (OLS, PLS, LASSO, ElasticNet, MARS, ANN).

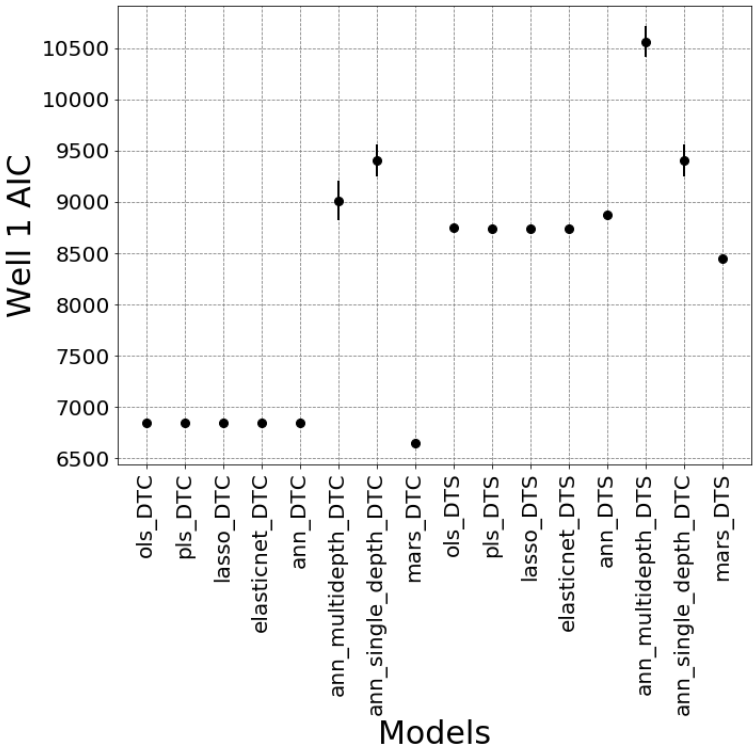
The performance of log synthesis is evaluated in terms of the four metrics. The log synthesis results for Wells 1 & 2 are shown in Table A-1. The columns of Table A-1 are the models implemented. The rows of the tables represent four metrics used to evaluate each of the models. Different models are built to predict DTC and DTS logs separately. Each model is evaluated on the test set in well #1 and all the data in well #2. Four metrics are used to evaluate the models. For example, the data in the first row and

first column of Table A-1 represents that the R^2 metric used to evaluate the OLS model, which is built to predict DTC log, is 0.782 on the testing data on well #1.

From Table A-1, we can have the following observations. First, the four metrics are mostly correlated. For example, the MARS models have a higher R^2, lower MSE, AIC, and BIC compared to other models. Second, the models' performance on well #1 testing dataset is better than that on well #2. For example, the R^2 metric on well #2 is much lower than well #1. This is because the models are trained with data sampled from well #1. The models learned the distribution of the data on well #1. Data from different wells will follow different distributions even if they are from the same reservoir. A similar trend can also be observed from the Figure 2-8.




Well 1 MSE
35
30
25
20
15
ols_DTC
pls_DTC
lasso_DTC
elasticnet_DTC
ann_DTC
ann_multidepth_DTC
ann_single_depth_DTC
mars_DTC
ols_DTS
pls_DTS
lasso_DTS
elasticnet_DTS
ann_DTS
ann_multidepth_DTS
ann_single_depth_DTC
mars_DTS
Models


Well 1 AIC
10500
10000
9500
9000
8500
8000
7500
7000
6500
ols_DTC
pls_DTC
lasso_DTC
elasticnet_DTC
ann_DTC
ann_multidepth_DTC
ann_single_depth_DTC
mars_DTC
ols_DTS
pls_DTS
lasso_DTS
elasticnet_DTS
ann_DTS
ann_multidepth_DTS
ann_single_depth_DTC
mars_DTS
Models

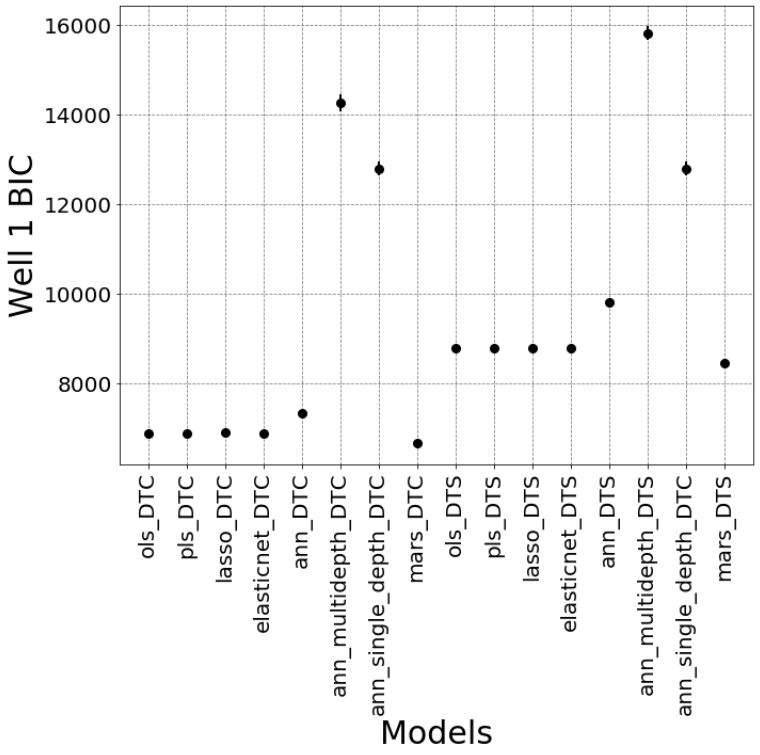
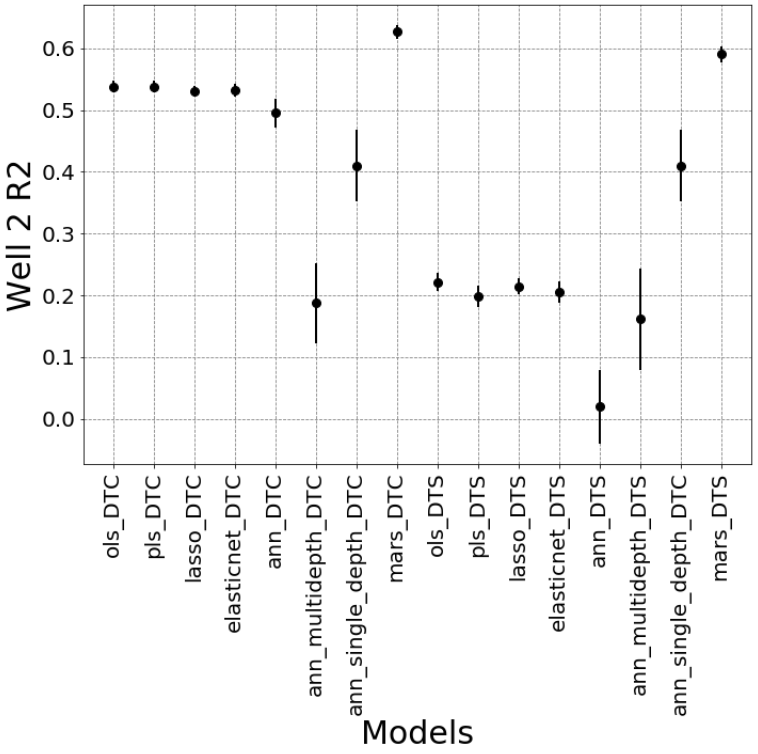


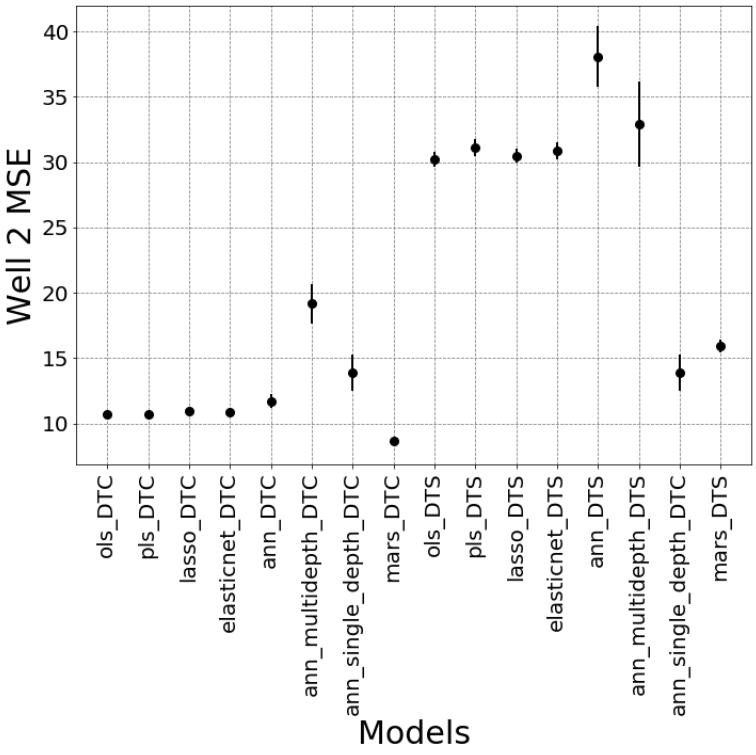
Figure 2-8. Confidence interval (95%) of the metrics evaluated using the bootstrap method. The models are trained with training data from Well #1. The figures show the models' performance on testing data that is sampled from the Well #1.

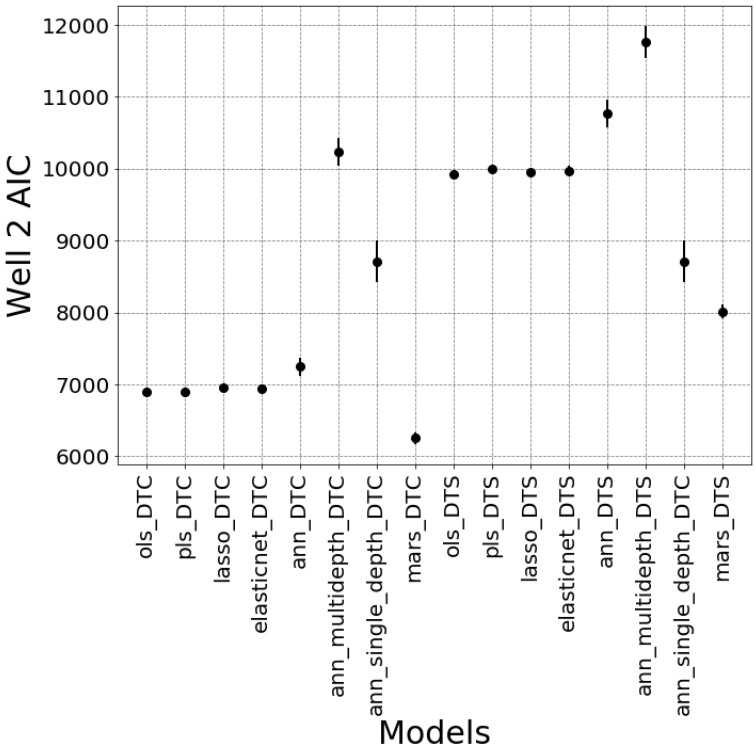
Models with stable prediction performance are preferred. Figure 2-8 and Figure 2-9 plot the confidence intervals (95%) of each model evaluated with the bootstrap resampling method. Each model is trained with resampled data generated by bootstrapping 100 times. The evaluation results of the 100 models are gathered, and the confidence intervals are calculated. This analysis helps us understand the models' stability when exposed to a new dataset.

Figure 2-8 plots the confidence intervals of the four metrics evaluated on the test set from well #1. Except for the 6 shallow machine learning models trained with the sampled train set, two ANN models trained to investigate the depth dependencies are also included (ann_multidepth model and ann_single_depth model). The two ANN models will be discussed in the following sections. This section will focus on the 6 shallow

models. By comparing the four metrics, we can have the following observations. First, the models implemented for the synthesis of DTC logs perform better than those implemented for the synthesis of DTS logs. Second, MARS models have better performance on both DTC and DTS logs compared to other models. For example, the R^2 of the MARS models on DTC and DTS logs are 0.798 and 0.766 respectively. The MSE, AIC, and BIC metrics of the MARS models are also the lowest. Third, the confidence intervals of the 6 shallow learning models are really small, which means the models are quite stable. From Figure 2-8, we can hardly recognize the confidence interval of the 6 shallow models. If the models have a large confidence interval, the bar represents the confidence interval should be recognizable (e.g. the ann_multidepth model and ann_single_depth model have large confidence interval).




Well 2 MSE
40
35
30
25
20
15
10
ols_DTC
pls_DTC
lasso_DTC
elasticnet_DTC
ann_DTC
ann_multidepth_DTC
ann_single_depth_DTC
mars_DTC
ols_DTS
pls_DTS
lasso_DTS
elasticnet_DTS
ann_DTS
ann_multidepth_DTS
ann_single_depth_DTC
mars_DTS
Models


Well 2 AIC
12000
11000
10000
9000
8000
7000
6000
ols_DTC
pls_DTC
lasso_DTC
elasticnet_DTC
ann_DTC
ann_multidepth_DTC
ann_single_depth_DTC
mars_DTC
ols_DTS
pls_DTS
lasso_DTS
elasticnet_DTS
ann_DTS
ann_multidepth_DTS
ann_single_depth_DTC
mars_DTS
Models

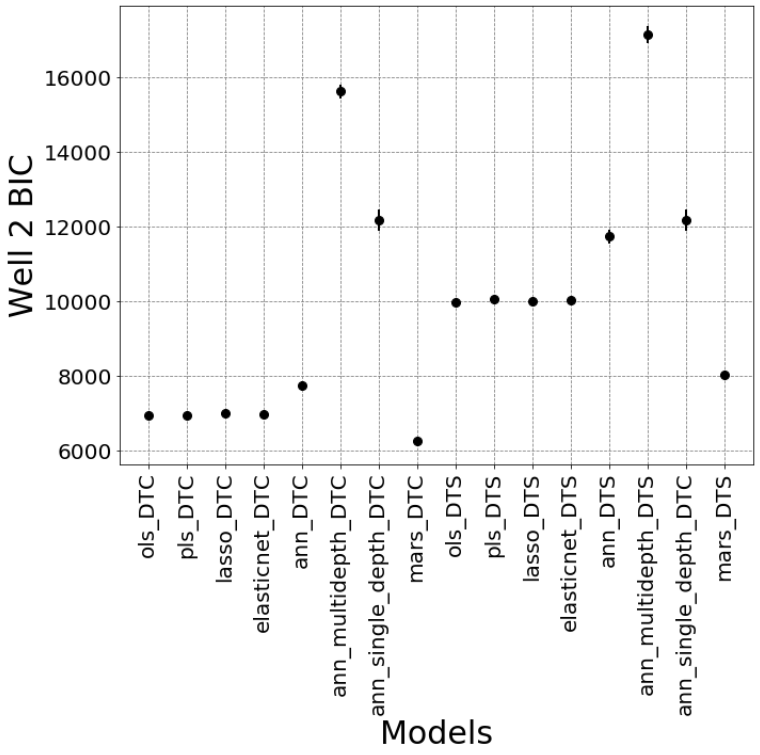


Figure 2-9. Confidence interval (95%) of the metrics evaluated using the bootstrap method. The models are trained with training data from Well #1. The figures show the models' performance on testing data from the Well #2.

Figure 2-9 plots the confidence interval of the models evaluated on data on well #2. The data plotted in Figure 2-9 tells us the models' cross-well testing performance. If the models learn the relationship of the input logs and the DTC and DTS logs that represents the relationship in the real world, the models should have a good performance when exposed to a dataset from a new oil well. Similar observations are observed in Figure 2-9 as in Figure 2-8. Apart from that, the performance of the models is dropped on the test set from well #2. For example, the mean R^2 value of the MARS DTC and DTS models are 0.798 and 0.766 on testing data from well #1. When the models are applied to well #2, the corresponding mean R^2 values are 0.627 and 0.590. MARS model shows great model stability compared to other models. Other shallow models (OLS, PLS, LASSO, ElastiNet, ANN) are not as good as the MARS model on well #2. The R^2 values

of the other 5 shallow learning models are around 0.5 and 0.2 for the DTC and DTS logs respectively, which are much lower compared to the MARS model. All of the 6 shallow learning models show a small confidence interval.

One thing to notice is that the ANN model has the worst performance compared to other models when applied in well #2. However, the performance of ANN in well #1 is pretty good. This indicates the ANN is overfitted the training dataset from well #1. The ANN model is trained with 5-fold cross-validation and exhaustive grid search for hyperparameter tuning. If the trained ANN model is exposed to well log data similar to well #1, we can expect a good performance of the ANN model. However, the distribution of the well log data in well #2 is different from well #1, which is a common phenomenon in the oil and gas industry. This problem is unique in the oil and gas industry. Although the ANN model is thoroughly validated and tested with data from well #1, the ANN model fails to predict the DTS logs on well #2. The other models have a limited fitting ability due to the limited number of parameters, and thus the other models perform better than ANN model on well #2.

The actual and predicted DTC and DTS logs are plotted in Figure 2-10 to give a more intuitive understanding of the MARS model's performance. The measured and synthesized sonic logs are compared across randomly selected 300 depth samples from Well #2. The first two tracks plotted the model's prediction and the relative error of the prediction for the MARS model built for the DTC log prediction. The last two columns show the same information for the MARS model used for DTS log prediction.

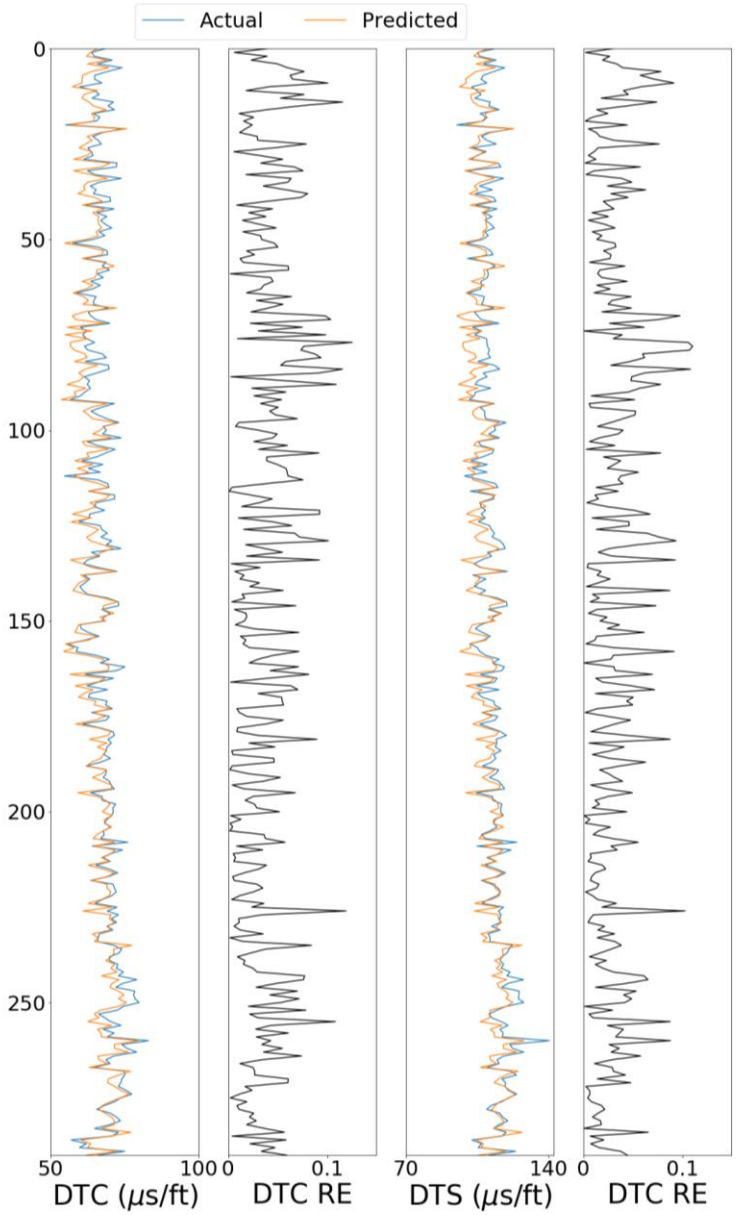


Figure 2-10. Comparison of true and predicted DTC and DTS logs in Well #2, when the MARS model is trained and tested in Well #1 and deployed in Well #2 to synthesize the DTC and DTS logs. Relative error (RE) of each prediction is also plotted.

In Figure 2-10, Relative Error (RE) is used to evaluate the prediction performance of the MARS model for log synthesis. RE for a log synthesis is formulated as

$$\boldsymbol{RE} = \frac{|\boldsymbol{P}-\boldsymbol{M}|}{\boldsymbol{M}} \tag{2-18}$$

where P is the predicted value and M is the measured value of either DTS or DTC log at depth i.

From Figure 2-10 we can see that the actual and predicted DTC and DTS logs are quite close. From the perspective of relative error, most of the relative errors are smaller than 10%. This can be further illustrated by Figure 2-11, which plots the distribution of the relative error of predicted DTC and DTS logs. From Figure 2-11, we can see that most of the predicted DTC and DTS values have relative errors smaller than 5% of the actual log value. Since the results are derived from the test set from well #2, we believe the results should represent the real-world performance of the model. The mean and median of the relative error of the synthesized DTC log are 0.0378 and 0.0338. The mean and median of the relative error of the synthesized DTS log are 0.0333 and 0.0292.

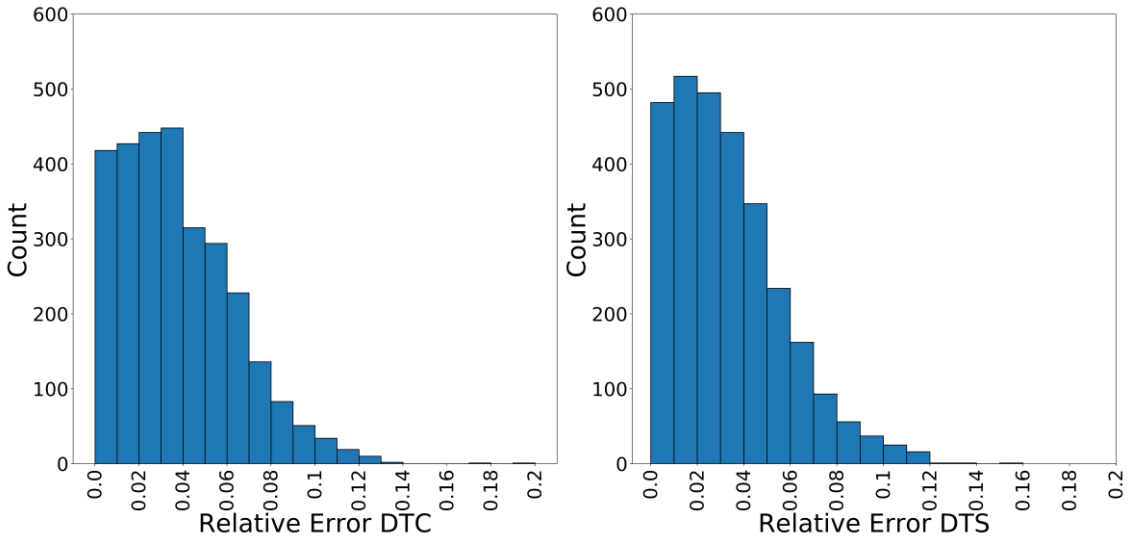


Figure 2-11. Comparison of relative error distribution of DTC and DTS logs for the MARS model deployed in well #2.

2.6.3 Comparison of MARS Model and Empirical Model on DTS Log

In the previous section, 6 shallow learning models are implemented and compared. MARS model shows better performance compared to other models. However,

it is hard to get an understanding of the model's performance by just comparing the 6 shallow learning models. In this section, two empirical models that are widely used in the oil and gas industry for the purpose of sonic log predictions are implemented as a baseline for other machine learning models. The empirical models of Castagna and Han are introduced in section 2.5.2.1. Here we just compared the prediction results of DTS logs using the dataset from well #2.

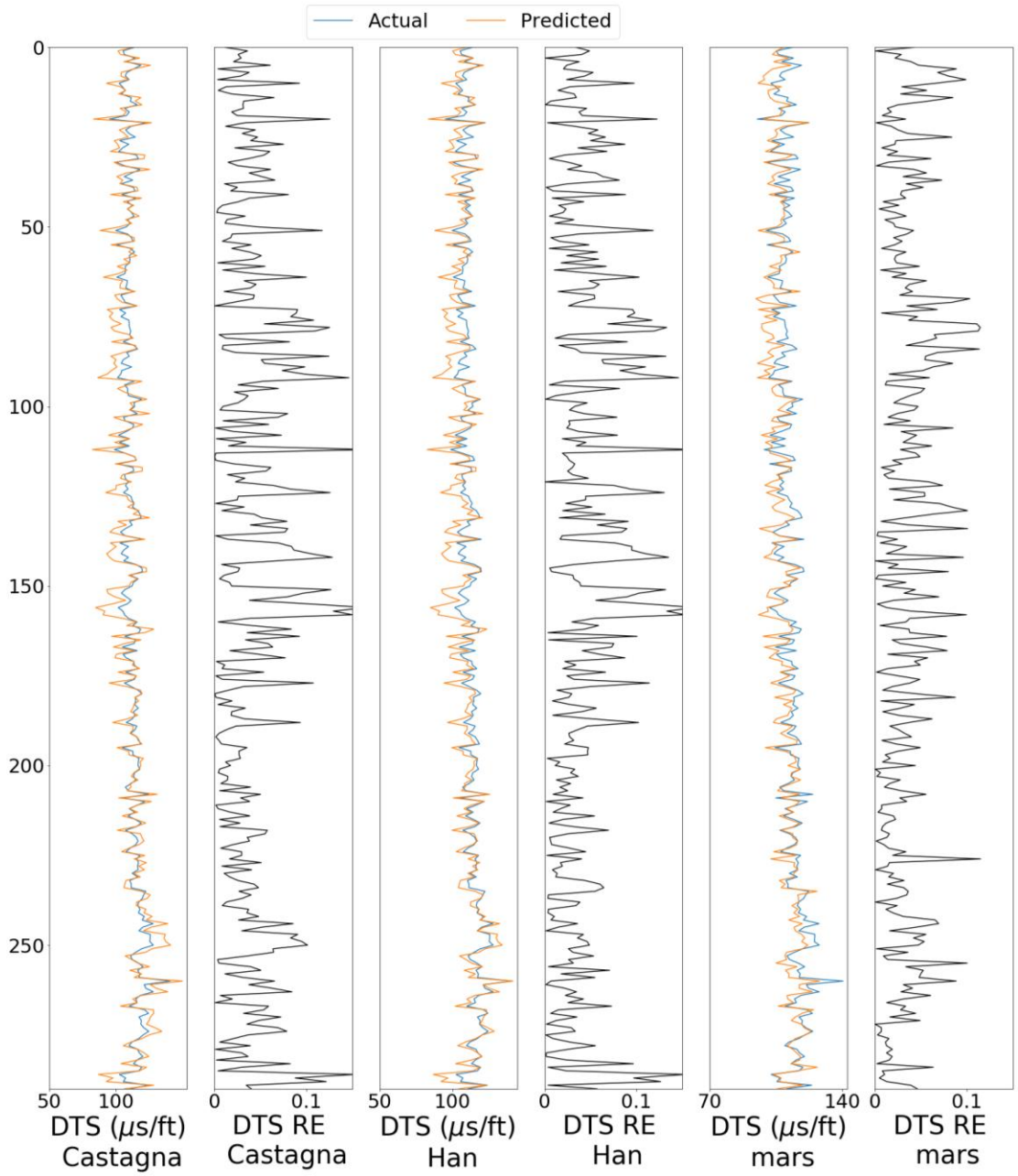


Figure 2-12. Comparison of actual and predicted DTS log of the MARS model and empirical models in Well #2.

The MARS model implemented in the previous section is used to compare with empirical models since it is the best performing shallow learning model. Figure 2-12 plots the prediction results and relative error of the MARS model on the last two tracks. The MARS model is trained using the training dataset from well #1. The prediction

results and relative error of the two empirical models are plotted on the first four tracks. The empirical models predict the DTS logs using the learning linear relationship of DTC and DTS described in section 2.5.2.1. From Figure 2-12 we can see that the predictions from all the three models are quite close to the actual DTS log. However, it is hard to evaluate the performance of the models by visual examination.

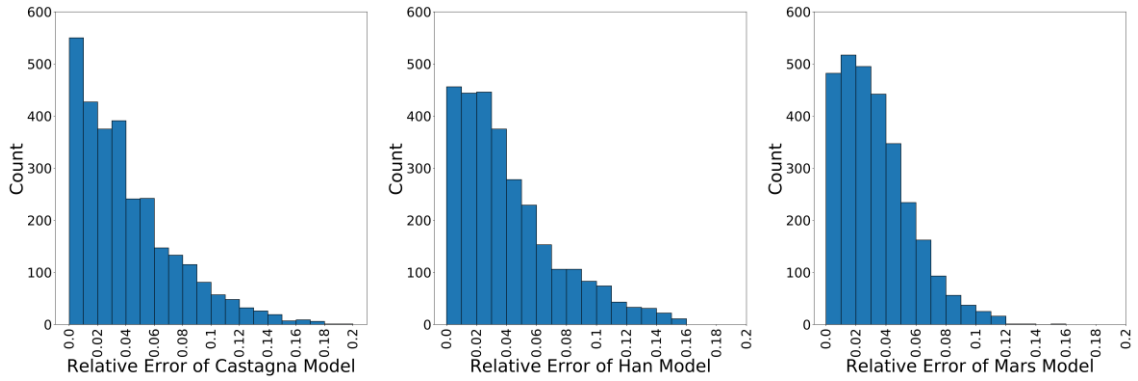


Figure 2-13. Comparison of relative error distribution on DTS logs for the MARS model and empirical models deployed in well #2.

Figure 2-13 plots the distribution of the relative error of the three models. Compared to the MARS model, the two empirical models have a longer 'tail' of the distribution. The maximum relative error of the MARS model is around 0.12, whereas the maximum relative errors of the Castagna and Han models are around 0.18 and 0.16 respectively. The relative error of the Castagna model is more concentrated in the range of 0~0.05. The mean and median of the relative error of the DTS log calculated by the Castagna model are 0.0409 and 0.0325. The mean and median of the relative error of the DTS log calculated by the Han model are 0.0418 and 0.0326. The mean and median of the relative error of the MARS synthesized DTS log are 0.0333 and 0.0292. We can see that the relative error of the MARS model is much smaller than the two empirical models.

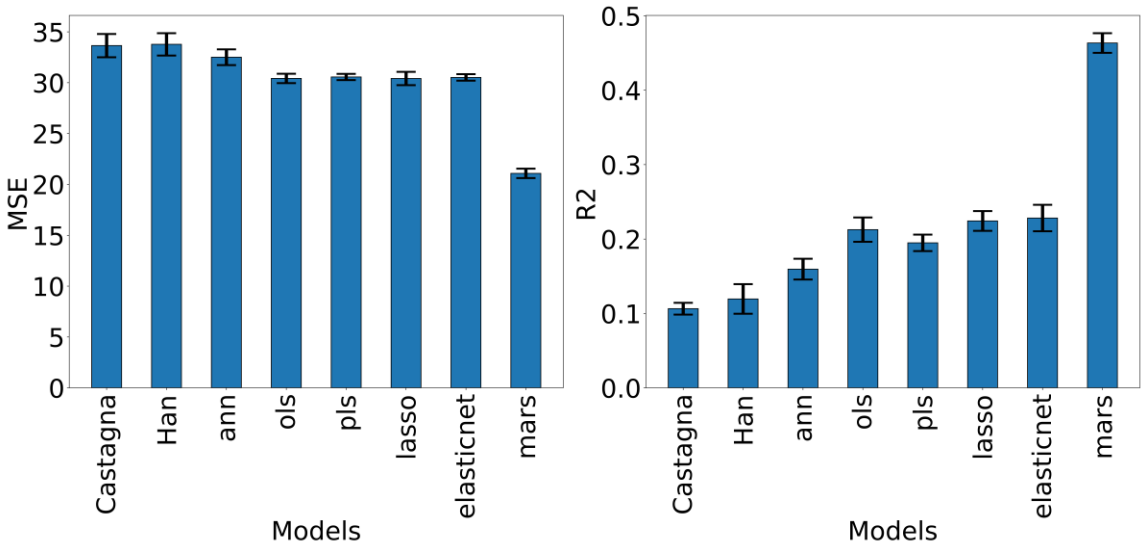


Figure 2-14. Comparison of MSE and R2 (with error bar) of all models deployed in well #2 on the DTS log.

Figure 2-14 plots the MSE and R2 metrics values of all the 6 shallow learning models implemented in the previous section, and the two empirical models implemented in this section. The results indicate that the two empirical models have similar performance. The performance of the empirical models is worse than most of the shallow learning models. MARS model still has the best performance. By comparing the empirical models and the 6 machine learning models, we can see that most machine learning models show better prediction performance than empirical models, which demonstrate the advantages of machine learning models.

2.6.4 Comparison of MARS Model and Multi-depth ANN Model Performance

In this section, two new ANN models are trained and tested with a similar procedure to investigate the possibility of including depth dependencies to improve the prediction accuracy of the DTC and DTS logs. The 6 machine learning models built in the previous section use input data from a single depth to predict the corresponding DTC or DTS log. However, the petrophysical properties of sedimentary rocks are usually

related when the rocks are located in neighboring formations. We assume that the well logs from the neighboring formations may provide useful information in the prediction of the DTC and DTS logs.

The two models implemented in this section follow the same train and test procedure as discussed in section 2.5.3.1 and 2.5.3.2. The train and test dataset split method is changed to take the depth dependencies into consideration. Instead of randomly sampling 70% of data points as the train set, the 70% data points from the upper formations are used as training, and the rest 30% are used as testing. All data points from well #2 are used as a cross-well test set.

The two ANN models implemented in this section are trained to use the new train and test set. The first ANN model is the same as the ANN model implemented in section 2.6.2. The first ANN model is used as a baseline. In the following section, the first ANN model will be referred to as the single-depth ANN model. The second ANN model takes the depth dependency into consideration. The input of the second ANN model is the 8 input logs from three consecutive depths, and the output DTC or DTS log is from the center depth. In other words, the new ANN models use information from three depths to predict DTC or DTS log in a single depth. In the following section, the second ANN model will be referred to as the multi-depth ANN model. The structure of the model is shown in Figure 2-15.

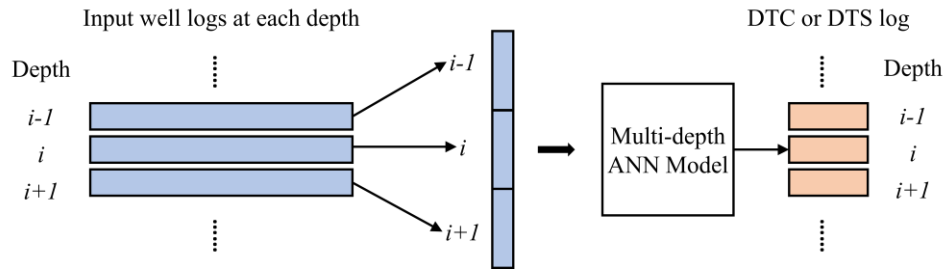


Figure 2-15. Illustration of the multi-depth ANN model prediction process.

The two ANN models are trained with the train set from well #1. The train set is the upper 70% of the dataset. 5-fold cross-validation and grid search are used for hyperparameter tuning. The models' stability is also investigated using the same bootstrap resampling method. The four metrics applied for the 6-shallow machine learning models are applied to the two ANN models. The results are presented in the following sections.

Figure 2-16 compares the prediction results of the multi-depth ANN model and the MARS model on both DTC and DTS data. The first four tracks compare the predicted and actual DTC log of the two models. The last four tracks compare the predicted and actual DTS logs.

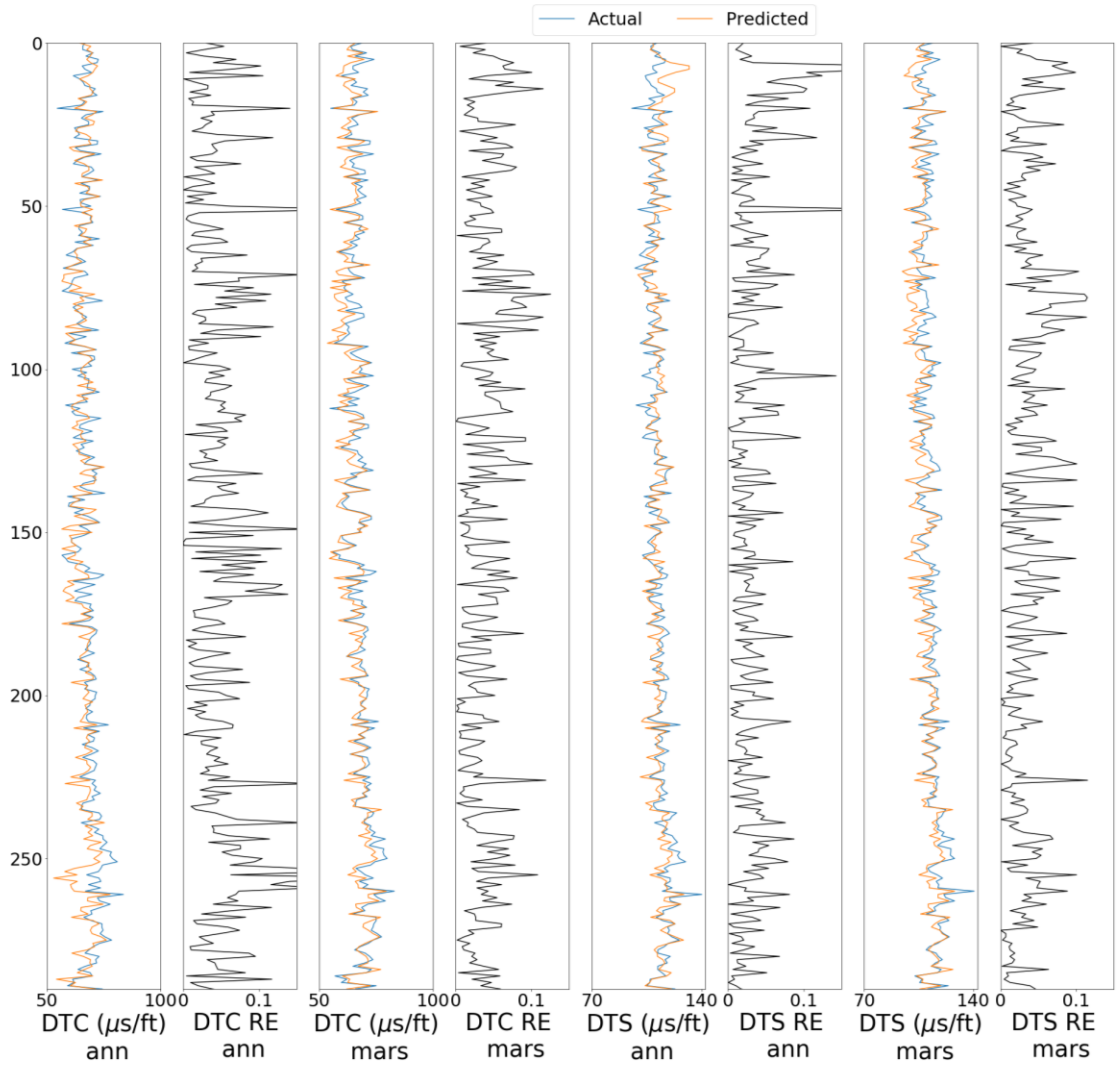


Figure 2-16. Comparison of actual and predicted DTC and DTS logs of multi-depth ANN model and MARS model in Well #2.

Figure 2-17 and Figure 2-18 plots the corresponding relative error distribution of the single-depth and multi-depth ANN models. From the distribution, we can see that the relative error of both models is spread between 0~0.2. Compared to Figure 2-11, fewer data points are located in the range of 0~0.05, which means the models' performance is worse than the MARS model.

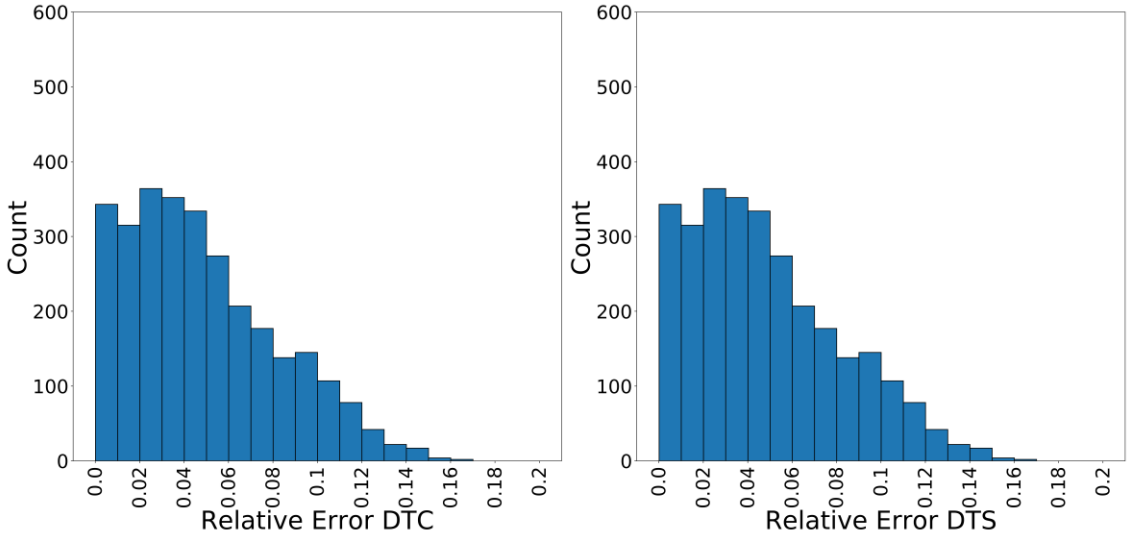


Figure 2-17. Comparison of relative error distribution of DTC (Left) and DTS (Right) logs for the single depth ANN model deployed in well #2.

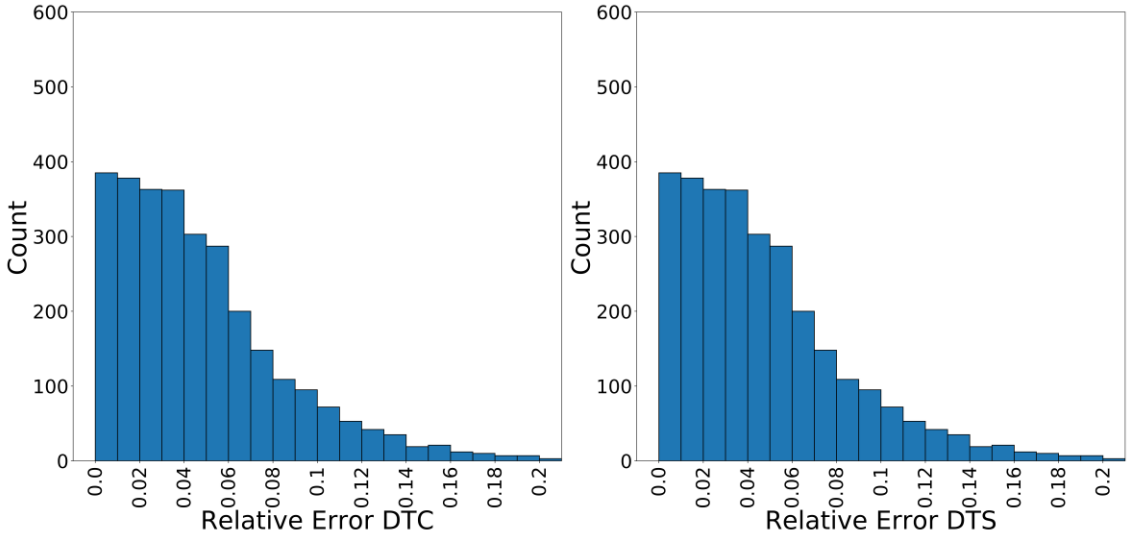


Figure 2-18. Comparison of relative error distribution of DTC (Left) and DTS (Right) logs for the multi-depth ANN model deployed in well #2.

The prediction performance of the single and multi-depth ANN models evaluated using the four metrics is shown in Table A-1. The data in Table A-1 indicates that both single depth and multi-depth ANN models perform worse than the MARS model. We can also refer to Figure 2-8 and Figure 2-9 to compare the metrics of single and multi-depth ANN models with other machine learning models. The bootstrap results indicate that the

single and multi-depth models are much less stable than the other machine learning models.

The confidence intervals of the two models are much larger compared to other models. The large confidence interval of the two ANN models may come from the following two parts. First, the models are only trained with the 70% data points from the upper formation, which means the models do not learn the relationship contained in the lower 30% data points. When exposed to a new data sample, the models may fail to predict the sonic logs. Second, the models are more complicated than the other 6 shallow learning models. With more parameters, the models may become more unstable due to overfitting. The other 6 models do not contain as many parameters, and thus the input-output relationships represented by the other 6 models are relatively simple. More importantly, the relationships between the other well logs and the sonic logs are simple. Many researchers have applied simple empirical models to describe the relationship. Applying complicated models for this simple relationship is not necessary and counterproductive.

By comparing the prediction results of the two models, we can see that when tested with the test set in well #1, the multi-depth ANN model performs better than the single-depth ANN model. Whereas the contrary is observed in well #2. This is due to the train test set split method used in this section. The training dataset is not randomly sampled, which means the train set, the test set, and the test set in well #2 each follow a different distribution. Even though the multi-depth model takes depth dependency into consideration, the multi-depth model still fails to generalize the relationship in the train

set. When exposed to a new dataset, both the single-depth model and the multi-depth model fail.

2.6.5 Factors That Improve the Model Performance

Through this study, I observe that the MARS model outperforms all the other models. Complicated models (like ANN) may not be suitable for this study and consequently, the complexity of the model should be tailored to the problem. The ANN tends to suffer from overfitting and performs poorly when deployed on a test well that was unseen for training.

Finally, selection of an appropriate training dataset is critical. The models should see most of the variability in the data and therefore, instead of selecting either the top or the bottom or middle segments of the data, a random sample acquired from over the entire dataset is more appropriate for training.

2.7 Novelty of this Study

- Different machine learning models are compared with empirical models. The results indicate that machine learning models outperform empirical models.
- This study shows that it is possible to use machine learning to generalize the hidden relationship between 8 easy-to-acquire raw logs and sonic logs.
- Machine learning models are evaluated by both accuracy and stability. The MARS model performs best among all the models.
- This study investigated the possibility of including depth dependency to improve prediction performance. The result indicates that predicting sonic logs with input logs from multiple depths does not improve prediction accuracy.

- This study compared the accuracy of two testing dataset. The test set in well #1 has the same distribution as the train set. The test set in well #2 has a different distribution, and thus the models' performance drop when deployed on well #2. This shows the importance of cross-well testing.

2.8 Assumptions and Limitations of This Study

This study is based on the following assumptions:

- The sonic logs can be estimated from other raw logs, and shallow models using machine learning can capture the relationship between input raw logs and sonic log with good accuracy.
- The relationships between sonic logs and other raw logs are simple, and simple machine learning models can learn these relationships.
- Well logs from different formations follow different relationships between sonic logs and other raw logs. Therefore, a larger training dataset improves the model performance.
- Well logs from different oil wells follow different distributions, which may pose challenges for models trained in a single oil well.
- Cross-well testing represents the real-world performance of the models.
- The properties of formation in neighboring depths are related. Raw logs from neighboring depths provide extra information to estimate the properties of the target formation.
- Model stability measured by bootstrap reflects the model's prediction stability when exposed to a new dataset.

2.9 Recommendations for Future Work

This study performed a thorough comparison of different shallow models developed using machine learning with strict data preprocessing and model training testing procedures. Different kinds of models are tested and compared. The work can be improved in the following ways:

- Train the models with data from more oil wells.
- Perform cross-well testing on different oil wells.
- Investigate the possibility of using depth dependency to improve prediction performance using a recurrent neural network when large dataset is available for training.

2.10 Conclusions

This study applied 6 shallow regression-type supervised-learning models to synthesize the compressional and shear travel-time logs in a shale reservoir. These travel time logs are needed for the geomechanical characterization. The 6 shallow models were trained to process 8 conventional easy-to-acquire logs for synthesizing the compressional and shear travel-time logs. The shallow supervised-learning models exhibit good log-synthesis performance (R^2 in the range of 0.5 to 0.6 in Well #1), and the MARS model is the best performing model. When deployed to a new oil well (Well #2), the MARS outperforms other models with R^2 of 0.63 and 0.59 for DTC and DTS logs. The R^2 for the OLS model is 0.54 and 0.22 for DTC and DTS logs. Apart from that, the stability of the models is tested using the bootstrap resampling method. The results indicate that the ANN model is less stable than other simple models. The 95% confidence interval of R^2 is around 0.1 for ANN model. The 95% confidence interval for other models are almost 0.

For the desired task, R^2 for the empirical models is around 0.12, which is much lower than that of the shallow learning models. Moreover, the depth dependence is investigated by predicting the target sonic logs from input logs at three consecutive depths. However, the results show that including well logs from neighboring formations does not improve the prediction accuracy. For this simple regression problem, the results show that selecting the right model is more important than selecting a powerful model.

2.11 Acknowledgements

The research findings in this chapter are based upon work supported by the U.S. Department of Energy, Office of Science, Office of Basic Energy Sciences, Chemical Sciences Geosciences, and Biosciences Division, under Award Number DE-SC-00019266.

Chapter 3 Deep Neural Network Based Synthesis of NMR T2 Distribution for Pore-Scale Characterization

3.1 Background

3.1.1 Subsurface Characterization

Subsurface characterization involves estimation, computation, and measurement of the physical properties of the subsurface earth formations. Surface-based deep sensing measurements (seismic), borehole-based near-wellbore measurements (logs), and laboratory measurements of geological core samples extracted from wellbores are first acquired and then interpreted using empirical and mechanistic models to quantify the physical properties of the subsurface formations. Nuclear magnetic resonance (NMR) logging tool was introduced for well logging applications in the 1980s. The NMR tool is deployed in a borehole to primarily acquire the T2 distribution responses of the subsurface earth formations intersected by the borehole. The acquired T2 distribution is further processed to obtain the physical properties of the formations, such as pore size distribution, fluid-filled porosity, bound fluid saturations, and permeability. I applied and compared four deep neural networks and several shallow machine learning models to generate the NMR-T2 distribution response of fluid-filled porous subsurface earth formations by processing the conventional and easy-to-acquire subsurface logs in the absence of a downhole NMR logging tool. The proposed synthesis of NMR T2 distribution will enable improved pore-scale characterization, which includes estimations of pore size distribution, bound fluid content, movable fluid content, and permeability, to name a few.

The use of logging tools, geophysical models, and inversion- and machine-learning-based data interpretation techniques for purposes of subsurface characterization has been evolving with the advancements in sensor physics and computational methods. Subsurface measurements along a borehole are generally recorded as well logs acquired using downhole logging tools, which sense the near-wellbore subsurface formations by exciting various physical phenomena in the geological formations and acquiring the resultant response. Subsequently, relevant tool physics modeling and geophysical interpretation models are used to process the logs for purposes of subsurface characterization. For example, electromagnetic short pulse borehole imaging method is used to characterize cracks and rugosity (Guo and Liu, 2010), poroelastic inversion of sonic velocity logs improves permeability characterization (Baron and Holliger, 2011), and triaxial electromagnetic induction measurement facilitates the estimation of dip and anisotropy of the formation (Hong et al., 2014). NMR logs acquired in boreholes are interpreted using inversion methods, mechanistic models, and data-driven models to estimate the pore size distribution, fluid-filled porosity, bound fluid saturations, and permeability of subsurface formations.

Well logs can also be interpreted using neural networks and data inversion techniques to obtain subsurface physical properties. For example, Wong et al. (Wong et al., 1995) classified well log data into different lithofacies followed by the estimation of porosity and permeability using genetic neural networks. Similarly, lithology determination from well logs was performed by Chang et al. (Chang et al., 1997) in Ordovician rock units in northern Kansas using fuzzy associative memory neural network. Recently, advanced neural network architectures have been applied to well log

data for purposes of improved subsurface characterization, such as Huang et al. (Huang et al., 2013).

3.1.2 Objectives of this Chapter

In the petroleum industry, the NMR logging tool is sensitive to the mobility of the pore-filling fluid phases, bound fluid volume, and the pore structure of subsurface hydrocarbon-bearing formations. NMR log contains valuable pore- and fluid-related information that cannot be directly estimated from other conventional logs, such as porosity, resistivity, mineralogy, pore size distribution, and saturation logs. The acquisition of the NMR log is more expensive and requires better well conditions to deploy infrastructure than other conventional logs. The NMR logging tool is usually run at a relatively slower speed of 2000 ft/hr in bigger diameter uniform boreholes around 6.5-in in diameter. Dielectric logging tool, which is also an advanced logging tool, can be run at a speed of 3600 ft/hr in 5.5-inch diameter boreholes at lower costs. NMR signals due to the inherent physics have a poor signal-to-noise ratio (Freedman, 2006). The acquisition of a high-quality NMR log in a subsurface borehole environment requires highly trained wireline field engineers. Oil and gas companies do not deploy an NMR logging tool in every well in a reservoir due to the financial and operational challenges involved in running the NMR tool. One alternative is to predict the entire NMR-T2 distribution spanning 0.3ms to 3000ms from conventional logs. An accurate prediction of NMR T2 distribution will assist the geoscientists and engineers to quantify the pore size distribution, permeability, and bound fluid saturation in hydrocarbon-bearing reservoirs; thereby improving project economics and reservoir characterization capabilities. Prediction of the entire NMR T2 without core data is a challenging and novel task.

3.1.3 Deep Neural Networks for NMR T2 Synthesis

Deep learning methods are advanced machine learning (ML) techniques for data processing, information retrieval, pattern recognition, and diagnostics. The use of deep learning algorithms is getting adopted in remote sensing applications.

This study applies novel deep-learning-assisted methods to synthesize the NMR-T2 distribution response of fluid-filled porous subsurface earth formations around the wellbore by processing the conventional and easy-to-acquire subsurface measurements in the absence of a downhole NMR logging tool.

Four types of deep learning neural networks are applied for the purpose of NMR T2 synthesis. The four models include: Variational-Autoencoder (VAE) assisted neural network (VAE-NN), Generative Adversarial Network (GAN) assisted neural network (GAN-NN), Long Short-Term Memory (LSTM), and Variational Autoencoder with a convolutional layer (VAEc) network. NMR T2 data has both spatial and time characteristics. NMR T2 data is a 64-dimensional vector, which contains spatial features. Besides, the NMR T2 data contains T2 relaxation times that are inverted from raw NMR logs. These two characteristics inspired us to use neural networks that suitable to deal with spatial data (VAE, VAEc), and neural networks that suitable to deal with time series (LSTM).

The four models mentioned before are implemented to extract the complex relationships between the NMR-T2-distribution log and other easy to acquire well logs. Two types of logs are used as inputs: 12 easy-to-acquire raw logs and 13 inverted formation mineral and fluid composition logs. 12 raw logs are used as input data comprising 5 array induction four foot resistivity log (AF10, AF20, AF30, AF60, AF90),

caliper (DCAL), compressional sonic (DTCO), shear sonic (DTSM), gamma-ray (GR), Neutron porosity (NPOR), PEFZ, and formation density (RHOZ). The inverted formation compositional logs are formation fluid and mineral compositions inverted from resistivity, neutron, density, GR, and dielectric. 10 inverted formation mineral composition logs of anhydrite, calcite, chlorite, dolomite, halite, kerogen, illite, k feldspar, montmorillonite, quartz, and 3 formation fluid compositional logs of free water, free oil, and bound water are used as the input of the synthesis process. The 12 raw logs and 13 formation mineral and fluid composition logs are not all used as inputs. Some logs are removed in the dimensionality reduction process due to high correlations. The four models applied in this study contain unsupervised ML algorithms with generative abilities. The training of the models requires exposure to a limited NMR-T2 distribution dataset prior to the prediction.

An autoencoder is a type of deep neural network that is trained to reproduce its input as its output by implementing an encoder followed by a decoder (Goodfellow et al., 2016). On the encoder side, it has a latent layer of lower dimension compared to the preceding layers. The encoder projects the input data to the latent layer, following that decoder decodes the latent vector encoded in the latent layer. With this bottleneck structure, an autoencoder learns to extract the most important information when the input goes through the latent layer. Therefore, an autoencoder can be taken as an effective way to project data from a high dimension to a lower dimension by identifying the most dominant features and characteristics. A variational autoencoder is a specific form of the autoencoder. The latent vectors are constrained to follow a Gaussian distribution (Kingma and Welling, 2013), which adds uncertainty to the latent variable. VAE arranges

the learned features with similar shapes close to each other in the projected latent space; thereby, reducing the loss in the reproduction of input.

A GAN is composed of two neural networks, generator, and discriminator, aiming to learn from training data and produce data that are similar to the training data (Goodfellow et al., 2014). GAN has shown great potential in image generation (Radford et al., 2015) and text to image synthesis (Reed et al., 2016). In this case, the generator learns to generate T2 distribution that cannot be detected by the discriminator as to whether the generated data is from measured training dataset or synthetically generated by the generator. With a proper architecture and training process, the generator will gain the ability to generate data that are very similar to those in the training dataset. GAN-NN learns from NMR T2 through the formation under investigation, and then make a prediction of T2 based on matrix composition and fluid saturations.

A CNN (Convolutional Neural Network) is inspired by the complex arrangement of the biological visual cortex, where a set of neurons are sensitive to a specific visual feature, such as angle, curvature or edge. A convolutional layer in CNN contains a combination of mathematical filters, in which each filter is sensitive to a specific feature of an image. When multiple convolutional layers are combined, CNN can learn features of varying visual complexity. A Variational Autoencoder is a special type of generative neural network that aims to reproduce the input of the network. It comprises two computational frameworks, an encoder followed by a decoder. The encoder encodes the input into a latent layer, whereas the decoder decodes the information in the latent layer to reconstruct the input. As a generative model, the VAE has been applied in various generative modeling problems like image generation (Dosovitskiy and Brox, 2016) and

text generation (Bowman et al., 2015). We assume that combining VAE and convolutional improves the learning ability of VAE.

A LSTM is a type of recurrent neural network (RNN) with an improved ability to solve the gradient vanishing problem in RNN. With the ability of 'memorizing' values for a long time, the LSTM is superior in dealing with sequences, like text, where values in a different position may have a strong dependency. LSTM applications include machine translation(Cho et al., 2014; Sutskever et al., 2014), natural language generation(Wen et al., 2015), and time-series prediction. The joint application of LSTM and CNN can accomplish the work of video description (Venugopalan et al., 2014), image captioning (Vinyals et al., 2015), and text to image generation (Reed et al., 2016). NMR T2 distribution is the distribution of the T2 relaxation time. We assume that the LSTM can deal with the dependencies of the NMR T2 relaxation time values batter than other models.

The NMR-T2 synthesis processes of VAE, VAEc, and GAN-NN are similar. First, a deep neural network is pre-trained to extract the features of the NMR-T2 distribution. NMR T2 distributions have inherent characteristics. All NMR T2 distributions are similar to combinations of Gaussian distributions. In this step, the models can learn the general characteristics of NMR T2. Second, a simple multilayer neural network is connected with the pre-trained deep neural network to predict NMR-T2 distribution from conventional logs. The two-step training process makes the prediction results stable. Instead of predicting each value of the NMR T2 distribution, our designed neural network predicts the complete NMR-T2 spectra; thereby providing greater constraint to the synthesis. To the best of our knowledge, no similar study has been

attempted to synthesize NMR-T2 distribution prior to our work published in various journals(Li and Misra, 2017a; Li and Misra, 2017b; Li and Misra, 2018; Li et al., 2019; Misra and He, 2019a; Misra and Li, 2019). Six shallow models are also trained and tested with the same data to compare with the deep models.

3.2 Motivations for this Study

NMR T2 is a valuable well log that contains information of formation porosity, permeability, pore size distribution, etc. The acquisition of the NMR log is more expensive and requires better well conditions to deploy infrastructure than other conventional logs. The acquisition of a high-quality NMR log in a subsurface borehole environment requires highly trained wireline field engineers. Oil and gas companies do not deploy an NMR logging tool in every well in a reservoir due to the financial and operational challenges involved in running the NMR tool. NMR T2 synthesis with machine learning models may provide engineers with an NMR T2 log while it is not available.

Machine learning, especially deep learning models have been successfully applied to problems like image synthesis, language generation, etc. Prediction of the entire NMR T2 without core data is a challenging and novel task. By applying machine learning models for NMR T2 synthesis, we hope to provide more information for engineers for the purpose of formation characterization by utilizing the power of machine learning.

3.3 Key Fundamental Questions to be Answered

- Can deep learning methods be used to generate the NMR T2-distribution responses of the near-wellbore region by processing easy-to-acquire well logs under data constraints?

- How to evaluate the generalization capability of data-driven models developed using various deep-learning architectures for the purpose of generating NMR T2-distribution under data constraints? What metrics are well suited for the evaluation of generalization capability?
- Are raw logs better than inverted/processed logs for NMR T2 synthesis under data constraints?
- Why certain deep learning architectures are better suited for NMR T2 synthesis under data constraints?
- Why certain depths are better suited for NMR T2 synthesis using deep-learning models under data constraints? For what depths do the deep learning models fail?
- What are the major limitations and drawbacks of developing deep learning models for NMR T2 synthesis under data constraints?
- Do shallow models perform better than deep models for the purpose of NMR T2 synthesis under data constraints? For what depths do the shallow learning models perform better than the deep learning models?

3.4 Data Description and Preparation

3.4.1 Description of Geology

A set of well log data were retrieved from a shale petroleum system comprising seven formations as shown in Figure 3-1. The top three formations (F1-F3) constitute a shale formation and the bottom four (F4-F7) are dolostone interbedded with clay-rich conglomeratic dolo-mudstone of Devonian age. F1 is an upper black shale, F2 is a middle sandy siltstone, and F3 is a lower black shale. Formations F1 and F3 are hydrocarbon source rocks that are organic-rich with total organic carbon ranging from 12-36 weight

percent. The clay mineral content is dominated by illite and quartz. Formation F2 is the hydrocarbon-bearing reservoir and has a low total organic carbon (TOC) content ranging from 0.1 to 0.3 wt% (weight percentage). Variation in formation mineral compositions leads to changes in the pore structure, grain texture, and surface relaxivity. These characteristics along with fluid saturations and their distribution in the pore network govern the NMR T2 distribution. Mineral compositions and fluid saturations in the seven formations were obtained by numerical inversion of resistivity, neutron, density, gamma-ray (GR), and dielectric logs using the mineral inversion module in TECHLOG. 12 raw logs and 13 formation mineral and formation fluid composition logs are acquired in the formation.

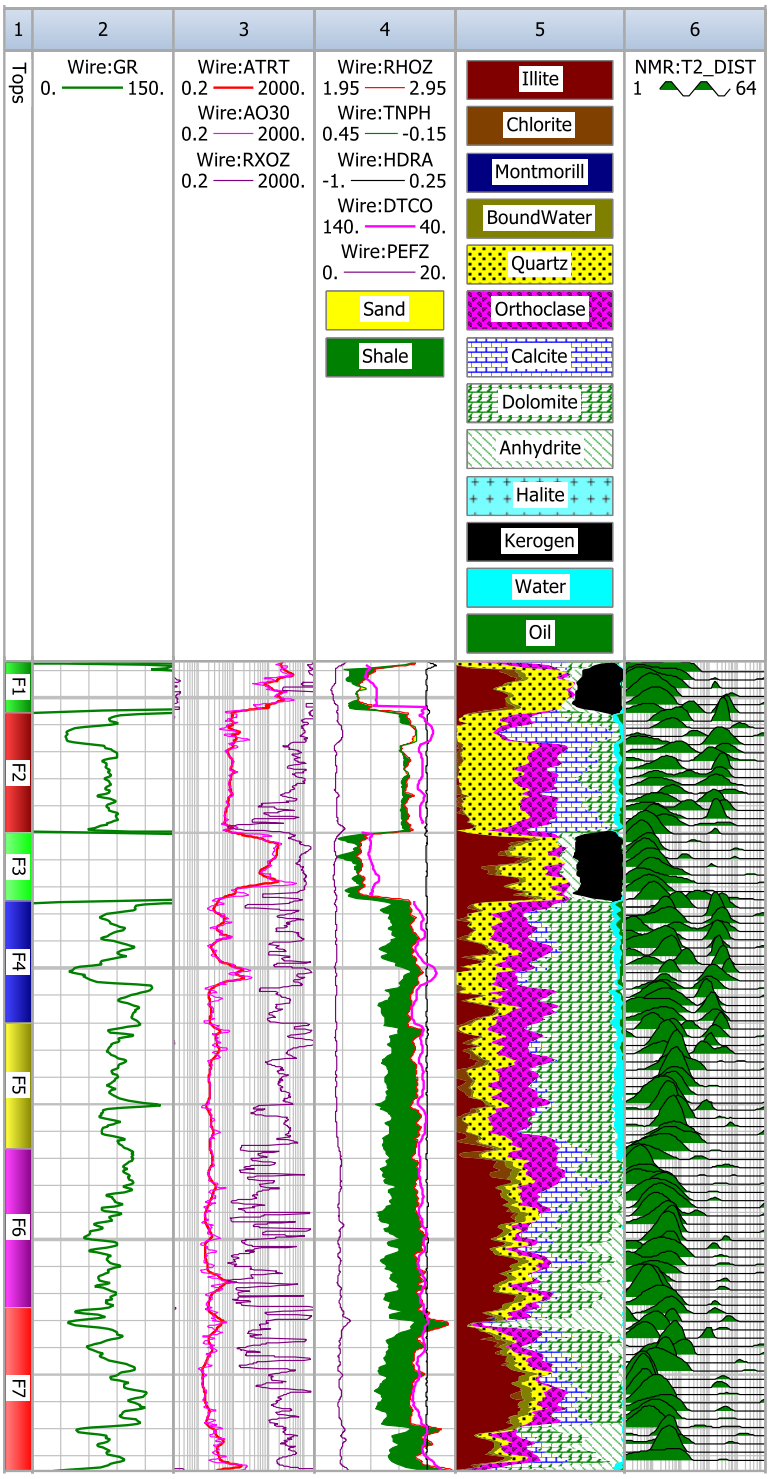


Figure 3-1. Well logs acquired in the shale petroleum system.

3.4.2 Data Preparation and Processing

Data preparation, or data preprocessing, affects the convergence time during the training phase. Each input has a different range, for example, illite volume fraction ranges from 0.1-0.5 m3/m3, whereas water and oil volume fraction is below 0.1 m3/m3. Each input data is scaled to have zero mean and unit variance to aid the loss function reach the global optimum during training by gradient descent algorithm. Output data does not require scaling or normalization. We removed the unrepresentative depths with three peaks in NMR T2 that were not conducive to prediction performance. Outliers were screened and removed by fitting NMR T2 with a cubic spline and then detecting those with three peaks and unusual characteristics of the fit. Depth shift was performed to ensure that inputs and their output corresponded to the same depth. Detailed data preprocessing and modeling training processes are shown in Figure 3-2.

3.4.2.1 Removing Outliers

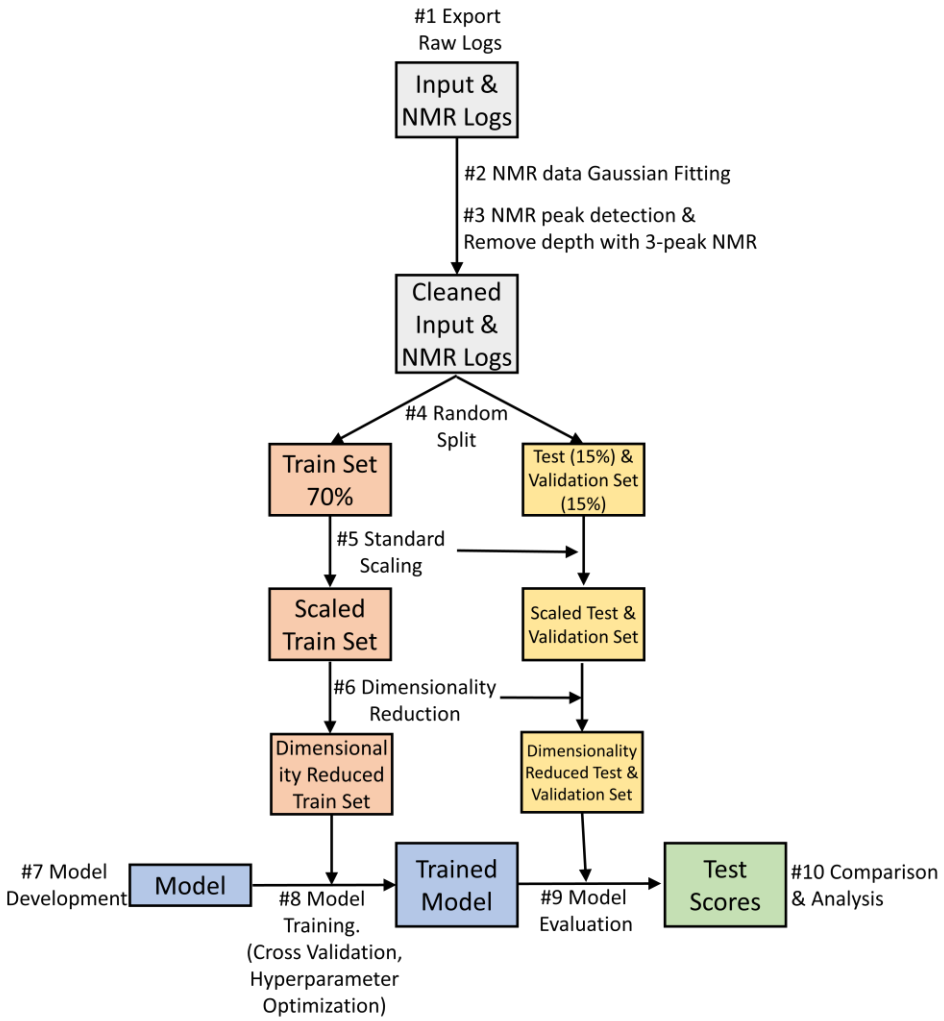


Figure 3-2. Investigation methodology flowchart.

The proposed models are trained and evaluated with the workflow shown in Figure 3-2. First, the input logs (raw logs or formation mineral and fluid composition logs) and output logs (NMR T2) are exported from the database. The formation under investigation is a 300-ft shale formation. The well logs are acquired every 0.5 ft. Around 600 data points are exported. Based on our previous experience on NMR T2 synthesis, we found that models do not perform well on NMR T2 with 3 peaks. This is because most of the NMR T2 data are single peak or double peaks. The dataset does not provide

enough 3-peak NMR T2 data points for models to learn from. So, we removed depth with 3-peak NMR T2 by fitting NMR T2 with Gaussian distributions and detect 3 peak NMR T2 data. Data points from 28 depths were identified as 3-peak NMR T2. Our methods are aligned with Misra et al.'s (2019) investigations on outlier detection methods suitable for geophysical data (Misra et al., 2019c).

3.4.2.2 Train-test Split

The cleaned data is split randomly into training, validation, and testing dataset by ratios of 0.7, 0.15, 0.15. The training dataset is used to train machine learning models, the validation set is used for hyperparameter tuning and overfitting prevention, and the test dataset is only used to test the best-performing model (Figure 3-2). 400 data points are randomly selected as training data, 80 are selected as validation, another 80 data points are selected as testing dataset. Both raw logs and inverted logs are processed with the same method.

3.4.2.3 Scaling and Dimensionality Reduction

Standard scaling and dimensionality reduction are performed on the training dataset. The standard scaling and dimensionality reduction are also performed on validation and testing datasets by using exactly the same parameters acquired from the training dataset. The standard scaling scales each feature with zero mean and unit variance. The dimensionality reduction is performed by removing logs that have high correlations with other logs. For the 12 raw logs, the 5 array induction four-foot resistivity logs (AF10, AF20, AF30, AF60, AF90) show a high correlation with each other. Only the AF10 log is kept as input. The compressional sonic (DTCO), shear sonic (DTSM), and gamma-ray (GR) log also show high correlations with each other, only

DTSM log is kept. After dimensionality reduction, only 6 raw logs are kept as input logs. For the 13 formation mineral and fluid composition logs, the halite log is removed because of zero variance; bound water log is removed because it is correlated to illite log and free water log. After dimensionality reduction, 11 out of the 13 logs are used as input logs. By removing well logs that have a high correlation with others, we can reduce the input dataset while keeping most of the information. Raw logs show higher correlations with each other compared with inverted compositional logs. More discussions on dimensionality reduction and feature scaling is presented in the study conducted by Misra and Chakravarty et al. (2019) (Misra et al., 2019a).

3.4.2.4 Model development

In this study, neural networks are built on the high-level machine learning platform Keras, with TensorFlow as the backend. TensorFlow is an open-source software developed by Google for machine learning. Keras can run on top of TensorFlow. Keras allows users to easily construct various kind of neural networks. Combining TensorFlow and Keras reduces the complexity of building a deep neural network model while maintaining high training and testing efficiency.

Different machine learning models are developed with different architecture and parameters. Both shallow models and deep models are tested and compared. The only difference in the synthesis procedure is the model. The data and evaluation methods are the same for all the models applied. The models are trained with training dataset, tuned and optimized with the validation dataset. For the deep models, a performance monitor is set to monitor the training and validation loss. Early stop callback is set to stop the training when overfitting is observed. The hyperparameters are tuned by grid search on

the validation dataset. Different combinations of hyperparameters of each model are tested on the validation set. The best-performing models are tested with the testing dataset. For shallow models, 3-fold cross-validation, and grid search are used for hyperparameter tuning. Details of the model architecture and training procedures are described in the following sections. More details on hyperparameter optimization is presented in the study conducted by Wu and Misra et al.(2019) and model development is presented in the study by Misra and Ganguly et al. (2019) (Misra et al., 2019b; Wu et al., 2019)

3.4.2.5 Model Evaluation Metrics

The synthesis accuracy is tested using the testing data. R^2 values are calculated using the predicted NMR T2 and measured NMR T2 distributions, such that

$$\boldsymbol{R^2(y,\hat{y}) = 1 - \frac{\sum_{i=1}^{n}(y_i-\hat{y}_i)^2}{\sum_{i=1}^{n}(y_i-\bar{y})^2}} \quad (3\text{-}1)$$

where $\hat{y}_i$ and y_i are predicted and actual NMR T2 amplitude at bin *i*, and *n* is the number of bins.

R^2 values evaluate if the synthesized NMR T2 is overlap with the true NMR T2. However, another important aspect to evaluate the synthesis results is to evaluate the smoothness of the NMR T2. The real-world NMR T2 are smooth lines with one or two peaks, which are similar to multiple combinations of Gaussian distribution. I utilized bending energy from Euler–Bernoulli beam theory to evaluate the smoothness of a curve (Van Vliet and Verbeek, 1993), which is represented as:

$$\boldsymbol{E = \frac{1}{2}\int_{x_0}^{x_n} y''(x)^2 dx} \quad (3\text{-}2)$$

where E is the bending energy of a curve, $y''(x)$ is the second derivative of the points on the curve. Here the second derivatives are approximated by cubic spline interpolation.

The basic idea behind the smoothness measurement method is to take the curve as an elastic beam. The smoother a curve is, the less bending energy it needs to bend the curve to go through all the points. The NMR T2 curve is represented by 64 data points, if the 64 points are smoothly located on a curve, placing an elastic beam going through all the 64 points should cost little bending energy. However, when the 64 points are not smooth, placing an elastic beam going through all the 64 points will cost a lot of bending energy. The study conducted by Misra and He explores various aspects of evaluation metrics (Misra and He, 2019b).

3.5 NMR T2 Synthesis using Shallow Models

In this section, I applied 6 shallow regression models for NMR T2 synthesis. Ordinary least squares models (OLS), least absolute shrinkage and selection operator (LASSO), and ElasticNet are simple linear regression models. Support vector regression (SVR) is an extension from the support vector machine (SVM) classification algorithm. The SVR model can only predict one target. In our study, the NMR T2 distribution has 64 dimensions, i.e. the NMR T2 distribution is a 64-dimensional vector. To predict the 64-dimensional NMR T2 distribution, 64 models are trained to predict one value at each T2 time. K-nearest neighbor regressor is an extension based on the k nearest neighbor classification methods. The neural network applied here is the basic multilayer perceptron. Each of the models is optimized by 3-fold cross-validation. Grid search is also applied for hyperparameter tuning when possible.

3.5.1 OLS

OLS is the simplest linear regression model. It minimizes the squared error between predicted and actual values:

$$\min_{\boldsymbol{\theta}} \|\boldsymbol{X\theta} - \boldsymbol{Y}\|_2^2 \tag{3-3}$$

where $\boldsymbol{X}$ is the matrix of inputs of 400 samples and each sample has 11 logs (for compositional logs). $\boldsymbol{Y}$ is the matrix of 400 output NMR T2 distributions. Each NMR T2 distribution is a 64-dimension vector. $\boldsymbol{\Theta}$ is the weight matrix, which contains parameters learned from data. We can see the OLS model is very similar to a simple neural network with no activation function and hidden layers.

3.5.2 LASSO

LASSO model is an extension of the OLS model. Based on the OLS model, the LASSO model added a regularization term in the cost function:

$$\min_{\boldsymbol{\theta}} \frac{1}{2n} \|\boldsymbol{X\theta} - \boldsymbol{Y}\|_2^2 + \boldsymbol{\alpha}\|\boldsymbol{\theta}\|_1 \tag{3-4}$$

where α is a constant to balance the regularization term and the cost between predicted and actual target, *n* is the number of samples. The added regularization term act similar to the regularization operation in the neural network. The regularization term reduces the number of variables in some contexts.

3.5.3 ElasticNet

ElasticNet is an extension of LASSO. Instead of adding an L1 norm of the weight, ElasticNet added both L1 norm and L2 norm of the weight to the cost function.

$$\min_{\boldsymbol{\theta}} \frac{1}{2n} \|\boldsymbol{X\theta} - \boldsymbol{Y}\|_2^2 + \boldsymbol{\alpha\rho}\|\boldsymbol{\theta}\|_1 + \frac{\boldsymbol{\alpha}(1-\boldsymbol{\rho})}{2} \|\boldsymbol{\theta}\|_2^2 \tag{3-5}$$

EasticNet added the L2 norm and use ρ term to balance the L1 and L2 terms. ElasticNet if more suitable for a dataset where multiple features are correlated with each other.

3.5.4 SVR

Support vector regression is based on the classification algorithm of the support vector machine. SVM classifies the dataset by maximizing the margin between two classes in high dimensional space. Based on SVM, the SVR algorithm tried to minimize the mismatch of the predicted and actual target values in high dimensional space while keeping the mapping function as smoothly as possible:

$$\textbf{minimize } \frac{1}{2}\|\boldsymbol{w}\|^2 + \boldsymbol{C}\sum_{i=1}^{n}(\xi_i + \xi_i^*) \tag{3-6}$$

$$\textbf{subject to } \begin{cases} y_i - \boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x_i}) - \boldsymbol{b} \le \varepsilon + \zeta_i \\ \boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x_i}) + \boldsymbol{b} - y_i \le \varepsilon + \zeta_i^* \\ \zeta_i, \zeta_i^* \ge 0 \end{cases} \tag{3-7}$$

ϕ is a function maps input dataset from current space to higher-dimensional space. The mapping can be achieved implicitly using kernel function. In this study, I use the radial basis function (RBF) kernel. ε is the error we can tolerate in high dimensional space. ξ_i, ξ_i^* are slack variables introduced if the optimization in high dimensional space with an error limit of ε is not feasible. The goal of the SVR is to keep the function smoothing by minimize the norm of the weights and minimize the slack variables. The SVR algorithm can only work with one target variable. I trained 64 different SVR models and each model predicts a single value of the NMR T2 vector.

3.5.5 K-neighbors regression

K-neighbors regression is based on the k nearest neighbors classification algorithm. The K nearest neighbors classification algorithm classifies new data points by computing the majority vote from the k nearest neighbors. K-neighbors regression algorithm extended from the classification algorithm and takes the regression problem as a classification problem where the target classes are labeled with continuous values. The

target is calculated as a weighted average of the k nearest neighbors. The weights are equal for the neighbors. The K-neighbors regression algorithm does not build a model. It calculates the output vector as a weighted sum of the nearest neighbors in the existing dataset.

3.5.6 Artificial Neural network

The artificial neural network (ANN) is widely used for dataset regression due to its simplicity and flexibility. Basically, ANN is a function approximator similar to OLS, LASSO, and ElasticNet models. ANN update weights by backpropagation of errors during training iterations. In this study, I built four-layer ANN with three hidden layers. Each hidden layer has 500 units. The number of layers and the number of units are optimized by grid search.

3.5.7 Results

The 6 models implemented are shallow simple regression models. In this section, the prediction accuracy and the running time of each model are compared and evaluated. Some of the models need adjusting of the parameter to achieve better prediction accuracy. I use the grid search to run the algorithm with a combination of a range of different parameters and record the performance. 3-fold cross validation is also used to tune the hyperparameters. In this way, I can select the best parameter for each model. Figure 3-3 shows the R^2 and MSE (mean squared error) values between real NMR T2 distribution and synthetic NMR T2 from shallow models. Table 3-1 shows the program running time of the shallow models. The R^2 and running time are evaluated for both inverted formation compositional logs and raw logs.

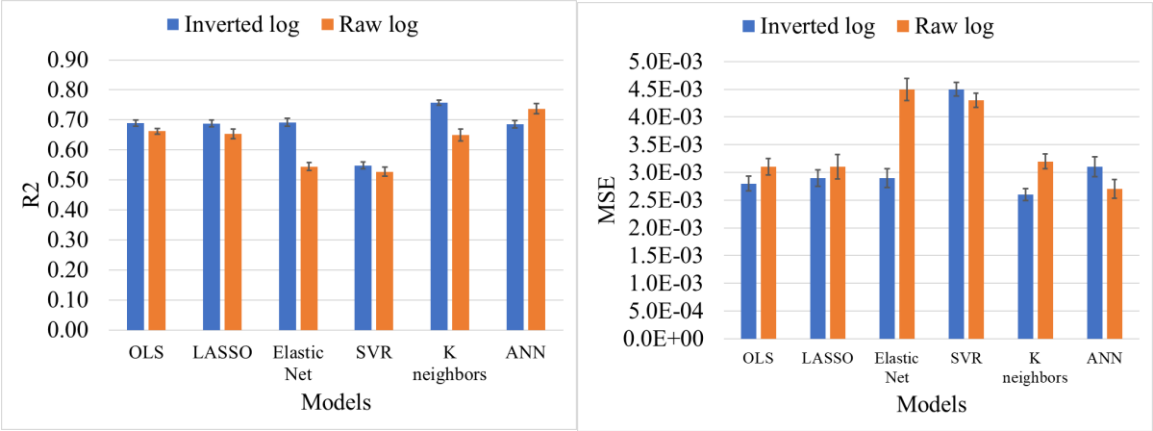


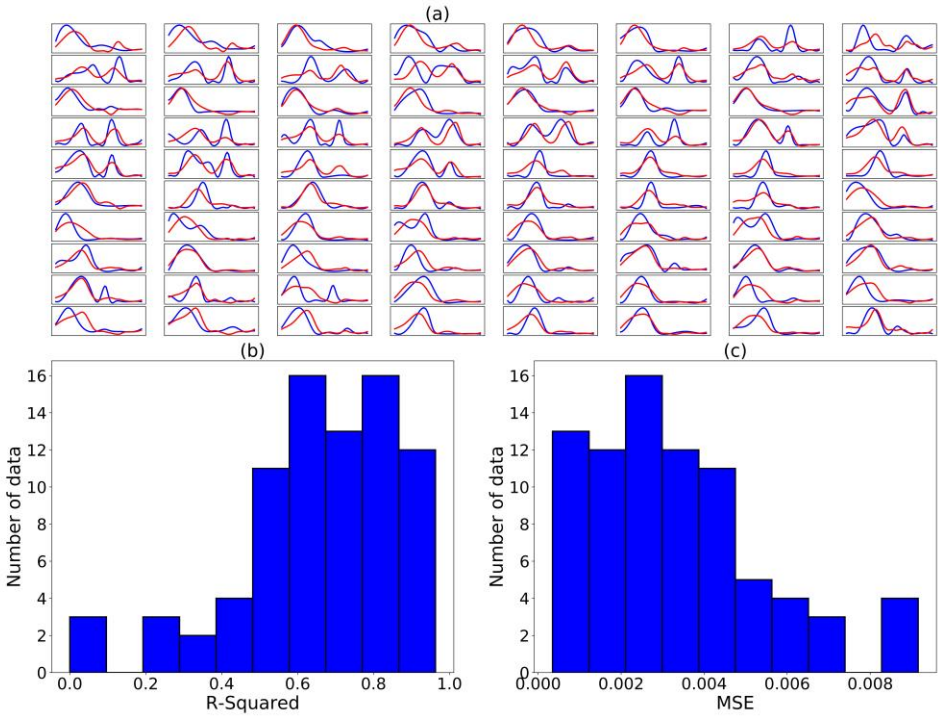
Figure 3-3. Median R^2 (Left) and MSE (Right) values (with 95% confidence interval) for synthetic and real NMR T2 distribution for different shallow learning models.

Figure 3-3 shows that the accuracies of the 6 models are quite similar. Most of the R^2 values range from 0.60 – 0.70. For OLS, LASSO, and ElasticNet models, they exhibit similar synthetic accuracy. These three models are similar linear regression models. When I applied these three models, I applied the grid search to try combinations of parameters and record the R^2 values of the validation dataset in order to find the best-performed parameters. Our grid search experiment shows that the best performing parameters for LASSO and ElasticNet are actually leading the models similar to the OLS model. The parameter α, which controls the regularization term, performs best when it close to 0. This lead the LASSO and ElasticNet model to perform similarly to the OLS model. The SVR model does not perform well on inverted well logs and raw logs. The K-neighbor algorithm does not build a parametric model, it just outputs the average of k nearest neighbors. So, the K-neighbor algorithm never makes big mistakes. Surprisingly, the K-neighbors model seems to be the best-performing model among the 6 shallow models on inverted logs. ANN model has a relatively higher capacity compared to other models, the R^2 values are good compared to other models. However, the NMR T2 generated by ANN lost the inherent features.

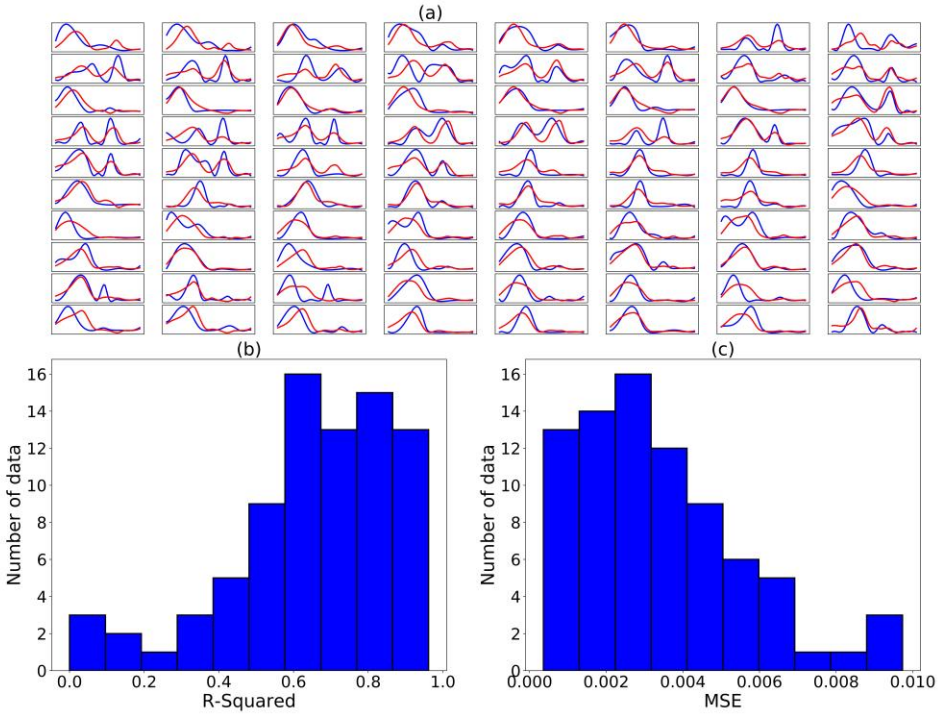
Table 3-1. Running time (s) for different shallow learning models.

	OLS	LASSO	ElasticNet	SVR	K-neighbors	ANN
Inverted log	0.002	0.33	0.009	0.33	0.0010	0.32
Raw log	0.005	0.34	0.009	0.21	0.0005	0.20

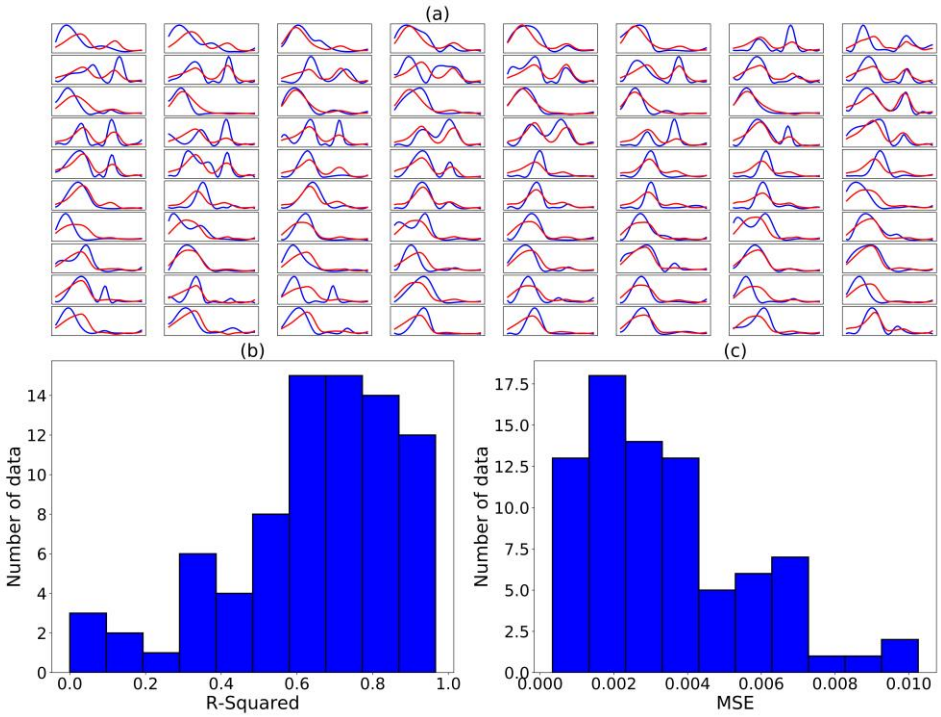
The running time of each algorithm ranges from 0.001s to 0.3s. The models are run on the same Dell workstation with Inter 6 core Xeon CPU and 32 GB RAM. K-neighbors algorithm is the fastest algorithm since it does not has a model. ANN, SVR, and LASSO are slower compared to the other three models. The running time does not differ greatly with different input logs.



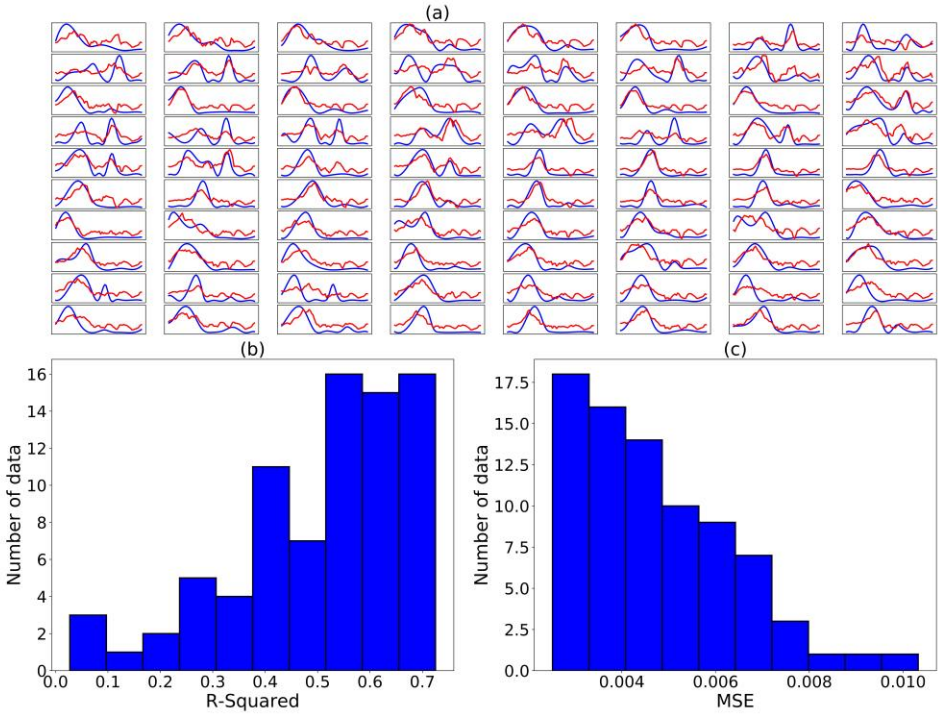
(a) OLS



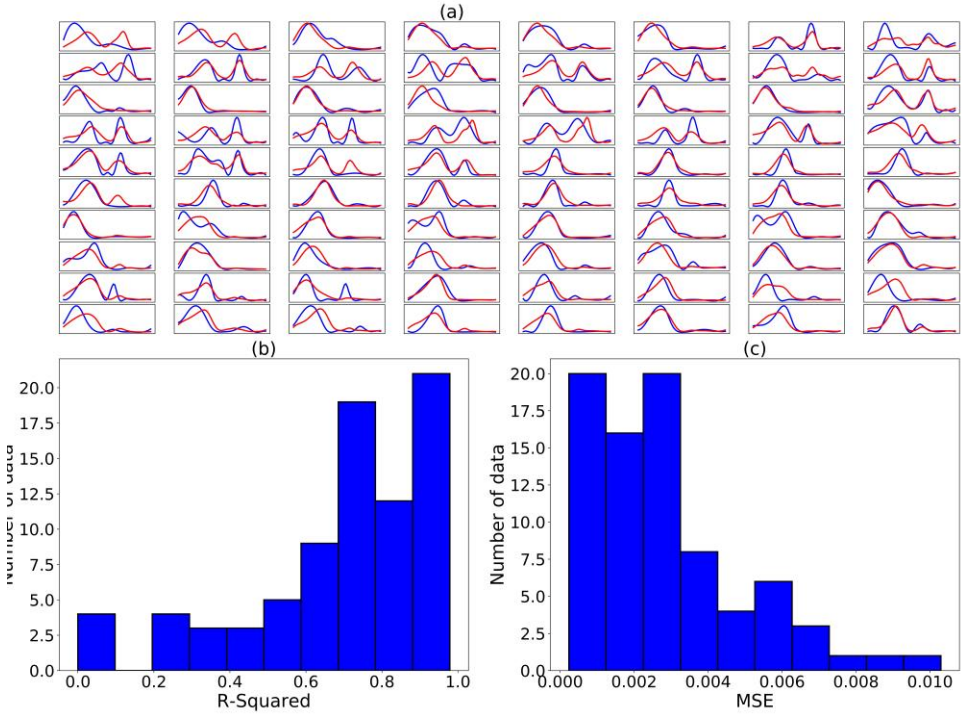
(b) LASSO



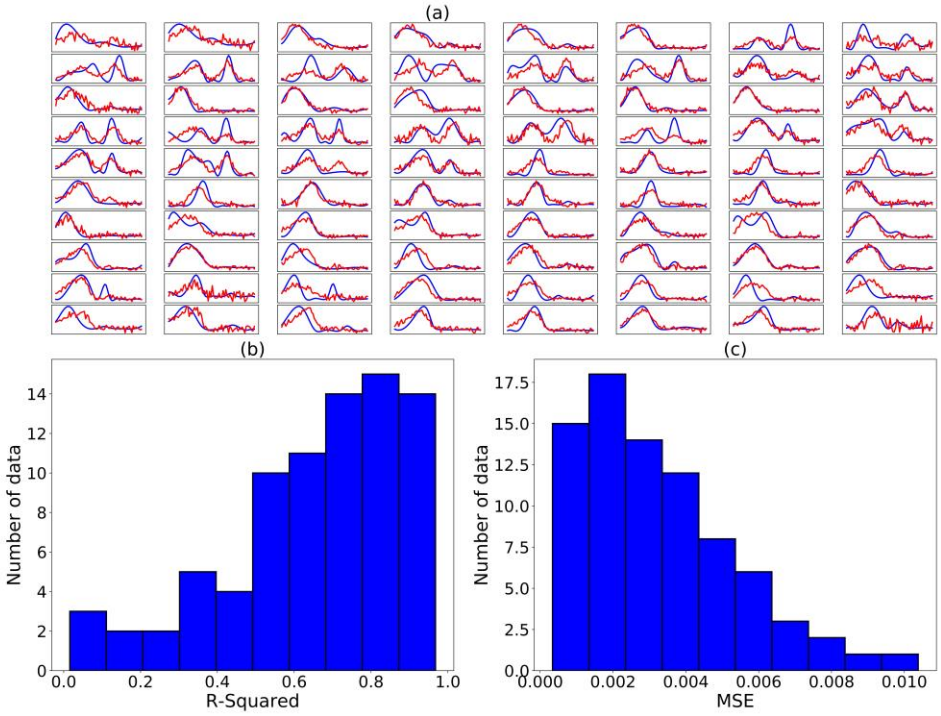
(c) ElasticNet



(d) SVR

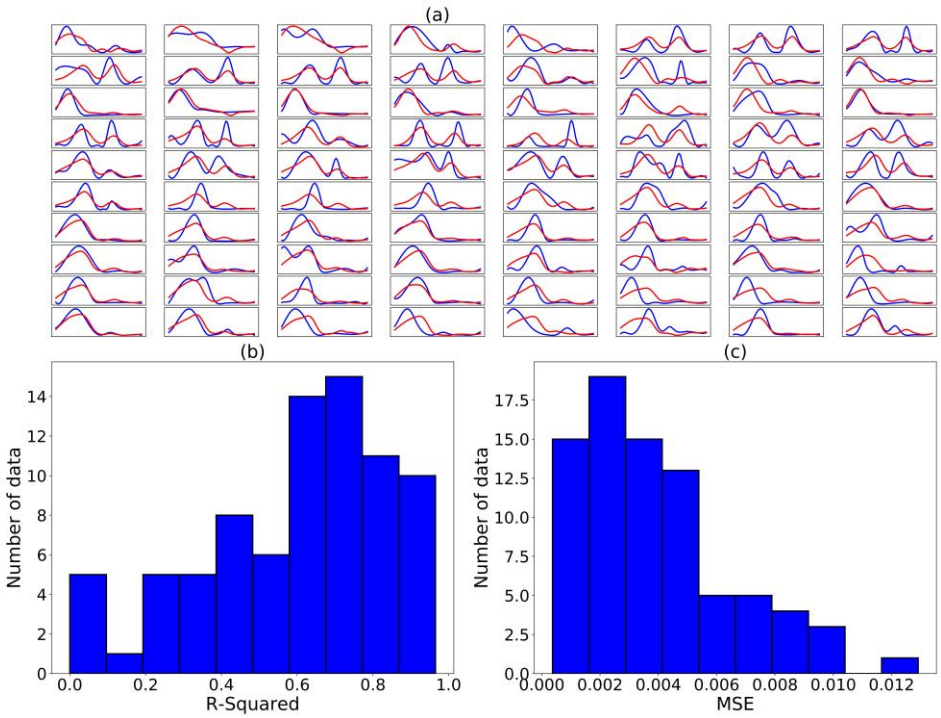


(e) K-neighbors

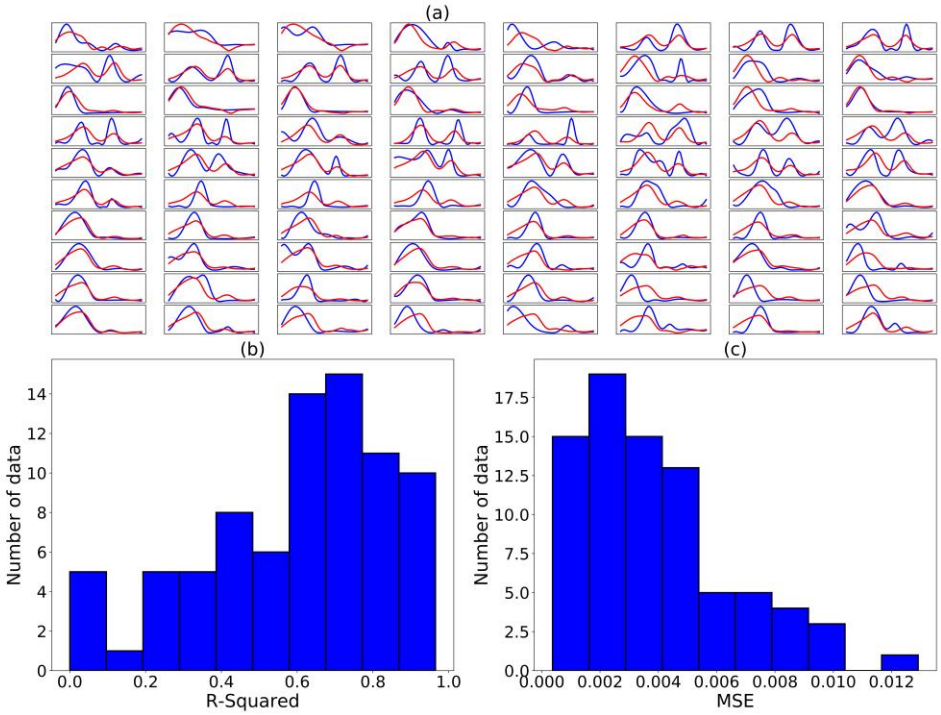


(f) ANN

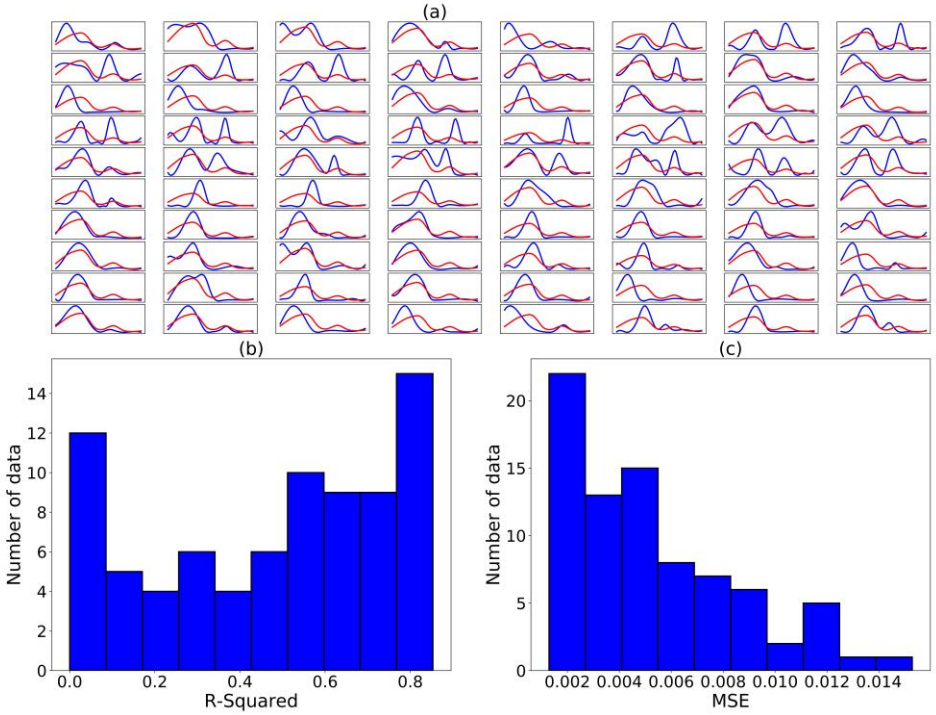
Figure 3-4. Comparison of synthetic NMR T2 and real NMR T2 using inverted logs.



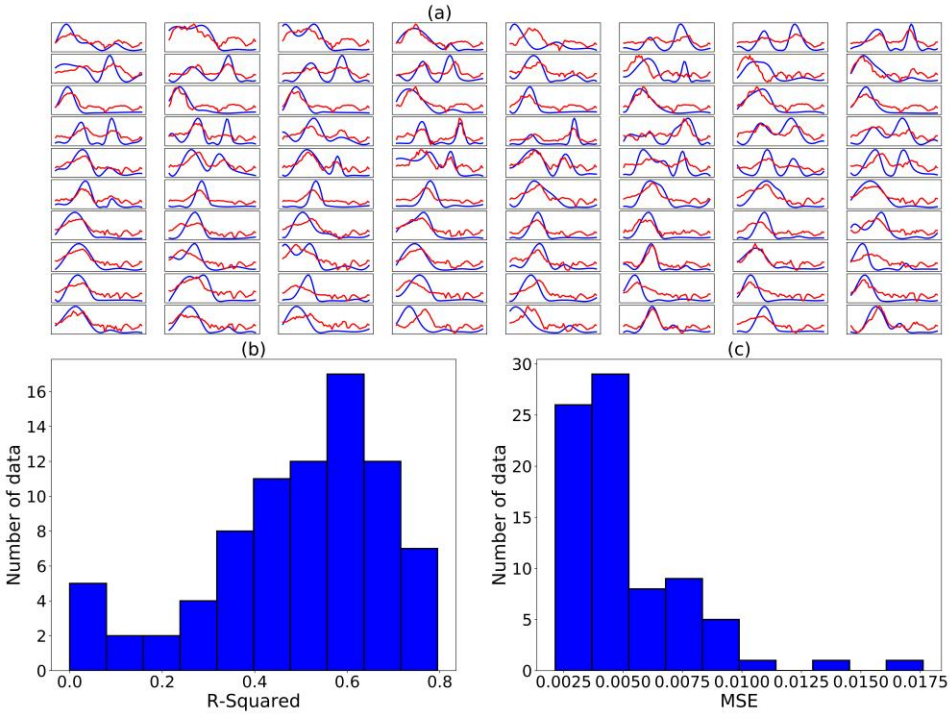
(a) OLS



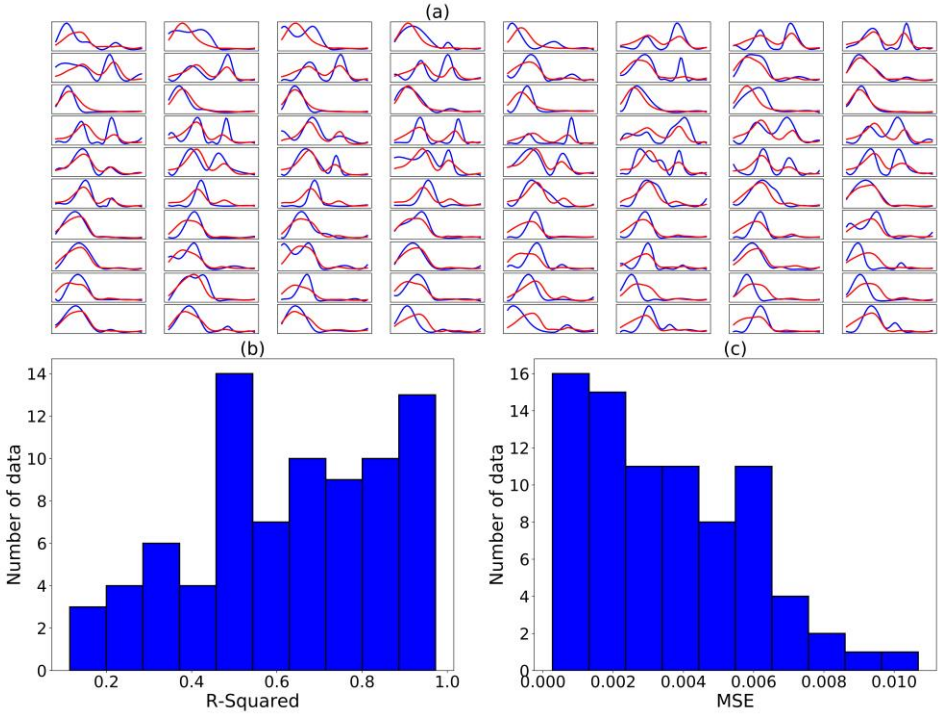
(b) LASSO



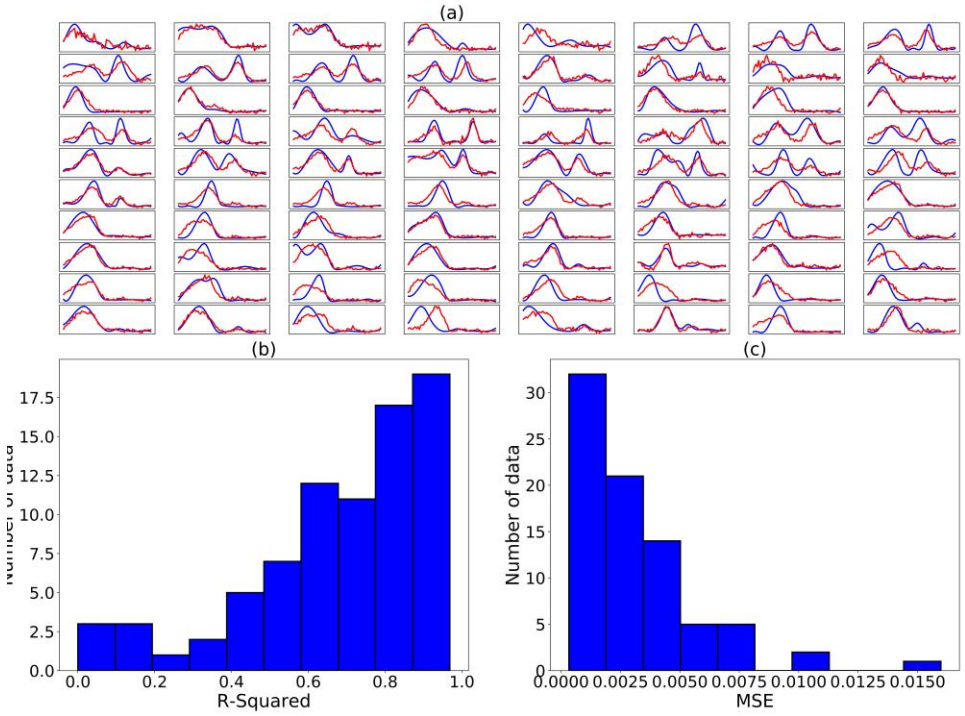
(c) ElasticNet



(d) SVR



(e) k-neighbors



(f) ANN

Figure 3-5. Comparison of synthetic NMR T2 and real NMR T2 using raw logs.

80 synthetic NMR T2 and real NMR T2 are plotted for each model using inverted logs (Figure 3-4) and raw logs (Figure 3-5) using the testing dataset as input. The OLS, LASSO, and ElasticNet have intermediate performance. In Figure 3-4 and Figure 3-5 the three models show the basic trend, but do not predict the NMR T2 with good accuracy. SVR shows bad synthetic accuracy with both inverted logs and raw logs as input.

ANN model has a good performance on inverted logs and raw logs (around 0.7), but the synthetic NMR T2 does not as smooth as real NMR T2 (Table 3-2). This is the problem for simple ANN models. For simple learning regression models of OLS, LASSO, and ElasticNet, the models are simple and thus do not synthesize the NMR T2 distribution with good accuracy; whereas for ANN models that have relatively larger estimation capacity, the model does not predict the whole NMR spectra as a whole. The ANN predict each value in the NMR T2 separately and ignore the basic spatial

dependency in the spectra. In the following section, I use four different deep learning models to synthesis NMR T2 while keeping the basic spatial features of a typical NMR T2 distribution.

Table 3-2. Smoothness of NMR T2 generated by different shallow learning models (smoothness of real-world NMR T2 is around 1e-5).

	OLS	LASSO	ElasticNet	SVR	K-neighbors	ANN
Inverted log	1.235e-06	4.690e-07	6.423e-07	0.336e-03	9.312e-07	1.639e-03
Raw log	3.515e-06	3.514e-06	1.706e-06	0.360e-03	3.310e-06	4.395e-04

Bending energy is calculated for each model to evaluate the smoothness of the synthesized NMR T2. For real-world NMR T2 data, the smoothness is around 1e-5. The lower the bending energy, the smoother. The bending energy is calculated with Eq. (3-2). The results are shown in Table 3-2. We can see that only the NMR T2 generated by ANN and SVR have high bending energy, which represents low smoothness. The smoothness of other models is small. However, the smoothness of all the shallow models is quite different from real-world NMR T2 data.

3.6 NMR T2 Synthesis using Deep Neural Networks

In this section, 4 types of deep neural networks are applied for NMR T2 synthesis. The two types of inputs, raw logs, and inverted (formation mineral and fluid composition) logs, are extracted and preprocessed. The 4 models are tested on the two types of inputs separately. The architecture and training details of the four types of deep neural networks are explained. The training results of each model are evaluated and compared.

Most deep learning models applied today is based on the artificial neural network. The definition of deep learning is changing with the development of computer hardware. Deep learning models contains artificial neural networks with more layers. Another difference between deep learning models and shallow learning models is that deep learning models uses specially designed architecture to approximate and generalize information in the training dataset more efficiently. In this section, I applied four different deep learning models for NMR T2 synthesis.

The first three models aim to train an NMR T2 decoder or a generator using unsupervised learning. VAE, VAEc and GAN models are trained in a two-step process. The VAE based model trains a decoder, and the GAN model trains a generator. The decoder and generator generalize the features of the NMR T2 distributions in the training dataset. In the second step, a simple ANN is connected with the trained decoder/generator to associate input logs with the learned NMR T2 distribution. This two-step training process increases the robustness of the model in that the trained decoder/generator generalizes the features of the NMR T2 distribution and is frozen during training in the second training step. VAEc model uses similar architecture with VAE model but uses convolutional layers.

3.6.1 VAE

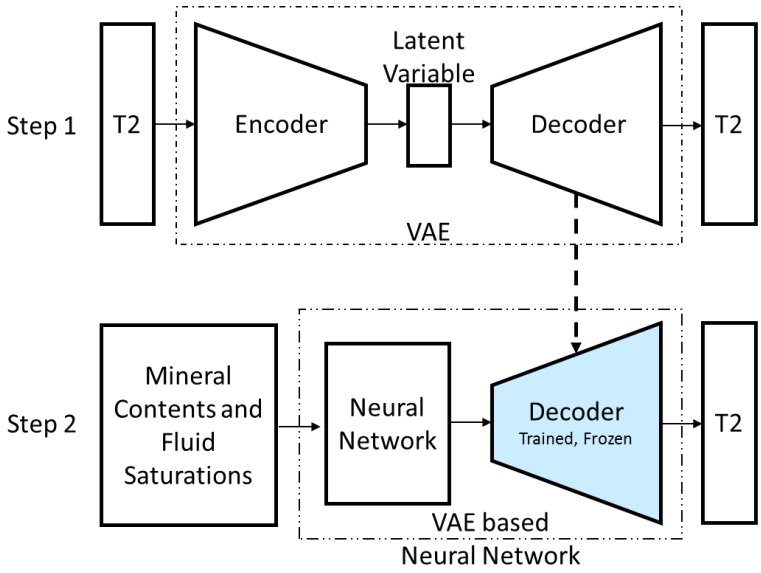


Figure 3-6. Training process schematic of VAE-NN.

To generate the NMR T2 distributions in subsurface shale formations using VAE-NN, a two-step training process is implemented prior to the testing phase (Figure 3-6). First, I train the VAE to extract and learn the dominant features of the NMR-T2 distribution data. VAE algorithm projects the generalized T2 data to the latent space, such that the latent variables can be used to reproduce the input T2 data. VAE accomplish this goal by minimizing a loss function, wherein the first component $\|\boldsymbol{X} - f(\boldsymbol{z})\|^2$ represents the difference between the input T2 ($\boldsymbol{X}$) and reproduced T2 ($f(\boldsymbol{z})$) and the second component of the loss function represents the KL divergence (Doersch, 2016), expressed in Eq. (3-8), which forces the latent variable $\boldsymbol{z}$ to follow a Gaussian distribution.

$$\boldsymbol{KL}[\mathcal{N}(\boldsymbol{\mu}(\boldsymbol{X}), \boldsymbol{\Sigma}(\boldsymbol{X}))||\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})] = \frac{\boldsymbol{1}}{\boldsymbol{2}}(\boldsymbol{tr}(\boldsymbol{\Sigma}(\boldsymbol{X})) + (\boldsymbol{\mu}(\boldsymbol{X}))^{\boldsymbol{2}}(\boldsymbol{\mu}(\boldsymbol{X})) - \boldsymbol{k} - \boldsymbol{\log\det}(\boldsymbol{\Sigma}(\boldsymbol{X}))) \quad (3\text{-}8)$$

where k is the dimensionality of the distribution and $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ represents Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$.

One advantage of VAE is that it can visualize what it has learned from the training data (Figure 3-7). The dimension of the latent layer is 2, which means the input data will be projected to a 2-dimensional latent space. When the shapes of T2 distributions are relatively simple and the training data set is limited, two dimensions are sufficient for the latent layer to generalize the features. The higher-dimensional latent layer will be required when the training dataset is extensive having a large range of distinct features. An increase in the dimensionality of the latent layer without an increase in the complexity of features in the data will increase make the VAE overfits the data, which will be counterproductive due to the generation of unrepresentative features. Figure 3-7 shows what VAE has learned from the training data. It shows what VAE's understanding of NMR T2 data. The VAE model project the NMR T2 in a latent space with dimensionality of 2. By sampling this space, the decoder can generate realistic NMR T2. We can see that most of the samples are similar to real-word NMR T2 data.

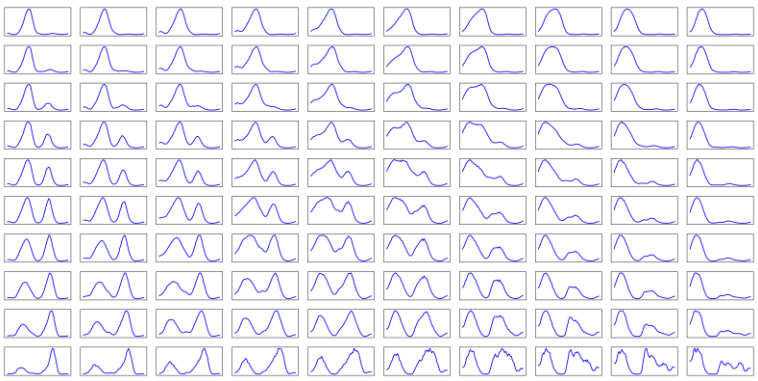


Figure 3-7. Example of latent representation of the NMR T2 distribution (What VAE learned from the training).

In the first step of training, VAE is trained to memorize and generalize the sequential and shape-related features in the T2 distribution. The sparsely distributed 64-dimensional T2 distribution is projected to a 2-dimensional latent. The VAE learns the

manifold of T2 in the latent space when minimizing the loss functions, as shown in Figure 3-7. There are infinite points in a 2-dimensional latent space on which infinite projections of T2 can be generated. The 100 randomly selected samples shown in Figure 3-7 helps to visualize the VAE's projection of T2 distribution in the latent space. In Figure 3-7, T2 distributions of various shapes are generated by sampling in the two-dimensional latent space, which leads to smoothly varying changes in the features of the T2 distribution. The VAE learning process ensures that the predictive model will have robust predictions in the presence of noise in input signals. Once the VAE is trained to memorize and generalize the dominant features in the T2 distributions, any new input of T2 distribution is projected to a corresponding location in the latent space where there already exists a projection of a similar T2 distribution, which was fed during the prior training phase.

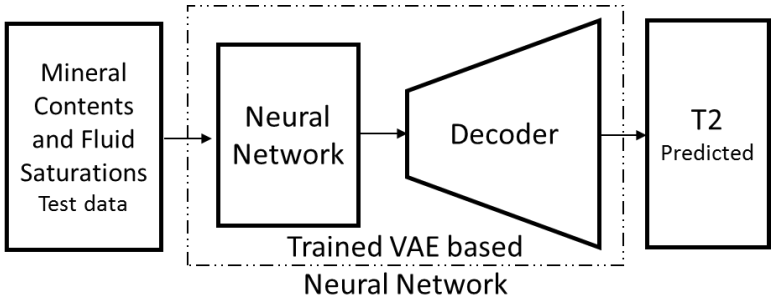


Figure 3-8. Testing process schematic.

After the VAE is trained, the trained decoder (the second half of the VAE) is frozen and connected to a neural network to associate the formation fluid saturations and mineral content with the dominant NMR T2 features extracted in the first step of the training process. The trained decoder is frozen to keep its 'memory' of the training T2 data, and only the neural network before the decoder is trained. In doing so, it is ensured that formations with similar fluid saturations and mineral contents will produce similar

NMR T2 distribution responses. After the training process is completed, the trained VAE-NN developed in the second step of the training process is used with the frozen decoder to predict the T2 distribution of subsurface formation using formation mineral volume fractions and fluid saturations, as shown in Figure 3-8 during the testing phase.

3.6.2 VAEc

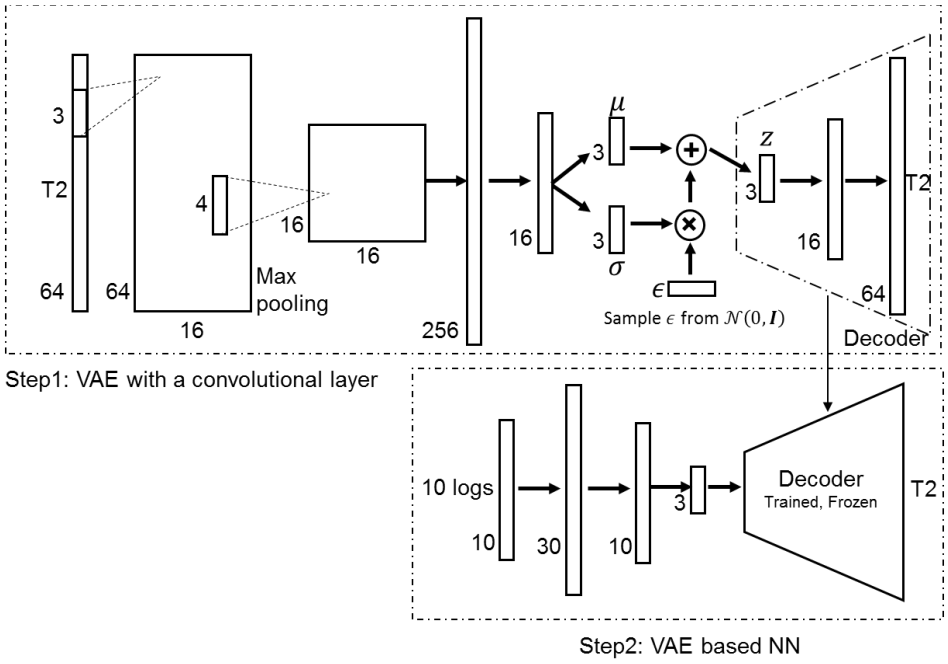


Figure 3-9. The architecture of the VAEc model during the training phase.

A convolutional layer is combined with a VAE-based neural network (NN) to better extract features of NMR T2 distribution, which is processed as a 1D vector. The architecture of the generative model for purposes of VAEc-based NN model training is presented in Figure 3-9. In the first part of the training, VAEc learns to reproduce the measured T2 distributions. The encoder part of the VAE compresses the 64-dimension T2 data to 3-dimensional latent space. Subsequently, the decoder decompresses the latent variables to reconstruct the measured T2 input data. The first step of training, which involves the encoder and decoder, comprises 8 layers: 1 convolutional layer, 1 max-

pooling layer, 1 latent layer with 3 neurons, and 5 dense fully connected layers. The first convolutional layer is 64×16×1 and is generated by filtering the 64×1 T2 distribution of a specific depth with 16 filters of size 3×1. The max-pooling layer processes the 64×16×1 output with a filter size of 4×1 to obtain a 16×16×1 output. The output of the max-pooling layer is then flattened and fed to two fully connected layers with 256 and 16 neurons each. The fifth layer of the VAEc is a 3-dimensional latent layer, which samples from the output of the previous layer. This layer is chosen to be 3 dimensional based on extensive numerical experiments. The 64-dimensional NMR T2 data is compressed to 3-dimensions through these five layers. After that, the decoder decodes the latent vector to 64-dimension through 3 fully connected layers containing 3, 16, and 64 neurons, respectively. The loss function in the VAEc model is comprised of two components: reconstruction loss and the Kullback–Leibler (KL) divergence loss. The reconstruction loss emphasizes the difference between the input and output NMR T2 data when forcing the VAEc to reproduce the input as precise as possible; whereas the KL divergence loss evaluates the difference between the distribution of the latent vector and the unit Gaussian distribution.

In the second step of training, a NN model comprising 4 layers are connected to the decoder trained in the first step. The trained decoder is frozen to preserve the generalization of the T2 distribution learned from the training data. The first three layers process input logs acquired at a specific depth in the subsurface to produce a 3-dimensional vector, which can be decoded by the trained decoder to generate the T2 distribution that best matches the T2 distribution measured at that depth. The architecture of the 4-layered NN model used in the second part of the training is shown in Figure 3-8.

3.6.3 GAN

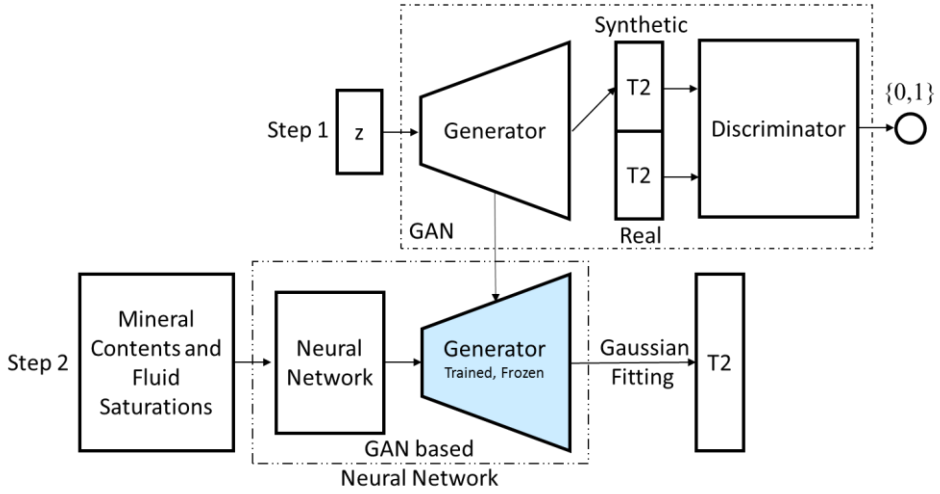


Figure 3-10. Training process schematic of GAN based neural network.

GAN includes a generative neural network G, or generator, to generate synthetic data that realistically reproduces the input T2; following that, a discriminative neural network D, or discriminator, tries to distinguish between the synthetic data from the generator and the real training dataset to maximize the objective function. The objective function $V(D,G)$ of the two players competing in the GAN is represented as (Goodfellow et al., 2014)

$$\min_{\boldsymbol{G}} \max_{\boldsymbol{D}} \boldsymbol{V(D,G)} = \mathbb{E}_{\boldsymbol{x}\sim \boldsymbol{p_{data}}}[\boldsymbol{\log D(x)}] + \mathbb{E}_{\boldsymbol{z}\sim \boldsymbol{p_z}}[\boldsymbol{\log(1 - D(G(z)))}] \quad (3\text{-}9)$$

where x represents the real training data and G(z) represents synthetic data generated by generator G.

The training process of the GAN-NN is similar to the training process of the VAE-NN involving a two-step process. In the first step of training (Figure 3-10), the GAN learns the features in T2 distributions. The generator and discriminator are trained alternatively until the generator can generate NMR T2 that cannot be identified by the discriminator as synthetic data. After the GAN is trained, the frozen generator is

connected to a neural network (step 2 in Figure 3-10) to learn to associate different mineral contents and fluid saturations with the learned features of T2 distributions. After the training process, NMR T2 distributions can be generated in the testing phase similar to Figure 3-8. T2 distributions generated by GAN-NN are not smooth. Therefore, I add a Gaussian fitting process to fit the synthetic NMR T2 to make it smoother and more realistic.

3.6.4 LSTM

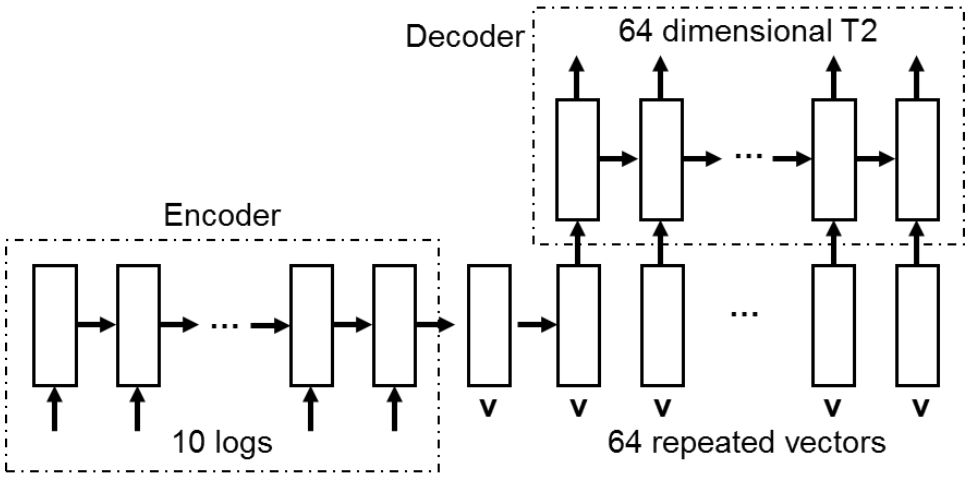


Figure 3-11. The architecture of the LSTM model during training and testing.

In the LSTM model, the input logs and NMR-T2 distribution measured in the subsurface along the length of a well are processed using LSTM as sequences. A LSTM is a recurrent neural network. A LSTM is applied for the sequence to sequence mapping, where a sequence is a list of ordered elements. A LSTM is good at dealing data with long-term dependency. This sequence to sequence learning model is successfully applied to language translation. Language translation relies on the fact that sentences in different languages are distinct representations of one common meaning. In a similar manner, different subsurface logs acquired at a specific depth are distinct responses to one common material. I implement LSTM to capture the dependency in NMR T2

distribution, which usually exhibits a smooth distribution. The smooth shape of NMR-T2 distribution illustrates the dependency between the data point in NMR T2.

The architecture of the LSTM model is demonstrated in Figure 3-11. Like VAEc model, this model comprises 1 encoder followed by 1 decoder. Both encoder and decoder are collections of LSTM layers. The encoder and decoder LSTM layers are made up of n and 64 chained LSTM modules, where n is the dimensionality of input logs. Each module is made of gates controlling data flow inside the LSTM modules. Controlled by various gates, the LSTM can choose to forget or update the information flowing through the modules. The encoder compresses the input logs into a single vector. Then, the decoder decodes the vector into target sequences. The input logs are taken as sequence and the encoder of the model takes one of the ten logs at each time step. After all the input logs are read by the encoder, the encoder generates a vector **v** of size 15 in the last step. Size 15 is found based on experience with the LSTM predictions of T2 data. The vector **v** generated by the encoder is repeated 64 times and fed to the decoder. The decoder takes the 15-dimensional vector **v** in 64 time steps and tries to reconstruct a 64-dimensional T2 distribution for a single depth corresponding to the 10 input logs relevant to that depth.

The loss function used for the LSTM model is the Mean Squared Error function. The optimizer used to train the LSTM model is Adam. During the training process, the prediction loss is evaluated using the Mean Squared Error function and Adam is used to update the weights during backpropagation.

3.6.5 Results

In this section, the running time and NMR synthesis accuracy are compared and evaluated for the 4 deep learning models. Raw logs and inverted formation compositional

logs are used separately as input for the 4 models. Similar to the shallow learning models, the R^2 and MSE values are computed for the four deep learning models.

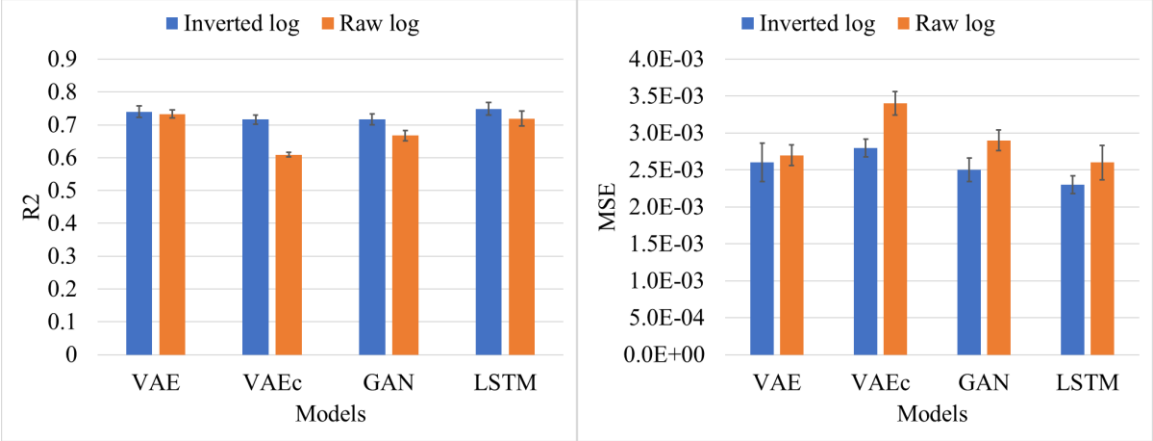


Figure 3-12. Median R2 (Left) and MSE (Right) value (with 95% confidence interval) for synthetic and real NMR T2 distribution for different deep learning models.

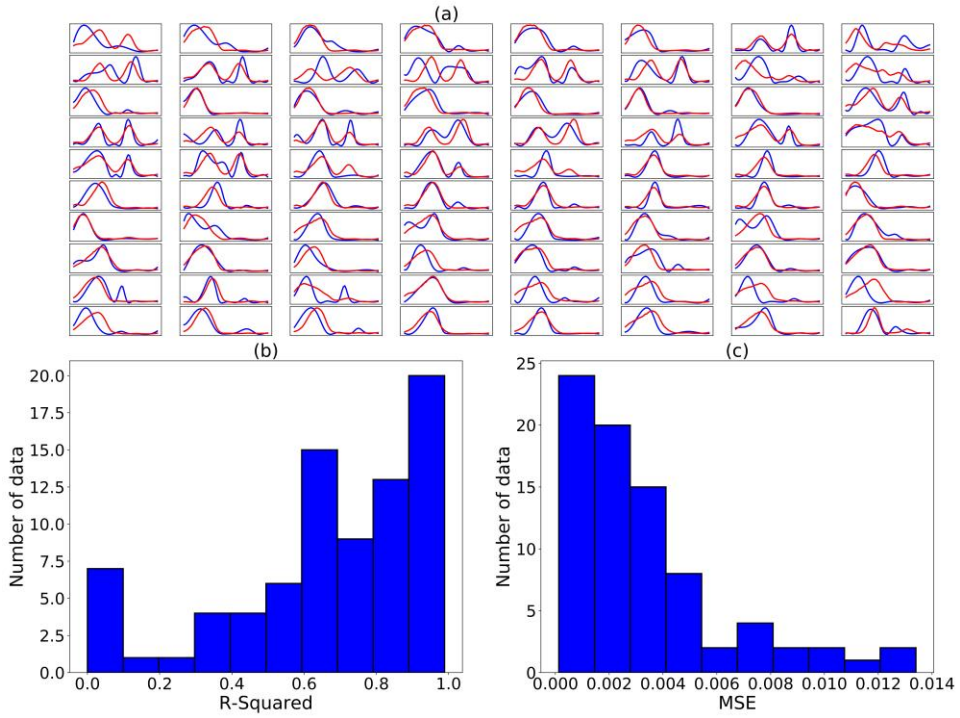
Table 3-3. Running time (s) and number of parameters of different deep learning models.

	VAE	**VAEc**	**GAN**	**LSTM**
Inverted log	47.55	47.61	22.91	569.14
Raw log	47.34	37.67	22.62	589.12
Number of Parameters	5380	6598	6114	5491

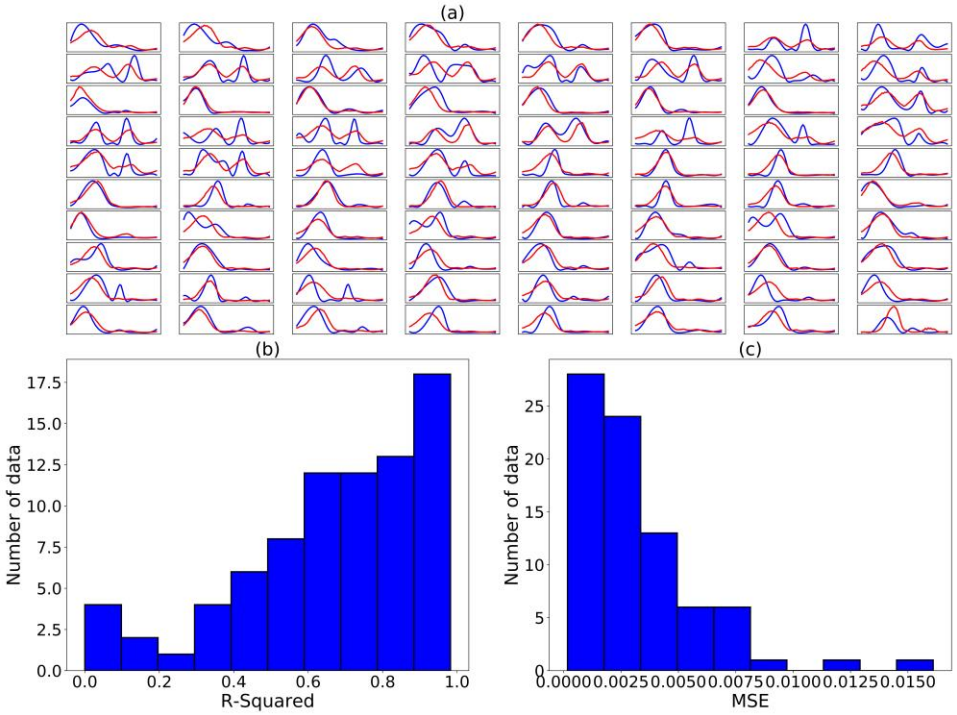
Figure 3-12 and Table 3-3 shows the R^2 and running time for each of the four deep learning models with different inputs. The four deep learning models have a good performance on the R^2 values but also has a longer running time. The four models have similar synthetic accuracy. Most of the R^2 values range from 0.70 to 0.75. Using inverted logs performs better or at least similar compared to raw logs. The inversion process that

used to generate inverted logs act like a preprocessing procedure. Machine learning algorithms usually perform better when the training data is preprocessed, and the right features are selected. Well log inversion takes multiple raw logs as input and the output of the inversion may be corrected from noise, missing data, and outliers. This may facilitate the NMR T2 synthetic since the input logs are cleaned, and more informative.

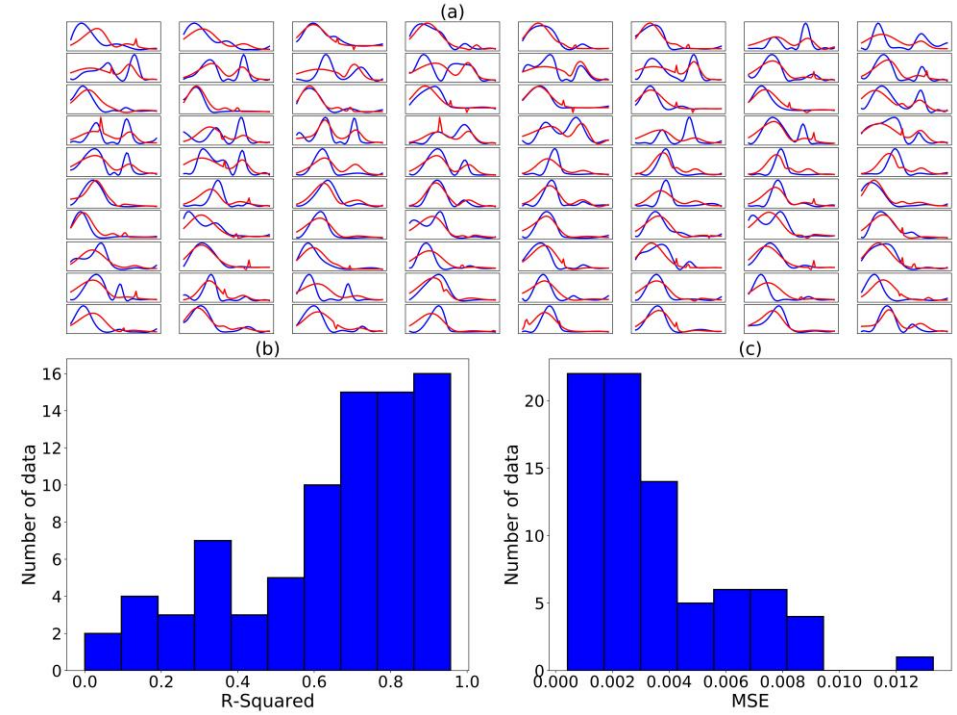
Since the first three models include a two-step training process. The total running time is the training time during the first step and the training time during the second step. LSTM model uses far more training time compared to the other three models. The following figures present the synthesized NMR T2 results compared with true NMR T2 on the testing dataset.



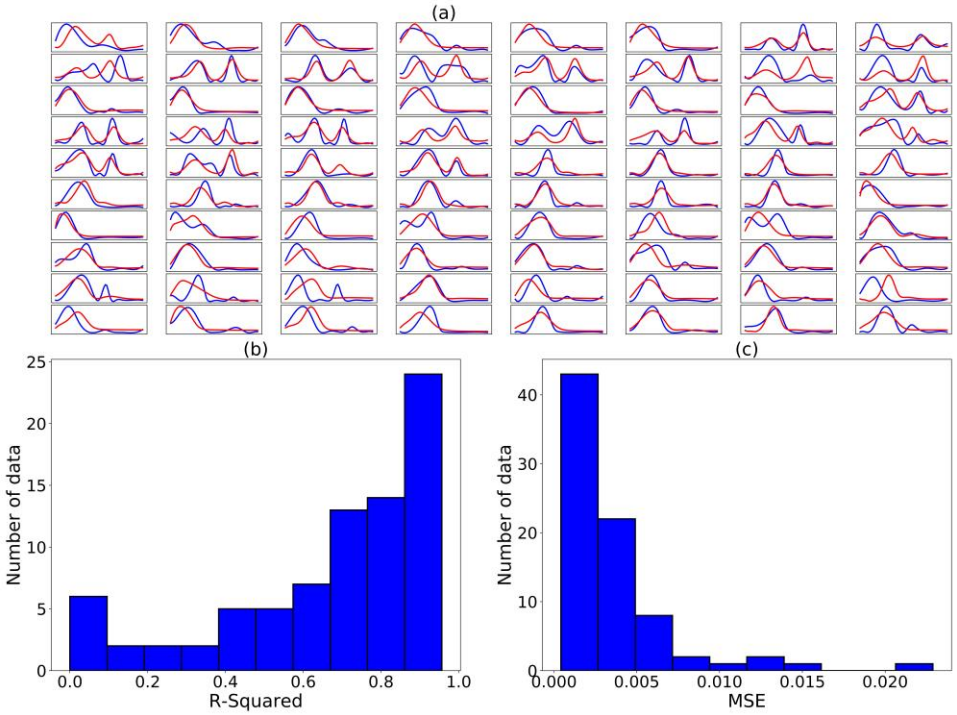
(a) VAE


(a)
(b)
(c)
Number of data
R-Squared
MSE

(b) VAEc

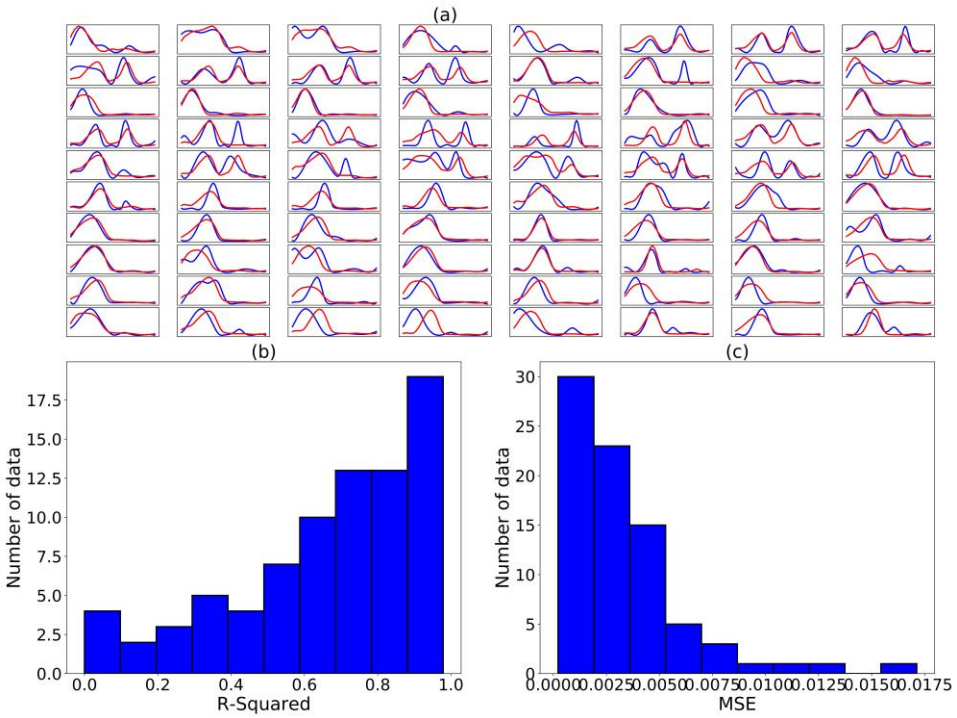

(a)
(b)
(c)
Number of data
R-Squared
MSE

(c) GAN

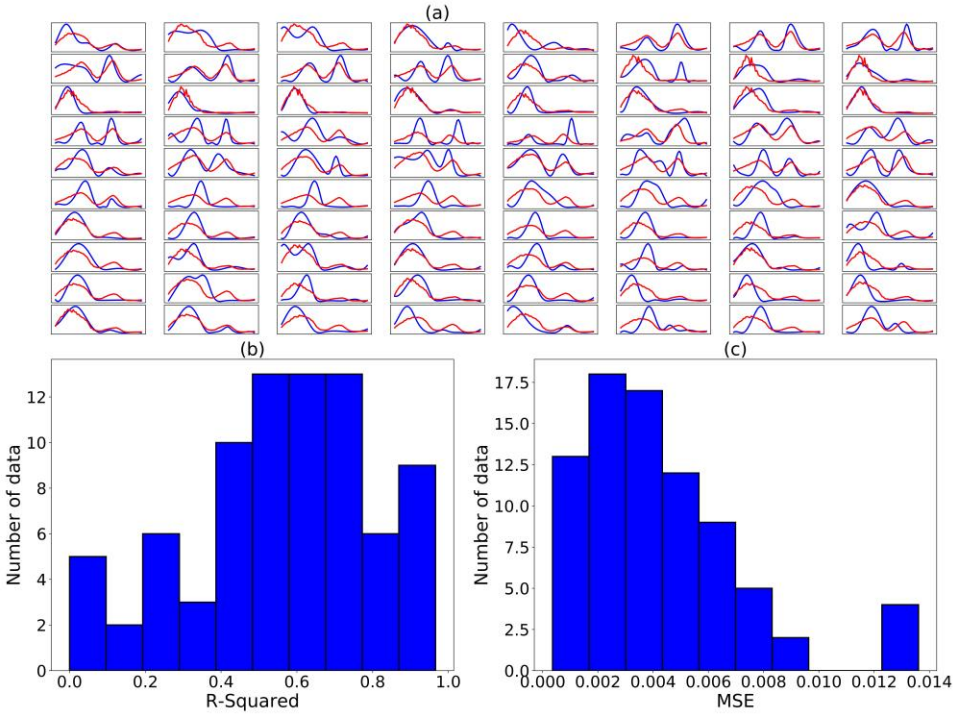


(d) LSTM

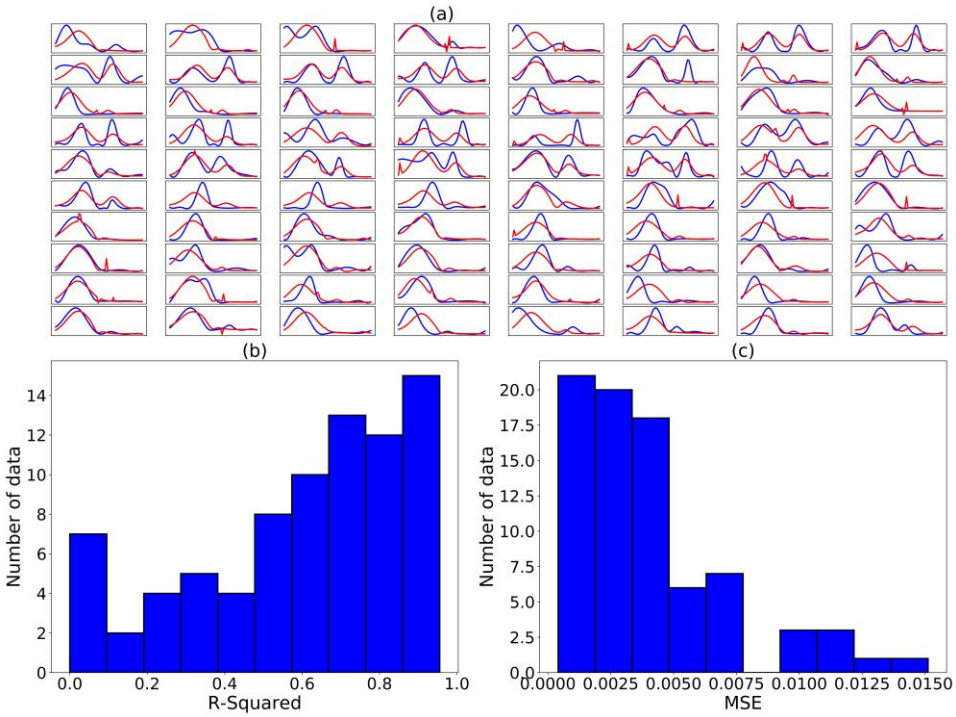
Figure 3-13. Comparison of synthetic NMR T2 and real NMR T2 using inverted logs.



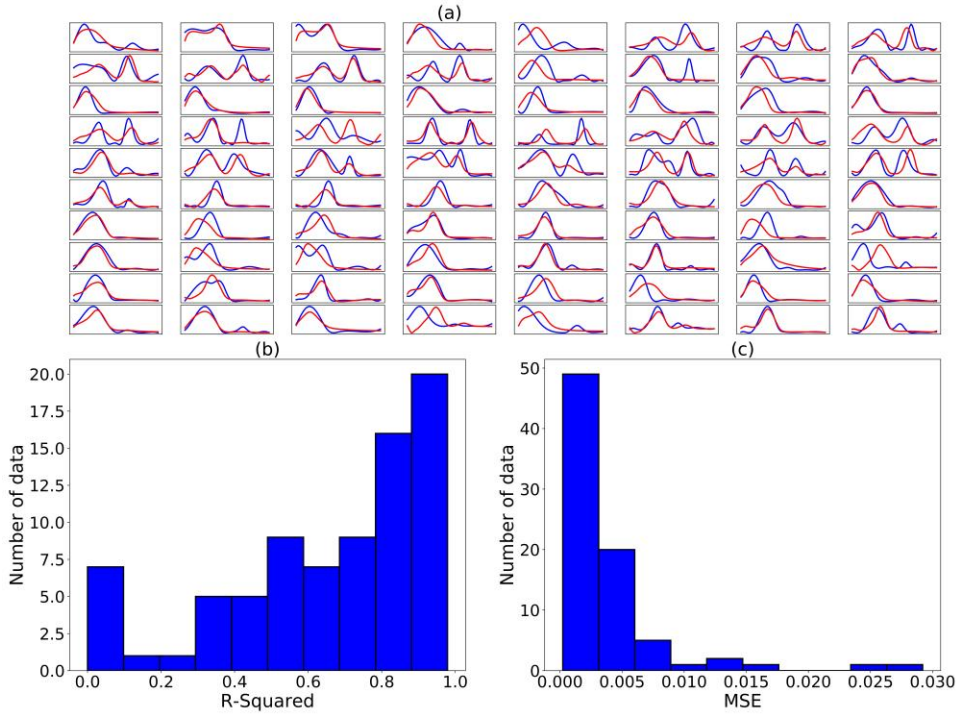
(a) VAE


(a)
(b)
(c)
Number of data
R-Squared
MSE

(b) VAEc


(a)
(b)
(c)
Number of data
R-Squared
MSE

(c) GAN



(d) LSTM

Figure 3-14. Comparison of synthetic NMR T2 and real NMR T2 using raw logs

Figure 3-13 and Figure 3-14 plot the real and synthetic NMR T2 distributions with inverted and raw logs as input for comparison. Two histogram plots showing the R^2 and MSE distribution are presented on the bottom of the figures. One obvious difference between the results of the deep learning model and shallow learning models is that deep learning models generates typical NMR T2. Shallow models, such as OLS, LASSO, are not sufficiently complex to generate accurate NMR T2 data. ANN is a more powerful model but the NMR T2 generated by ANN is not sufficiently smooth. The deep learning model I produce is similar to the shallow learning ANN model in the previous section. The VAE, GAN, VAEc model utilizes a two-step training process that produces a neural network with multiple layers. Each layer is the same as the ANN layer. The layers are trained in the same way with a forward prediction and backward error propagation. The only differences are that I applied a two-step training process to learn the patterns of

NMR T2 and I applied different architecture. Although the final model is similar to the ANN model, the synthetic results are smooth and more accurate. This difference shows the advantages of the two-step training process.

The LSTM model performs best among the 4 models and the running time is the longest. VAE model has intermediate performance. GAN and VAEc model perform the worst. Also, models perform better with inverted logs. The LSTM model performs best may come from the following reasons. First, the model architecture is relatively simple compared with other deep models, which make it easier to tune using a grid search. Other models utilize a two-step training method to learn the inherent NMR T2 patterns, which makes it harder to tune the hyperparameters. Second, the LSTM model is powerful enough to learn the inherent pattern of the NMR T2 data. The LSTM model learns the long-term dependencies of the NMR T2 relaxation time data with its recurrent modules. During the training of the GAN model, I found it is hard to tune the GAN model compared with other models. GAN model training requires the two neural networks, generator and discriminator, to have a similar capacity. One of the reasons that lead to GAN perform worse than other models is that it is hard to balance the capacity of generator and discriminator.

Table 3-4. The smoothness of NMR T2 generated by different deep learning models (smoothness of real-world NMR T2 is around 1e-5).

	VAE	**GAN**	**VAEc**	**LSTM**
Inverted log	1.748e-05	2.22e-04	2.952e-05	0.829e-05
Raw log	1.359e-05	3.006e-04	3.367e-05	0.800e-05

Another aspect to evaluate the performance of the models is whether the NMR T2 generated resembles the real-world NMR T2. For example, a simple ANN model only learns to predict the values of NMR T2 separately and ignore the inherent patterns of NMR T2 data. I applied the bending energy method to evaluate the smoothness of the synthesized NMR T2. The results for the 4 deep models are shown in Table 3-4. We can see that the smoothness of the four models is very similar, both on raw logs data and on composition logs data. The smoothness of the NMR T2 generated by deep learning models is similar to the smoothness of the real world NMR T2.

3.7 LSTM Input Sequence Order Investigation

The results presented in the last section shows that the LSTM model performs better than other deep learning models. This section investigated how the input sequence order affects the performance of the LSTM model. The LSTM model used in this section has the same architecture as the model applied in the previous section. The sequence of the 11 input logs is shuffled randomly. By shuffling the sequence of the input logs, I want to see if the input sequence has a great effect on the LSTM model's performance. If the input sequence does not have a great effect on the LSTM, the R^2 metric should have a similar value on the test set. The LSTM model is trained and tested 80 times, each time with a shuffled train set. The performance of the 80 LSTM models is shown in Figure 3-15.

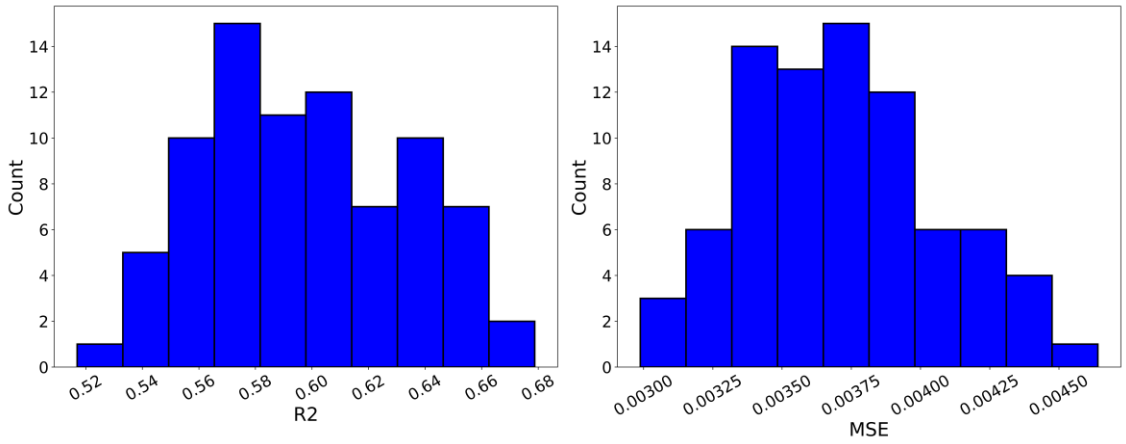


Figure 3-15. Distribution of the R2 and MSE values of 80 models trained with shuffled input sequence.

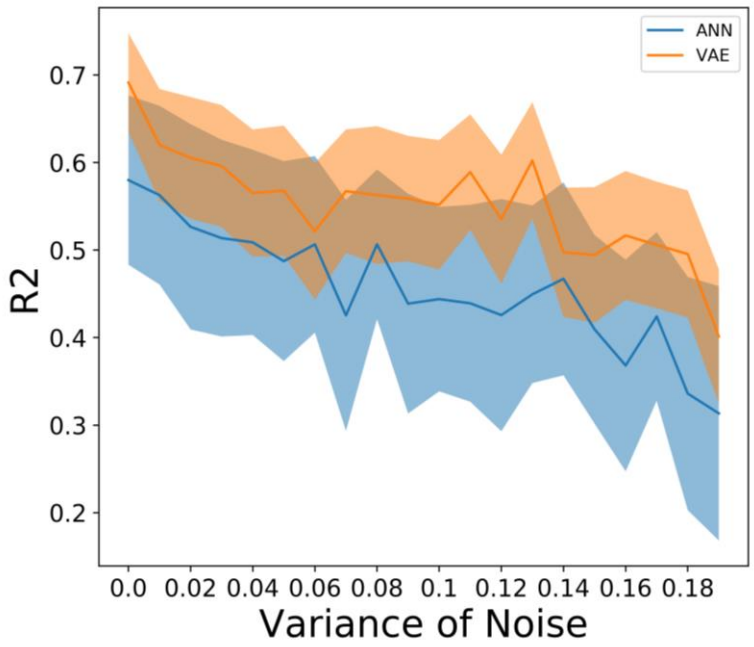
In Figure 3-15 we can see that the 80 LSTM models' performance is much worse than the LSTM model in the previous section. When the order of input logs are shuffled, the R^2 values are between 0.5-0.7 for the 80 models. In the previous section, the LSTM model architecture, learning rate, etc. are tuned by grid search on the validation set. By tuning the model, the architecture, learning rate, etc. are best suited for the given input sequence. We assume that the LSTM models trained with shuffled input logs perform worse than the model in the previous section because the model architecture etc. are not tuned. The 80 models trained in this section are built with the same architecture as the LSTM model in the previous section. The results indicate that the LSTM model is sensitive to the input sequence. The LSTM model should be tuned with the validation set to gain the best performance. This obvious drop in the R^2 shows a disadvantage of deep learning models, it is much harder and complex to tune the hyperparameter of deep learning models. It usually takes a longer time to tune deep learning models.

3.8 Models' Stability to Noise

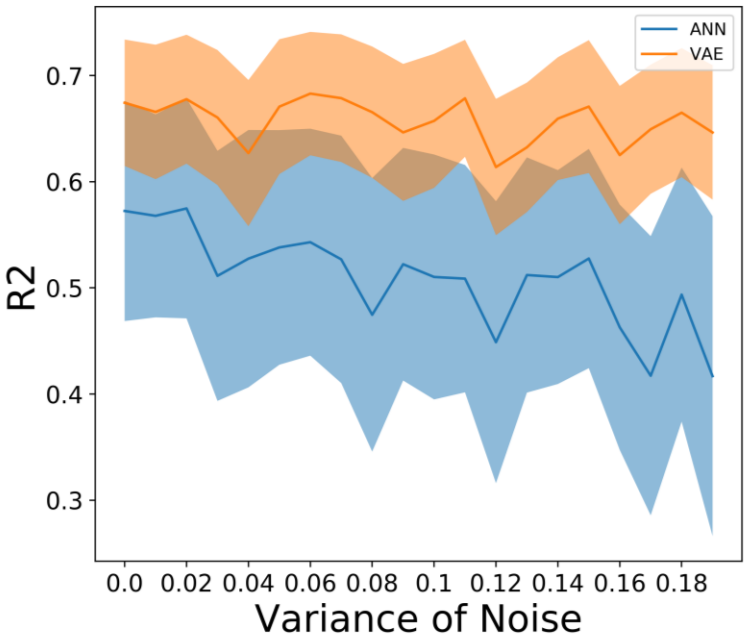
In this section, the VAE model and ANN model are trained and tested with data corrupted with noise. VAE model outperforms other deep models in terms of simplicity

and the NMR T2 smoothness. Noise is usually contained in the well logs due to the complex downhole environment. It is necessary to investigate the models' stability when the input logs are adversely affected by noise.

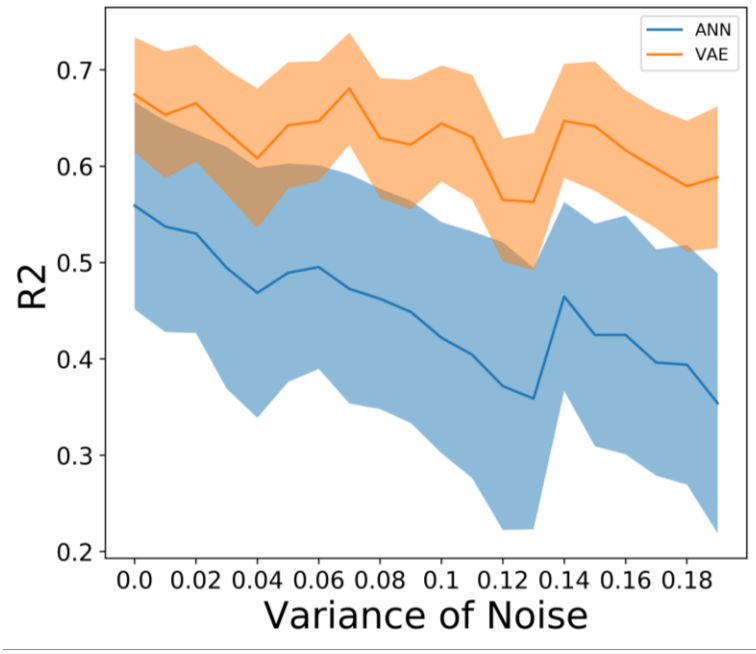
The noise is generated using Gaussian white noise. The level of the noise is controlled using the variance of the Gaussian distribution. VAE and ANN models are chosen to represent the deep and shallow learning models. The models are trained with different levels of noise added to the train or test set. Three experiments are conducted to investigate the effect of noise. First, the noise is added only to the test set. The models are trained with the clean train set and then tested on the 'noisy' test set. Second, the noise is added to the train set, but the test set is clean. Last, both the train and test set are corrupted with equal amount of noise. The models are evaluated in terms of R^2. 95% confidence interval is also calculated using bootstrap resampling method. The method used to calculate the confidence interval is introduced in section 2.5.3.2. The basic procedure includes regenerating a dataset by resampling the original dataset. The results are shown in Figure 3-16. The results show large confidence interval ranges. This is largely due to the limitation of the small dataset. The total number of the data points is too small. Resampling a small dataset will definitely cause a large variance of the results.



(a) Models trained with the clean train set, tested with the noisy test set with 95% confidence interval.



(b) Models trained with the noisy train set, tested with the clean test set with 95% confidence interval.



(c) Models trained with the noisy train set, tested with the noisy test set with 95% confidence interval.

Figure 3-16. Model performance in terms of R^2 of the three experiments.

By comparing the results of the three experiments in Figure 3-16, we can have the following conclusions. First, adding noise in the train set improves the models' stability. This is not the case when noise is added only to the test dataset, e.g. Figure 3-16 (a) shows that the models' R^2 values decrease from around 0.7 to 0.5 for both models. This is as expected since the noise changed the distribution of the test set. However, Figure 3-16 (b) and (c) show that by adding noise to the train set, the R^2 of the VAE model only decreases from around 0.75 to 0.7. The VAE model shows great stability when trained with data that are contaminated with noise.

Second, the VAE model is more stable than the ANN model. In Figure 3-16 (a), the R^2 of both ANN and VAE dropped from 0.7 to 0.5. However, in Figure 3-16 (b) and (c), the R^2 of VAE stays stable with the increase of noise. The R^2 of ANN dropped from 0.7 to 0.5. The results indicate that the VAE model has better resistivity to the noise when trained with the noisy train set. The stability of the VAE model may come from the two-

step training process. The VAE model is first trained using the NMR T2 logs to learn the distribution of NMR T2 data. This step enables the VAE model to generate a typical NMR T2 log regardless of the quality of the input log. This experiment shows the advantage of the VAE model to shallow learning models.

3.9 Novelty

- The study successfully synthesized NMR T2 distribution by processing easy-to-acquire well logs using deep neural networks. The neural networks utilize the hidden relationship between different well logs to synthesize NMR T2.
- Long short-term memory network and variational autoencoder based neural network synthesized the NMR T2 distribution at high accuracy and high smoothness. Quantification of the smoothness of the synthesized T2 distribution is needed to understand the realistic performance of the deep models.
- Deep learning models performed significantly better than shallow models.
- Utilized a two-step training process to learn the inherent features of NMR T2 and predict the entire NMR T2 distribution, comprising 64 T2 amplitudes measured at corresponding T2 times.
- A study of stability of the deep neural networks to the presence of noise indicates that addition of noise to training data makes the deep networks more stable to the presence of noise in test data.

3.10 Assumptions and Limitations of This Study

Each type of borehole-based subsurface measurements (referred here as well logs) are sensitive to specific formation properties. The compositions and contents of formation minerals and fluids have complex and hidden relationships with the various well logging

responses. Various well logs contain redundant information about formation properties. In well logging, engineers/geoscientists utilize dozens of well logs to better characterize the subsurface formation. Are all the well logs necessary? What are the minimum number of well logs needed to get all the information to represent a formation? The models applied in this study show that it is possible to generate NMR T2 distribution by processing other easy-to-acquire logs. Such a synthesis is essential for pore-scale characterization.

The idea is similar to the user portrait task in the computer science area. Big social network companies push content based on the users' interests. Different users are presented with different content. Just like it is hard to describe the formation properties, it is hard to describe users' habits. The users' habit and interest can be inferred from all the activities associated. By processing the activity history, machine learning models can predict users' characteristics. The characteristics are usually stable over time. The users can be divided into several categories and each user can be represented as an embedding vector in latent spaced. Every user is different, it is not possible to exactly measure every aspect of a user, but users' behavior is statistically similar and stable over time. A vector is enough to represent a user in the social network industry. Maybe we don't need all of the logs to represent a formation in the oil and gas industry.

In this work, I applied a similar idea. Formation measurements are statistically stable for different types of formations (e.g shale is usually associated with high GR, low porosity, etc.). Even for the shale formation in the same reservoir, the formation well logs are not the same. However, the same type of formations should have similar well log values. If we project all the formations into latent space, formations with similar

properties should be projected into similar places. The well logs derived from formations with similar properties should be similar. Machine learning models can learn the subtle differences of formation from existing well logs and utilize the hidden relationship that engineers cannot find. This work is based on the following assumptions:

- Information redundancy exists in well logs. It is not necessary to utilize dozens of well logs to describe a formation.
- NMR T2 logs can be generated from processing other formation logs by utilizing the information redundancy.
- The well logging responses for formations with similar properties are similar.
- Machine learning models can learn the hidden relationships between different well logs for generating other logs.
- Inverted well logs are more suitable for the NMR T2 synthesis task compared to raw logs.

3.11 Recommendations for Future Work

In this study, the models are trained and tested with a small dataset. Preprocessing and dimensionality reduction are performed to ensure the data quality, validation set, and training monitor is used for hyperparameter tuning and prevent overfitting. However, the following steps can be used to improve this work:

- Apply the model on a larger dataset.
- Perform cross well validation to generate a more robust conclusion about log synthesis.
- Investigate the stability of the models to the noise in the well logs.

- Log generation from embeddings. Just like a user in a social network can be presented as a vector, the formation can also be presented as a latent vector. Formation properties can be estimated from formation embeddings instead of well logs.

3.12 Conclusions

In this chapter, shallow and deep machine learning models are applied for the purpose of NMR T2 log synthesis from other logs. Two types of input logs are tested. The shallow and deep models are tuned by grid search on the validation dataset. For deep models, training monitors (early stopping callbacks) are applied to prevent overfitting. The models' performance is evaluated on the testing dataset in terms of R^2 values, mean square error (MSE), and smoothness. The following conclusions can be drawn from this study:

Deep learning models can be used to generate the NMR T2 distribution responses of the near-wellbore region by processing easy-to-acquire well logs under data constraints.

Bending energy from classical beam theory is used to evaluate the smoothness of NMR T2 generated. Smoothness is the inherent characteristic of NMR T2. Deep models show a good ability to generate NMR T2 with good smoothness. The simple ANN model does not generate NMR T2 with good smoothness.

Models with inverted formation composition logs as input perform better than models with raw logs as input. This is because raw logs contain a lot of highly correlated logs. Inverted formation composition logs are acquired by processing raw logs. Inverted logs do not exhibit high correlations between each other.

For the shallow models, K-neighbors regressor performs best. OLS, LASSO, and ElasticNet models have similar performance. NMR T2 generated by the ANN model is not as smooth as real NMR T2. The smoothness of real NMR T2 is around 1e-5 in terms of bending energy, whereas the NMR T2 generated by ANN is around 1e-3. SVR model has the worst performance in terms of both accuracy and smoothness.

For the deep models, all the four models generate NMR T2 as smooth as real NMR T2 (around 1e-5 in terms of bending energy). On average, the deep model performs better than shallow models on both inverted and raw logs. The R2 of best-performing shallow models is around 0.7. The best performing deep models achieve R2 of 0.75. LSTM and VAE models perform better than VAEc and GAN models. LSTM model takes a longer time (570s) to train, but it performs best among the four models (R2 of 0.75).

The performance of the models is greatly affected by the quality of the data. The models perform better on certain depth because there are not enough data samples that are similar for the models to learn from.

Deep learning models require more data to train, and more time for hyperparameter tuning. GAN does not perform as good as other models because it is hard to tune the generator and discriminator.

Shallow models use much less computational resources compared to the deep models. The NMR T2 generated by shallow models are not as good as deep models in terms of both accuracy and smoothness. Best-performing deep models can generate NMR T2 with R2 of 0.75 and smoothness of (1e-5), which outperforms all shallow models. But the k-neighbors model can generate NMR T2 data with R2 of 0.748 with inverted logs as input.

Chapter 4 Classification of Multipoint Compressional-Wave Travel Times for the Characterization of Mechanical Discontinuity

4.1 Background – Characterization of Fractures, Cracks, and Discontinuities

4.1.1 Physical Phenomena of Discontinuity

4.1.1.1 Crack vs. Fracture vs. Discontinuity

The terms fracture and crack can be used interchangeably in most cases. Both refer to the mechanical discontinuity inside materials. Discontinuities, on the other hand, is a more general concept. A discontinuity can be a fracture, crack, bedding, joint, etc. A discontinuity refers to a plan that can be described by physical or chemical characteristics change. Predicting and monitoring the geometry and condition of fractures is critical in areas like rock mechanics, geotechnical projects, structural health monitoring, geothermal reservoir development, and hydraulic fracturing. In the oil and gas industry, the characterization of geometry and direction of induced fracture systems are critical to the hydrocarbon production rate. Monitoring, description, and prediction of the state and behavior of fractures is an important research topic.

Fracture types are defined in terms of the relative movement of the fracture planes. Fractures are created when the stress inside the material exceeds the strength of the material, causing the material to lose cohesion among its weakest plane. Fractures can be formed due to compression, tension, or shear stress. When tensile stress is acted normal to the plane of the fracture, the fracture planes move away from each other, which is referred as Mode I (opening) crack. Mode II (in-plane) crack is the sliding mode, where shear stresses acting parallel to the plane of the crack and perpendicular to the

crack front. Mode III crack is called tearing mode and is formed when shear stress acts parallel to the plane of the crack and parallel to the crack front.

Fracture generation and propagation characteristics are governed by multiple physical and mechanical properties of the material. Brittleness is an important characteristic that affects the fracture development rate. Brittle material breaks without significant plastic deformation. Ductile materials have a small region of elastic behavior and a large region of ductile behavior before they fracture. Different factors govern the brittleness. At low temperature, materials are brittle due to the constrained molecular motion. High confining pressure hinders the generation of fractures and thus make materials ductile. Some minerals are brittle due to the characteristics of the chemical bonds. Water weakens the chemical bonds, making rocks more ductile.

Fracture forms in stages. First, micro-fractures exist in the material. These micro-fractures are in the open or closed state under current stress status. When stress inside rock changes, fracture initiates at the tip of a minute defect. This is the initiation stage of rock fractures. Second, when fracture propagates, it may encounter other existing fractures. Based on the orientation, stress status, etc. the fracture may cross the existing fracture or dilate the existing fracture. This is the propagation stage of the fracture. When the fracture system inside the rock is well developed and the rock is not strong enough to withstand the stress anymore, the material breaks. Different fracture stages exhibit distinct acoustic signatures due to three types of fracture modes (Aggelis, 2011). The tensile mode includes opposing movement of fracture planes and shear-type of fracture includes slides of fracture sides. Studies have successfully applied a clustering method to

cluster the acoustic emission data with different signatures into different groups (Aggelis, 2011; Aggelis et al., 2009).

4.1.1.2 Disciplines That Require Characterization of Discontinuities

The characterization of discontinuities is required in many disciplines. In the oil and gas industry, various kinds of discontinuities are characterized on different scales for the purpose of efficient oil and gas production. The geometry of beddings, joints, faults in the reservoir is important for reservoir characterization and modeling. Large scale seismic measurements are utilized to acquire the information about the discontinuities on reservoir scale. For the petrophysical characterization of oil and gas-bearing formations, the existence of fractures may improve the oil and gas flow efficiency. Coring and various kinds of laboratory testing techniques are developed for core scale characterization. Imaging well logging tools are also used to measure the beddings and fractures near the wellbore. For an unconventional reservoir, e.g. shale reservoir, hydro-fracking is a widely used method to create artificial fractures in the reservoir to improve oil recovery. In the civil engineering area, fracture characterization is important for structure health monitoring. Sonic and ultra-sonic waves are utilized to measure the discontinuities in steel, concrete, and other materials used in civil engineering.

Laboratory experiments and numerical simulations are two important ways to understand the processes of fracture imitation and propagation. Among all the experimental methods, acoustic emission (AE) is a popular method for monitoring the condition of materials and the state of embedded cracks/fractures. AE is generated due to the evolution of fractures. The evolution of fractures often relates to unevenly distributed stress. The change of the unevenly distributed stress may be caused by temperature,

confining pressure, etc. Fractures form or propagate when stress is higher than material strength. The stress inside the material is redistributed during fracture propagation and part of the energy is radiated as acoustic events. The sensors of the AE device detect the signal and the position of the acoustic event can be calculated. The AE equipment calculates the 3D position of the acoustic event by placing multiple sensors around the fractured material. The arrival time of the same event is different for different sensors. Based on the difference of the arrival time, and the relative position of the sensors, the 3D position of the sonic event can be calculated.

4.1.2 Existing Crack Characterization Techniques in Laboratory

4.1.2.1 Static vs. Dynamic

The crack characterization techniques used in the laboratory condition can be roughly classified into static and dynamic methods. The static method characterizes materials without changing the interior structure of the material. For example, CT scanning can acquire the fractures and structures of material without breaking the material.

The dynamic characterization method observes the signals emitted from the material during a dynamic change process. For example, the acoustic emission method records the acoustic emission signals in tri-axial experiments. During the experiment, external forces are exerted on the material to break the material. Acoustic signals are generated during fracture propagation. The dynamic characterization method has been successfully used in the characterization of hydro-fracture geometry in hydro-fracking experiments in laboratory conditions.

4.1.2.2 Popular Crack Characterization Techniques

Different laboratory techniques and physics-based numerical modeling techniques have been applied to predict, monitoring and crack evolution processes for structure health condition monitoring, remaining useful life prediction, crack path prediction, etc. The following sections summarized popular fracture characterization techniques applied in laboratory conditions. The laboratory techniques summarized in this section include dynamic measurement techniques: Acoustic Emission (AE) and other acoustic related measurement techniques, true-triaxial laboratory experiment, laboratory hydraulic fracturing; and static techniques: 3D X-ray CT scanning, micro-CT.

AE is applied in many studies to identify the location of the fracturing event inside the material. AE is one of the most popular fracture characterization techniques in laboratory conditions. The application of the AE measurements is usually simple and finally feasible. AE signals have been used to characterize the seismic scattering of the subsurface fractures (Zhu et al., 2015), identifying the cracking mode (Aggelis, 2011), identifying bedding planes of rocks (Wang et al., 2018), monitoring the fracture conditions during triaxial and hydraulic fracturing experiments (He et al., 2010; Yang et al., 2012).

Zhu et al. (2015) used the scattered acoustic wave to characterize the seismic scattering of the subsurface fractures (Zhu et al., 2015). The authors applied models with different fracture types. Acoustic emission signals before and after the application of the fracture systems are measured and compared. The author applied three types of fracture models: single fracture, dual fracture, and fracture zone models. The results show that reflection from the fracture tip is proportional to the aperture of the fracture. This study

indicates that different fracture systems will result in acoustic signals with different characteristics. The scattered acoustic wave is a good way to characterize the subsurface fracture.

Different fracturing stages exhibit distinct acoustic signatures. Tensile cracks are developed at the initial stage of loading, whereas shear cracks dominate the later (Aggelis, 2011). Aggelis et al. studied the characteristics of concrete models with different health conditions. The authors tried to use clustering algorithms to cluster the acoustic signals from the concrete models into different groups. 25 concrete mixtures were produced. The acoustic signals are measured for each of the concrete models when the models are cracked. The results show that before and after crack, the acoustic parameters show a different pattern. Based on a few AE parameters, the stage of the damage for the concrete material can be estimated. By utilizing a clustering algorithm, the authors successfully clustered acoustic signals into two groups. Each group contains AE signals generated from materials with different structure health. This study shows that the AE signals can at least given the average health condition of the material. Whether or not AE can generate the 3D geometry of the fracture system is still unknown.

True-triaxial, uniaxial, and laboratory fracturing are experiment types commonly used to investigate the fracture evolvement under laboratory condition. He et al. used AE to investigate the rock failure process during the true-triaxial experiment (He et al., 2010). The authors identified three stages based on the analysis of AE signals. The first stage includes rearrangements of the rocks, closing of cracks, friction along the pressure plate. In the second stage, the stress is redistributed, and the boundary condition is changed. In the third stage, a large number of cracks are generated, and the rock burst

appears. A similar study is performed by Yang et al. (Yang et al., 2012). Laboratory fracking is another popular way to investigate the rock behavior under both triaxial stress and the hydraulic pressure in the wellbore. Zhou et al. (Zhou et al., 2008) used laboratory hydraulic fracturing to investigate fracture propagation behavior and fracture geometry. The results indicate the hydraulic fracture dilate existing fracture when the horizontal differential stress is low.

Another popular laboratory method to investigate the geometry of fractures inside the rock is CT scanning. CT scanning utilizes X-ray source to scanning materials. The internal structure of a material can be reconstructed by the absorption of the X-ray. Watanabe et al. (Watanabe et al., 2011) applied CT to build a 3D numerical model of rock fractures. Single-phase flow simulation was performed with those models. Based on the 3D numerical model of the fracture system, the authors calculated the distribution of porosity, permeability, and the flow rate of the numerical core models. The study helps to understand the real geometry of fractures and the fracture flow phenomenon. A similar study is performed by Cai et al. Cai et al. (Cai et al., 2014) reconstructed the 3D fracture network and investigated the permeability change of coal samples during cyclic loading. The results indicate that coal permeability is directly related to fracture connectivity, geometry, density, and stress conditions. The application of CT scanning requires complex and expensive scanning equipment.

Zhou et al. (Zhou and Yang, 2017) investigated the effect of the layer orientation of fracture propagation in shale formation using CT scanning. AE, microseismic, and CT are used to measure the fracture system of shale samples. Uniaxial compression experiments are applied to shale with different bedding. The results show that shale has

obvious brittle characteristics. During the uniaxial experiment, there is no obvious plastic deformation stage observed. Wang et al. (Wang et al., 2018) also investigated how bedding planes of shale affect mechanical properties. Both AE and CT are applied in this study. Different shale samples with different bedding types are tested using a uniaxial compression test. Four typical stages of AE behavior based on accumulated release energy are found. Bedding inclination has a clear effect on the morphology of fracture based on the CT images.

Through the literature survey, AE is the most popular method to investigate fracture geometry under laboratory conditions. AE identifies the crack event location by inversion. However, the AE signal is not informative enough. It can be used to characterize the general fracture condition inside rocks, but it may not able to reconstruct the 3D geometry of the fracture system. CT is a way to reconstruct the fracture system inside rocks. It is more expensive than AE, but it provides exact 3D fracture geometry. CT cannot acquire real-time signals like AE. CT and AE measurements are often used to measure the fracture geometry of rocks in different geomechanical experiments like laboratory fracking or triaxial experiment.

4.1.3 Machine Learning Workflows for Crack/Fracture Characterization

Different machine learning algorithms have been applied to solve the rock fracture characterization problems using the AE signal. Artificial Neural Network (ANN) is one of the most popular machine learning algorithms. Other supervised learning algorithms like Random Forest, Support Vector Machine (SVM), Convolutional Neural Network (CNN), are also used. Some research aims to find patterns in AE data using

unsupervised learning algorithms. K-means clustering algorithm is a popular unsupervised algorithm.

ANN is a simple and flexible supervised learning algorithm. Many researchers used ANN to generalize the relationships between different parameters of fractured rocks. Moore et al. (Moore et al., 2018) applied ANN to predict fracture growth from simulation data. The authors gathered data from the Finite-Discrete Element model. The study aims to predict whether two fractures will coalesce or not based on the parameters of fracture orientations, distances between two fractures, the minimum distance from one of the fractures to the nearest boundary, etc. This is a simple supervised learning problem. The authors predicted the results with an accuracy of 0.68. The study from Zhou et al. (Zhou et al., 2018) applied ANN and SVM to classify rock fracture and blast events. AE signal is gathered from the rock fracturing experiment. The data is preprocessed the detect the event in the data and extract features. The results show that ANN has the best classification accuracy. A similar study conducted by Liu et al. (Liu et al., 2015) used ANN to predict rock types from AE measurements. Four types of rocks were selected to conduct uniaxial compression. The authors assume that different rocks have different failure modes and different failure modes generate different types of AE signals. The ANN is proved to be a good method to recognize AE signals from different types of rocks.

Another popular method used is SVM. SVM is a popular supervised learning algorithm. SVM aims to maximize the margin between different classes. Farhidzadeh et al. (Farhidzadeh et al., 2014) used SVM to classify the AE signal acquired from experiments. They extracted different features like aptitude, duration, etc. from the AE

signal, and used the extracted features to predict whether the fracture is a tensile fracture or shear fracture. The study is conducted based on data acquired from two experiments. I think the results of the study are promising, but more data samples are needed to validate. A similar study is performed by Jingrong et al. (Jingrong et al., 2010). The authors applied wavelet packet analysis to extract features from AE data. The authors aim to identify two types of fractures using SVM classification.

Other researchers applied CNN to process the measurements during fracture propagation. Miller et al. (Miller et al., 2017) applied CNN to process 2D images acquired from rock fracturing simulation. The 2D fracture image is processed as a weighted graph. CNN is used to learn the graph features that are most predictive of final fracture length distribution. Researchers from Harvard and MIT applied CNN to process seismic data (Perol et al., 2018). They developed a CNN called ConvNetQuake, which is trained on a large dataset of labeled raw seismic waveforms. The ConvNetQuake learns a compact representation that can discriminate seismic noise from earthquake signals. The Net takes seismic as input and outputs a probabilistic location of an earthquakes' source from a single state. This study can be used as an example in the process AE signal to predict the fracture location.

Acoustic emission (AE) is one of the most widely used methods in laboratory conditions to monitor the health condition of geomaterial, composite material, and etc. Loutas et al. (Loutas et al., 2017) applied a data-driven method to predict the fatigue life of composite material based on AE data under laboratory conditions. AE is widely used because it is easy to apply, and its cost is low. The authors applied a probabilistic machine leering method, which is a Non-Homogeneous Hidden Semi Markov Model

(NHHSMM), to predict the remaining useful life of the material under fatigue loading in real-time. AE signal is preprocessed using wavelet transform and principal component analysis (PCA) methods to extract damage sensitive health indicators. The neural network is used to generalize the trend in the training data and predict the remaining life of the material. The proposed frame successfully predicted the remaining useful life of the material under laboratory conditions.

Gui et al. (Gui et al., 2017) applied a support vector machine (SVM) with optimization techniques to monitor a structural health condition. The authors applied features extraction and data mining methods to convert existing features into a lower-dimensional space. The reduced features will help reduce the redundancy in the high-dimensional space. Eight parameters are extracted to describe the characteristics of the AE signals. After features are extracted, SVM is applied with a 5-fold cross-validation method. The results show that SVM has high accuracy to distinguish between undamaged and damaged materials. Cross-validation is the key to improve the performance of the model. Wang et al. used machine learning algorithms to predict the crack growth rate. Different parameters describing the stress filed like stress ratio, intensity factor, etc. are used as input to predict the crack growth rate. Three machine learning algorithms are applied and compared. Data are acquired from laboratory experiments.

Apart from the studies processing laboratory data, other researches aim to use simulated data to monitor and predict the fracture evolution process. Pierson et al. (2018) applied a data-driven correlation analysis between observed fatigue crack path and simulated crack (Pierson et al., 2018). The authors aim to investigate the similarity between the simulated crack geometry and the real crack geometry in the specimen under

cyclic loading. The results are critical to validate the numerical models used in predicting the geometry of geomaterial. Multi-scale finite-element simulation of cyclic loading was performed using a high-fidelity model representing a real geomaterial. The correlation between the results from simulation and real experiment is calculated. The results show that the micromechanical fields of the uncracked microstructure might provide some degree of predictiveness for the small fatigue-crack path. This study provides an example of investigating the geometry of cracks using simulated data.

Other studies use simulated data to predict the crack path related parameters in geomaterial. Rovinelli et al. (2018) applied machine learning to identify small crack driving force in a special material. A Bayes Network is used to identify relevant micromechanical and microstructural variables that influence the direction and rate of crack propagation (Rovinelli et al., 2018). 11 possible crack driving forces are utilized within the model to describe the current state of the material. The goal is to predict the crack propagation direction and associated crack propagation rate. This study shows the advantages of using simulated data. Experiment data does not contain detailed force distribution inside the material. Simulated data are clean, precise, and well organized.

Numerical simulation is commonly used in evaluating the fracture propagation process. This is especially true when machine learning models are involved to predict the characteristic of the fractures. Laboratory experiments usually generate limited data. AE experiments generate a large size of data, but the quality of the data is low. Applying machine learning models requires a large amount of clean or labeled data as the training set. The results of the models rely more on the quality of the data than the machine learning model itself.

4.2 Motivations for this Work

Characterizing discontinuities (e.g. fractures, cracks) in materials in the laboratory conditions is a challenging task. Many factors affect the geometry of the fracture system. Material mechanical properties, stress distribution, etc. may affect the generation and propagation of the fracture system. Three commonly used fracture system characterization techniques are acoustic emission, ultrasonic imaging, and CT scanning. AE method reconstructs the fracture system by detecting the acoustic events during fracture propagation. AE is a relatively cheap and simple technique, but it requires the change of the internal structure of the materials. CT scanning method is based on the X-ray absorption phenomenon. Different kind of material has a different absorption rate to X-ray. Based on the absorption rate of the material, the CT scanning equipment can reconstruct the internal structure of a material. Compared to other methods, CT scanning requires a radioactive source and detection sensors, which makes the equipment more complex and expensive. CT scanning is a relatively time consuming and high-cost rock characterization technique. Ultrasonic imaging is another way for fracture system characterization under laboratory conditions. It also requires high-cost equipment for accurate rock fracture characterization.

To overcome the disadvantages of AE and CT scanning, we proposed to investigate the feasibility of utilizing multi-sensor acoustic measurements to predict the internal fracture system characteristics of materials. The proposed method is a static noninvasive characteristic method, which does not break the internal structure of materials. The design of the experiments is based on real-word experiments to ensure it is simple and cheap enough to implement. We explored the feasibility of characterizing

fracture systems by training machine learning methods to classify various types of fracture systems based on the traveltimes of diffusion/wavefront. For this study, we perform the following three broad tasks: (1) numerical modeling of crack bearing material models with different statistical parameters, (2) Fast Marching Method simulation to simulate the sonic wave propagation inside the numerical crack-bearing models, and (3) train machine learning methods to characterize the crack-bearing materials based on the simulations of sonic front propagation through crack-bearing material.

4.2.1 Description of Workflow

This study aims to characterize discontinuities in crack-bearing material by sonic wave travel-time measurement and classification models. Numerical 2D material models with embedded fractures are implemented with different fracture systems. Sonic wavefront propagation is simulated on the 2D models. The front wave travel times are measured with pre-embedded sensors. Machine learning classification models are trained and tested on the travel time dataset. We assume that if the classification models can classify the 2D models with high accuracy based on the travel time data, we can also implement the same method on real-world experiments.

The experiments configurations of the sonic source and receiver for the measurement of sonic travel times are inspired by real-world laboratory experiments (Bhoumick, 2018). The 2D simulation models and the position of the sonic sensors are similar to the real-word experiment. The equipment and design of the experiments are much simple than CT scanning experiments. Moreover, combining machine learning models and acoustic propagation measurements enables the fast characterization of

material in a noninvasive manner. Sonic wave propagation process is simulated using the Fast-Marching Method (FMM), which enables fast simulation of the sonic propagation process. Different 2D materials models with different fracture systems are built with a stochastic modeling method of fractures. The simulation experiments simulate the sonic wave propagation process originated from an embedded sonic source, and the sonic wavefront travel time is measured by placing 28 signal receivers. Fracture systems with different statistical parameters are embedded in the materials models to create different groups of models. The training and testing dataset are created by simulating the sonic wave propagation processes on the crack-bearing material. Different machine learning models are applied to explore the possibility of using sonic wavefront travel time to characterize the statistical parameters of the crack-bearing systems.

4.2.2 Key Fundamental Questions to Be Answered

This chapter aims to answer the following questions.

- Can sonic wave travel time be used for crack-bearing material characterization with machine learning models?
- Is fast marching method accurate and applicable for simulation of sonic wave propagation in crack-bearing material?
- What is a good method to build crack-bearing material models? What parameters control the distribution and characteristics of fracture systems?
- How do the characteristics of the fracture system affect the classification ability of various machine learning models?
- Based on the experimental configurations, what is the importance of each sensor?

4.3 Fast Marching Method Introduction and Validations

4.3.1 FMM Validation Method Introduction

Eikonal equation characterizes the evolution of a closed surface through a material with a specified velocity function. The fast-marching method (FMM) is developed to solve the Eikonal equation, expressed as

$$|\boldsymbol{\nabla u}(\boldsymbol{x})| = \mathbf{1}/\boldsymbol{f}(\boldsymbol{x}) \text{ for } \boldsymbol{x} \in \boldsymbol{\Omega} \quad (4\text{-}1)$$

$$\boldsymbol{u}(\boldsymbol{x}) = \mathbf{0} \text{ for } \boldsymbol{x} \in \boldsymbol{\partial\Omega} \quad (4\text{-}2)$$

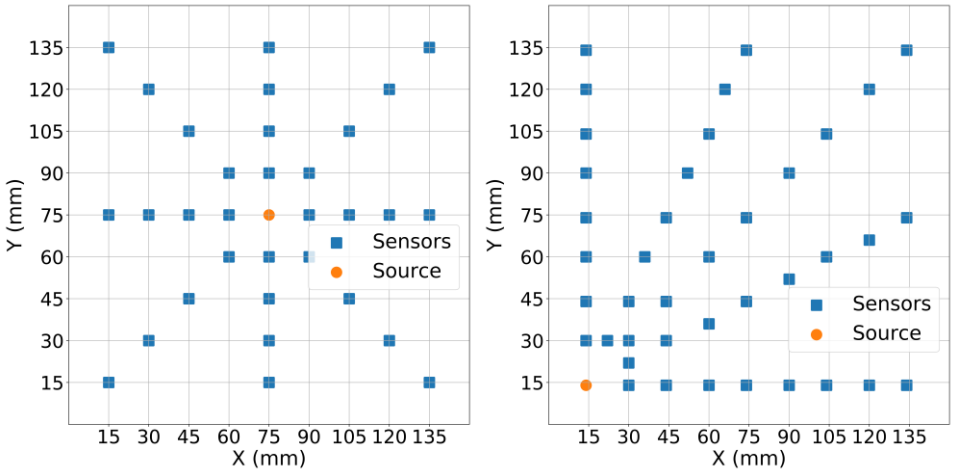
where $u(x)$ represents the travel time of the front (first arrival time), $f(x)$ represents the velocity function for the material, Ω is the open set with well-behaved boundary, $\partial\Omega$ is the boundary, and x is the coordinate. Eikonal equation characterizes the evolution of a closed surface through a material with a specified velocity function. The FMM algorithm is similar to Dijkstra's algorithm. Both algorithms monitor a collection of nodes and expand the collection of nodes by including a new node with the lest value of the travel time. FMM algorithm has been successfully applied in calculate the wavefront evolution in strong heterogeneous layered geomaterial (Rawlinson and Sambridge, 2004). In our study, FMM is used to simulate the sonic wavefront propagation process in 2D numerical models of crack-bearing material (Furtney, 2015). The FMM algorithm is validated using the k-Wave MATLAB toolbox for sonic wave simulation (Treeby and Cox, 2010) when applicable. The analytical solution for the sonic wave propagation is also calculated.

The FMM algorithm is a popular and efficient algorithm used to solve the boundary propagation problem. To validate the FMM algorithm is reliable in solving the sonic wave propagation in crack-bearing materials, I compared the FMM simulation results with analytical results using different types of crack-bearing materials. The FMM algorithm is validated with 6 different experiment scenarios. Digital 2D crack-bearing

materials are built with different grid numbers, different properties, different fractures numbers, etc. The digital crack-bearing materials are simulated with both the FMM algorithm and the k-Wave toolbox. When applicable, an analytical solution is also calculated and compared with the FMM algorithm simulation results. The 6 different experiments are listed in the following sections.

4.3.2 FMM Validation with The Different Grid Number

The number of grids affects the accuracy of the FMM simulation. With a higher number of girds, the simulation of the front travel time will be more accurate, but the running time will be longer. To select the best number of grid size, I used two experiments to compare the front travel time from different simulation methods. The digital 2D models are built as Figure 4-1.

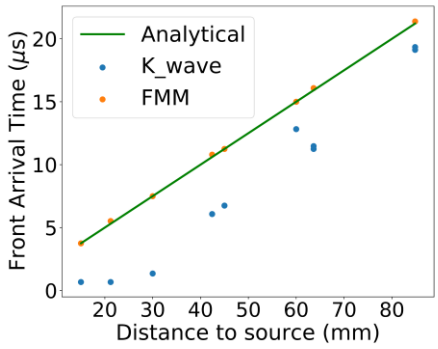


(a) Case #1. (b) Case #2.

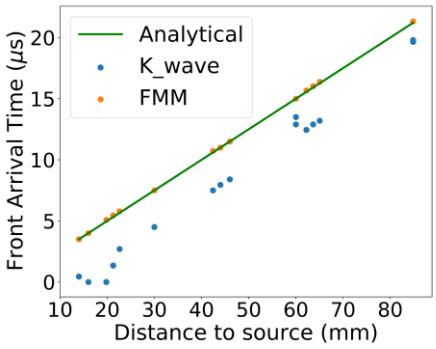
Figure 4-1. Two types of sensor locations for FMM validation.

Two digital 2D models with a size of 150mm by 150mm are built with different sensor locations. During the simulation process, the FMM simulator will simulate the sonic wavefront propagation process. The sonic wavefront is originated from the source point marked in Figure 4-1 at time 0. The simulation process finishes when the wavefront

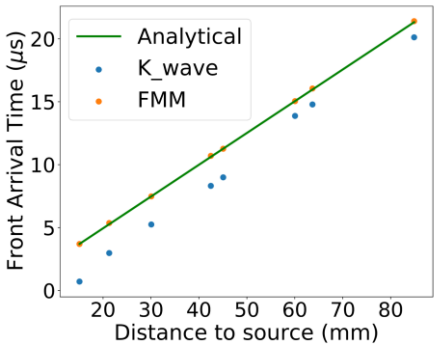
is propagated out of the model. The first experiment places the sonic source in the center of the model. 32 sonic sensors are placed symmetrically around the source. The second experiment places the sensor at the lower-left corner of the materials sample, 40 sensors are placed on different angles with respect to the x-axis. 4 different grid size is selected to simulate the sonic wave propagation process: 50 by 50, 75 by 65, 100 by 100, 500 by 500. Both experiments are simulated with different grid sizes. The simulation results from FMM, k-Wave, and analytical solution methods are shown in Figure 4-2 and Figure 4-3.



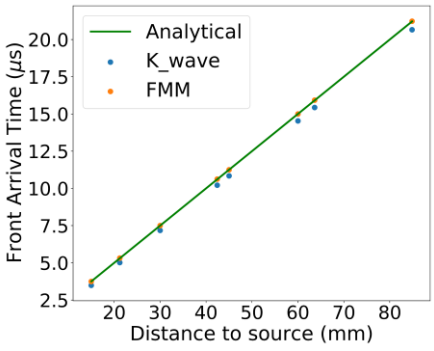
(a) The number of grids: 50 by 50.



(b) The number of grids: 75 by 75.

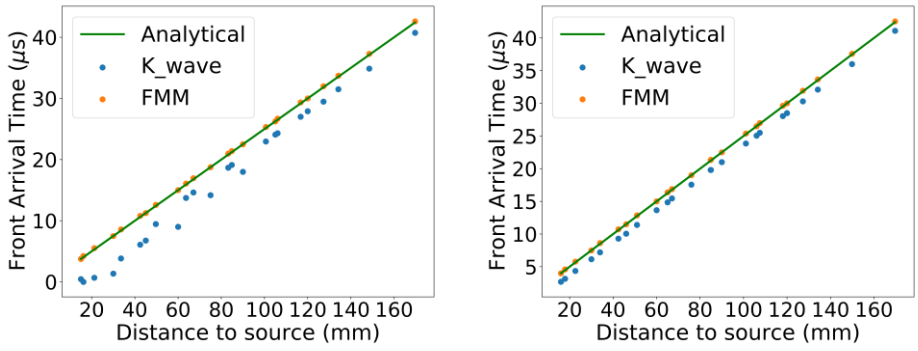


(c) The number of grids: 100 by 100.



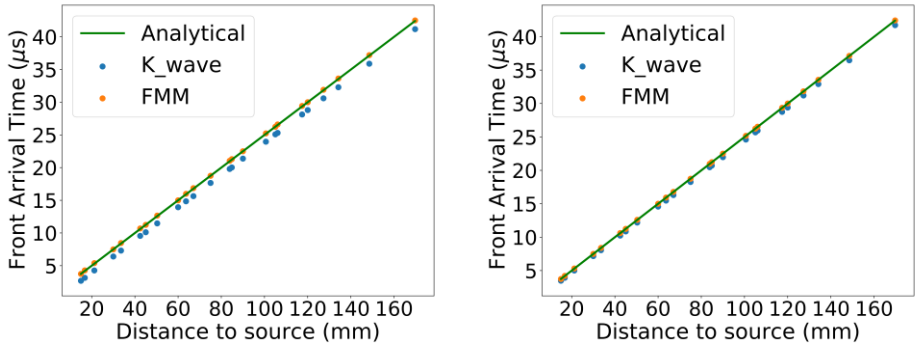
(d) The number of grids: 500 by 500.

Figure 4-2. Comparison of sonic wavefront arrival time measured by each sensor calculated by FMM, k-Wave, and an analytical method for case #1. Four different number of grids selected. X-axis represents the distance between the source and the sensor; y-axis represents the wavefront arrival time.



(a) The number of grids: 50 by 50.

(b) The number of grids: 75 by 75.



(c) The number of grids: 100 by 100.

(d) The number of grids: 500 by 500.

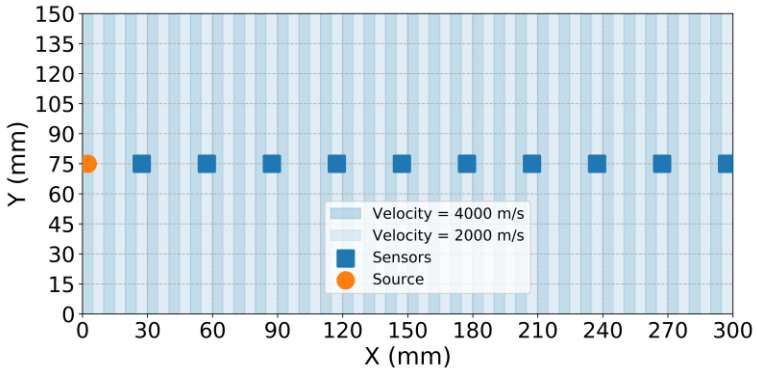
Figure 4-3. Comparison of sonic wavefront arrival time measured by each sensor calculated by FMM, k-Wave, and an analytical method for case #2. Four different number of grids selected. The X-axis represents the distance between the source and the sensor; the y-axis represents the wavefront arrival time.

From Figure 4-2 and Figure 4-3, we can see that FMM always generated reliable simulation results, but the simulation results from the k-Wave simulator are greatly affected by the number of grids. The k-Wave simulation deviates from the analytical solution when the gird number is small. FMM algorithm has high accuracy even with a small number of grids. This is because FMM and k-Wave use different methods to solve the wave propagation process. FMM only monitor the wavefront, the calculation process is fast and efficient. K-Wave simulation simulates the whole process of wave propagation. Complex differential equations are solved in order to get the whole sonic

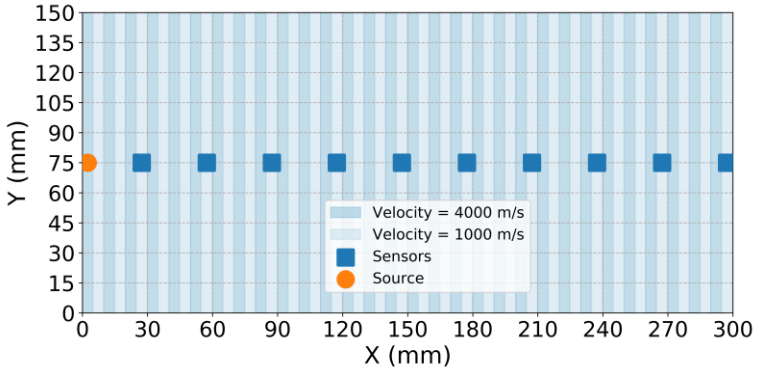
waveform. From this experiment, we can see that when the grid size is 500 by 500, both k-Wave and FMM solutions coincide with the analytical solution. In the following section, the grid number of 500 by 500 will be used when similar experiment setups are applied.

4.3.3 FMM Validation with Alternating Material Properties

In this section, a model with alternating properties is produced. The model is shown in Figure 4-4. This experiment is to test the performance of simulators on material with different layers. The materials have a dimension of 150mm by 300mm. The sonic signal source is located at the center of the left boundary. 10 sensors are placed equally on the horizontal center of the model. Two cases are studied with different wave propagation speed. For case #1, the sonic wave velocity for the matrix is alternating between 4000m/s and 2000m/s. For case #2, the velocities are alternating between 4000m/s and 1000m/s. 60 layers are embedded in the models.



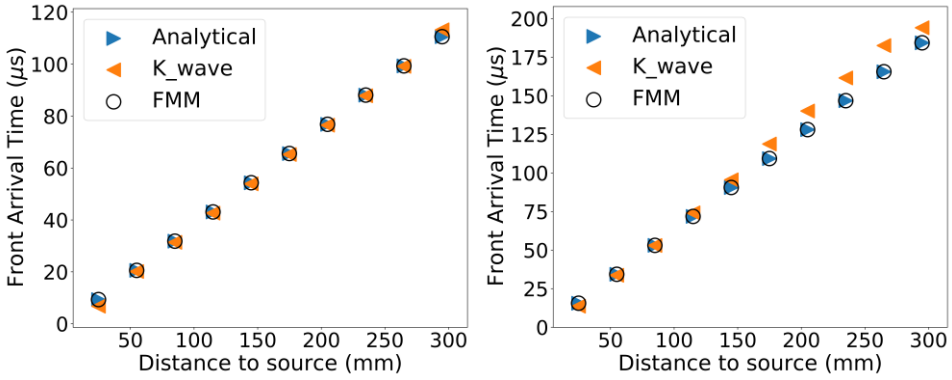
(a) Case #1: Velocities alternating between 4000m/s and 2000m/s.



(b) Case #2: Velocities alternating between 4000m/s and 1000m/s.

Figure 4-4. Sensors location for FMM validation with alternating material properties. The change of material properties is reflected in the change of the wave propagation speed.

The simulation results from FMM and k-Wave are shown in Figure 4-5. For the first case, the wavefront arrival time calculated from both FMM simulation and k-Wave simulation is close to the analytical solution. For the second case, the FMM simulation is still accurate, but the k-Wave simulation deviates from the analytical solution after the wave travels further. What we observe during the k-Wave simulation is that the wavefront reflects back and forth between two interfaces of the layers and gradually lost energy when it propagates. FMM simulation only cares about the wavefront. It will not be affected by the strength of the sonic wave.

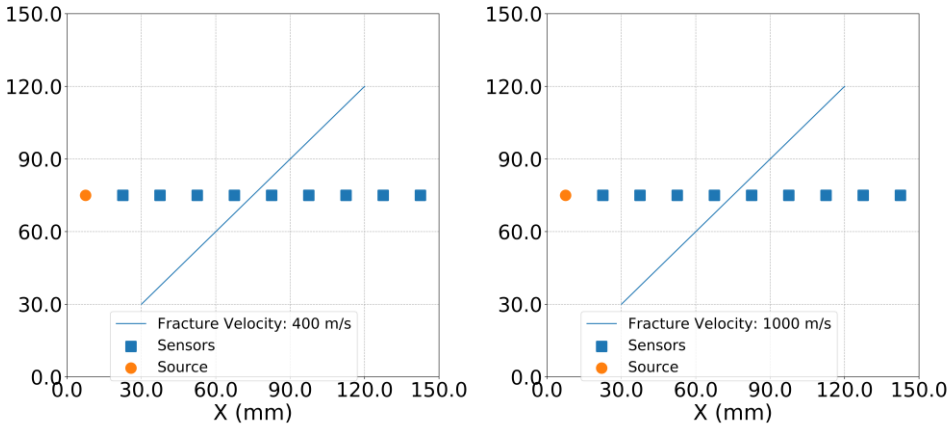


(a) Case #1. (b) Case #2.

Figure 4-5. Comparison of sonic wavefront arrival time measured by each sensor calculated by FMM, k-Wave, and an analytical method for two cases with alternating material properties.

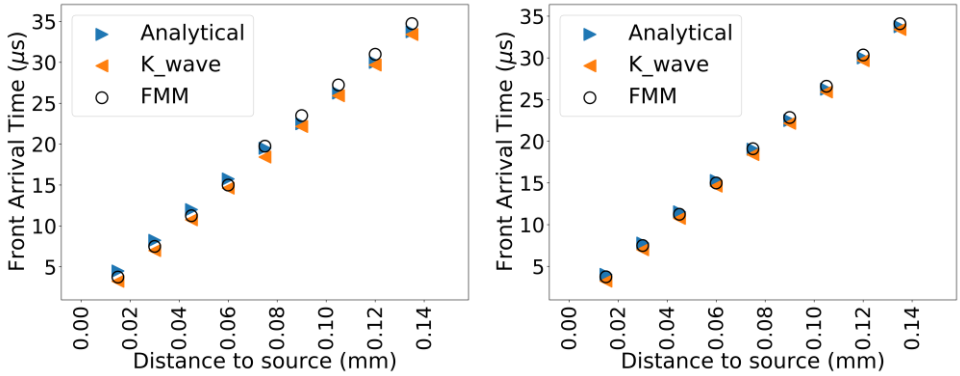
4.3.4 FMM Validation with A Single Fracture

Since the goal of this study is to characterize fracture systems based on sonic wave propagation, it is important to validate the FMM algorithm with materials with fractures. In this section, I implemented a single fracture in the 2D model. The experiment setups are shown in Figure 4-6. The sonic wave propagation speed in the matrix is 4000m/s, and the speed in the fractures is 400m/s and 1000m/s for the two cases respectively. The simulation results from FMM and k-Wave are shown in Figure 4-7. We can see that both FMM and k-Wave generate similar results to the analytical solution.



(a) Case #1. Sonic speed in fracture: 400m/s. (b) Case #2. Sonic speed in fracture: 1000m/s.

Figure 4-6. Sensors location for FMM validation with a single fracture. The sonic speed for the fractures is 400m/s and 1000m/s for cases #1 and #2.



(a) Case #1. (b) Case #2.

Figure 4-7. Comparison of sonic wavefront arrival time measured by each sensor calculated by FMM, k-Wave, and an analytical method for two cases with different fracture properties.

4.3.5 FMM Validation with Fractures

In this section, two models with vertical parallel fracture systems are created. The crack-bearing models are shown in Figure 4-8. The crack-bearing material has a dimension of 150mm by 300mm. 300 vertical fractures are embedded into the crack-bearing materials. A similar source the sensor configuration is applied in this experiment. Two cases are studied with this source-receiver configuration. In the first case, the sonic

wave propagation speed is set to 45m/s; in the second case, the speed is set to 450m/s. This experiment is to investigate whether the FMM simulation can generate accurate results in crack-bearing materials with high heterogeneity.

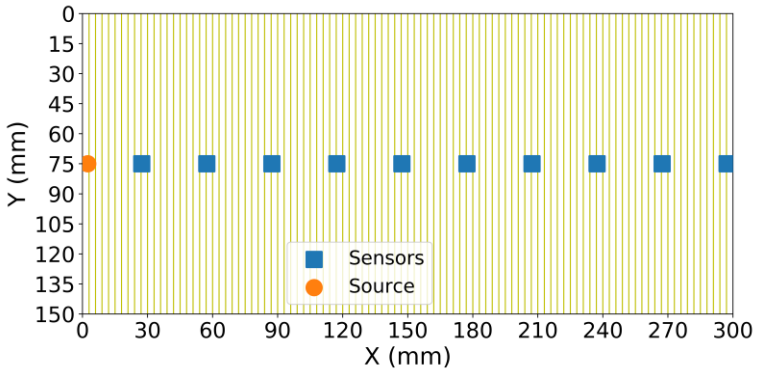
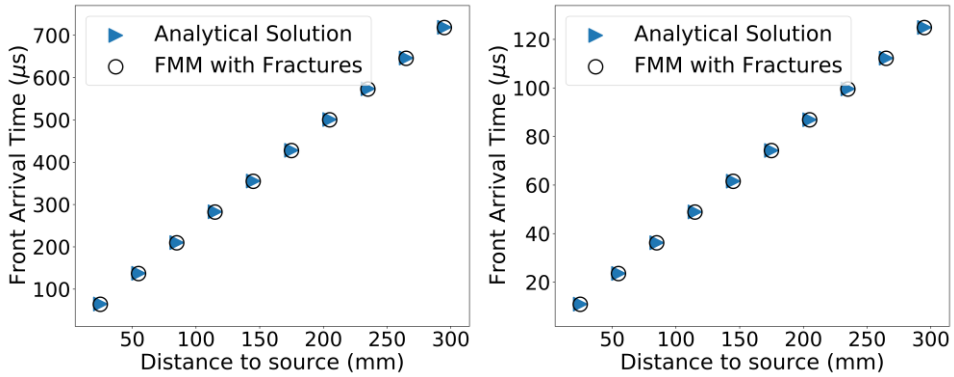


Figure 4-8. Sensors location for FMM validation with a single fracture. The sonic speed for the fractures is 45m/s and 450m/s for the two cases.



(a) Case #1. Sonic speed in fracture: 45m/s. (b) Case #2. Sonic speed in fracture: 450m/s.

Figure 4-9. Comparison of sonic wavefront arrival time measured by each sensor calculated by FMM, and an analytical method for two cases with different fracture properties. Case #1 the sonic wave propagation speed is set to 45m/s in the fracture; Case #2 the speed is set to 450m/s in the fracture.

Figure 4-9 shows the wavefront travel time calculated by FMM and analytical solution. The x-axis is the index of the 10 sensors, the y-axis is the wavefront arrival time measured by the sensors. The results show that the FMM simulation is not affected by the high contrast of speed between the fracture and matrix. However, the k-Wave simulation

is affected greatly due to the existence of a large number of fractures. The simulation process is so slow that it may take weeks to finish the simulation.

4.3.6 FMM Validation with Varying Velocity Distribution

Another common scenario is that the properties of the matrix are different everywhere in the material. In this section, I compared the FMM simulation results and the analytical solution on a model with a velocity distribution. By applying the sonic wave simulation on the materials, I assume that the property change of the material will in the sonic wave propagation speed. The simulation process is basically to calculate the wavefront position at different times given the wave propagation speed on the material. In previous sections, I validated with materials that are divided into different areas. The velocity inside each area will be the same. In this section, a velocity distribution is defined by a function expressed as:

$$\boldsymbol{f} = \frac{\mathbf{1}}{((-\mathbf{2}\boldsymbol{x})^{\mathbf{2}}+(-\mathbf{2}\boldsymbol{y})^{\mathbf{2}})^{\mathbf{0.5}}} \tag{4-3}$$

The corresponding analytical solution for arrival time is:

$$\boldsymbol{u} = \boldsymbol{x}^{\mathbf{2}} + \boldsymbol{y}^{\mathbf{2}} \tag{4-4}$$

The source-receiver configuration, velocity distribution of the material, and the analytical solution of the Eikonal equation with the velocity distribution are shown in Figure 4-10 (a-c).

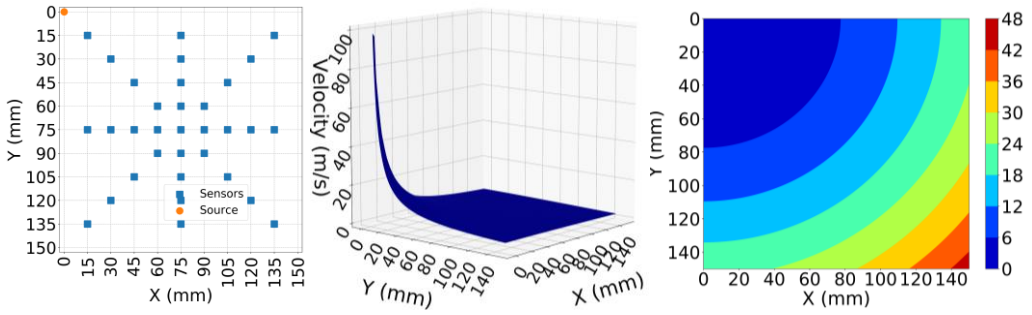


Figure 4-10. (a) Sensor and source location; (b) Velocity distribution; (c) Analytical solution of the arrival time.

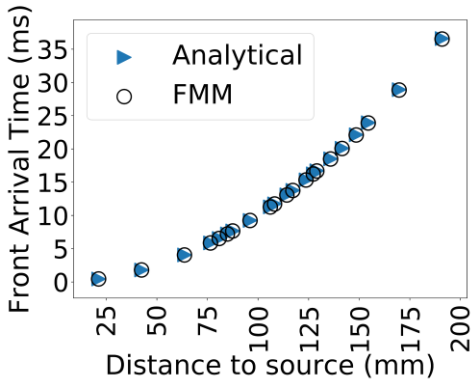
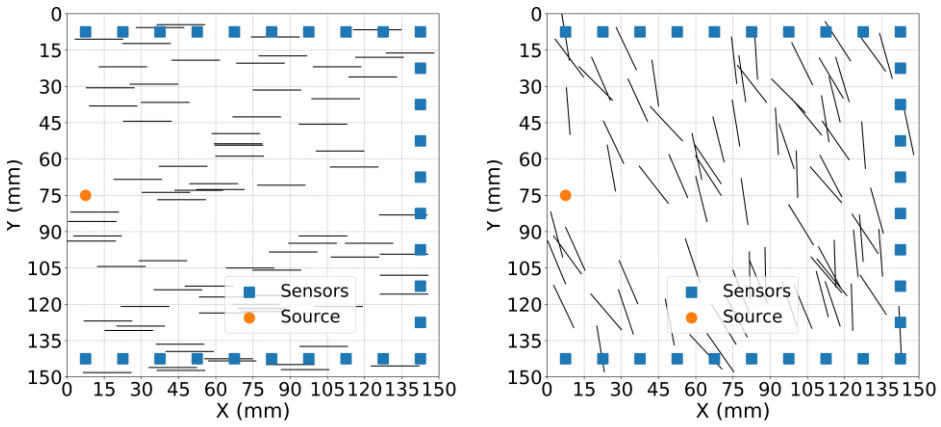


Figure 4-11. Comparison of sonic wavefront arrival time measured by each sensor calculated by FMM, and analytical method.

We can see that the FMM results are the same with the analytical solution.

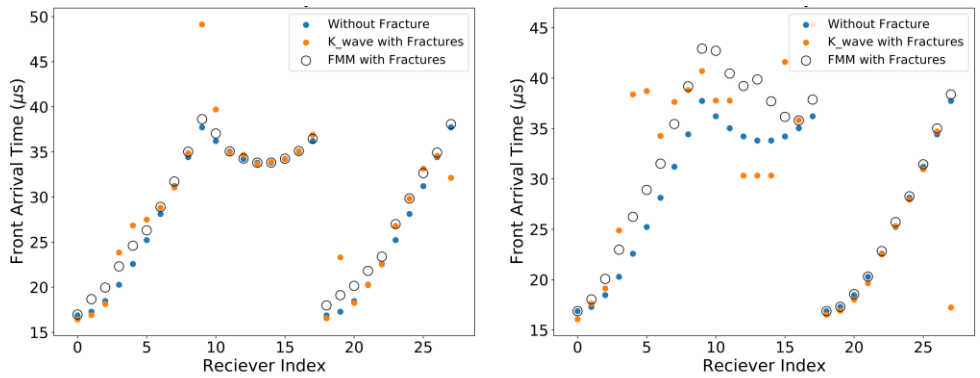
4.3.7 FMM Validation with Fracture Systems

In the last section, I validated the FMM algorithm with crack-bearing materials with embedded fracture systems. Two cases with different fracture systems are tested. In the first case, 100 horizontal fractures are embedded. In the second case, 100 fractures are embedded with the main orientation of 45 degrees. The two crack-bearing materials are shown in Figure 4-12.



(a) Case #1. (b) Case #2.

Figure 4-12. Sensors locations for FMM validation with fracture systems. In the first case, a horizontal fracture system is embedded; In the second case, fractures are mostly in the 45-degree orientation. Sonic wave speed in the matrix and fractures are 4000m/s and 400m/s respectively.



(a) Case #1. (b) Case #2.

Figure 4-13. Comparison of sonic wavefront arrival time measured by each sensor calculated by FMM, k-Wave, and analytical solution from materials without fractures.

In this experiment, analytical solutions are not applicable because the cases are too complicated. So, I compared the arrival times acquired by FMM, k-Wave, and from the materials without fractures. For case #1, the results are quite close. This is because the horizontal fractures do not block the wave propagation paths. For case #2, the FMM results are a little higher than the arrival time from the material without fractures. Since

there is no analytical solution, and the k-Wave simulate seems to get unstable results from this case. I can only say that the FMM simulation gets reasonable and stable results for the two cases.

4.4 Workflow to Generate the Training/Testing Dataset

4.4.1 Travel-Time Measurement Setup

The experiment setups, matrix and fracture parameters, the locations of the sensors and source are introduced in this section. After I validated the FMM algorithm, the next problem is about how to select the right parameters to build numerical crack-bearing material models that are comparable in the real laboratory environment. The 2D numerical models of crack-bearing material implemented in this study are inspired by the laboratory experiments conducted at the Integrated Core Characterization Center (IC3) presented in Bhoumick (Bhoumick et al., 2018) and Misra et al. (Misra et al., 2019a) who placed multiple sonic wave sources and receivers around a cylindrical rock samples to visualize crack distribution inside the rock samples.

In this study, I focus on characterizing fracture systems inside crack-bearing models with limited sonic data. The 2D models and the locations of the sonic wave source and receivers are designed as Figure 4-14. The digital crack-bearing material has a dimension of 150mm by 150mm. Dimension selection affects the computation time of the simulation. As illustrated in the previous section, when the dimension of the models 150mm by 150mm and the grid numbers are 500 by 500, the FMM simulation results are accurate enough. The machine learning models require a large number of training data. A larger dimension of the crack-bearing material needs a higher number of grid and a higher number of fractures, which increase the computation time. For the 150mm by

150mm model as shown in Figure 4-14, generating 10,000 simulation results takes around 2 hours for Dell workstation with 3.5GHz Intel Xeon CUP and 32GB RAM. As an exploratory study, the selected experiment setups are accurate enough for the goal of the study. When generating different models with fractures in the following section, the fracture system is controlled by some statistical parameters (e.g. main orientation, location distribution, etc.). Every 2D model is unique and different, but the fracture systems are generated from specific probability distributions.

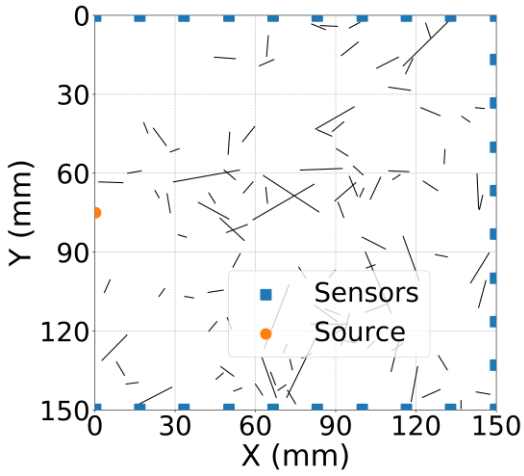


Figure 4-14. Digital crack-bearing material dimensions and locations of the sensors for the experiments in this study.

Both the compressional wave and shear wave can be used for crack-bearing material characterization. Since we only care about the wavefront in this study, only the compressional wave is studied. The sonic wave propagation speed is affected by a variety of parameters. Here we assume the matrix is clean sandstone. The compressional wave propagation speed in clean sandstone is around 4500m/s. The fracture is assumed to be filled with air. The compressional sonic wave propagation speed is set to 340m/s. In the following section, the wave velocity referred to the sonic compressional wave velocity by default. In this study, I use simple numerical models of crack-bearing material to increase

the speed of generating the required simulated travel-time dataset for purposes of proof of concept.

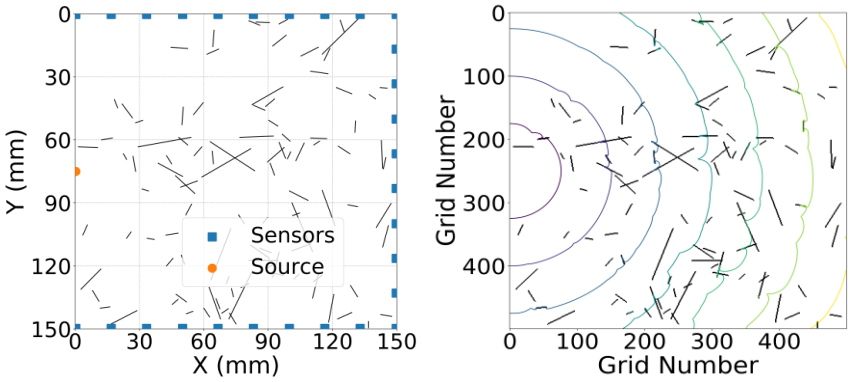
The digital crack-bearing model has a grid number of 500 by 500. The corresponding fracture width is 0.03mm. Increasing the model dimension or decrease the grid number will change the fracture width, affect the computation time, and simulation accuracy. The fracture system embedded in the fractured material is generated using the Discrete Fracture Network (DFN) method. The fracture system is determined by the location, orientation, and length of the fractures. The location, orientation, and length of the fractures are characterized by different distributions (Fadakar Alghalandis, 2014). The detailed methods and parameters used to generate the crack-bearing materials are listed in Table 4-1.

Table 4-1. The detailed fractured material simulation parameters.

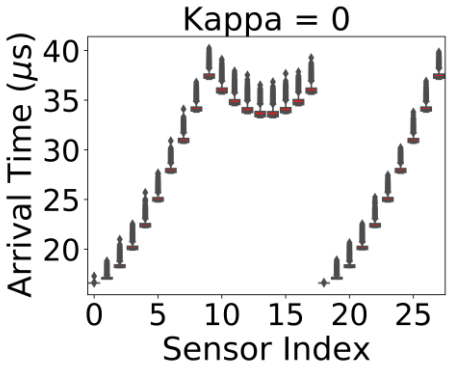
Parameters	Simulation method
Matrix dimension	150 mm by 150 mm
Signal source	1
Signal receiver	28
Number of fractures	100
Fracture length	3mm – 30mm simulated by the exponential distribution
Fracture orientation	Simulated by von Mises distribution
Fracture location	Controlled by intensity functions
Sonic wave speed in the matrix	4500m/s (Assuming clean sandstone)
Sonic wave speed in fracture	340m/s (Assuming filled with air)

4.4.2 Building the Dataset for developing the Classifiers

A large dataset is needed to train machine learning methods for characterizing the embedded crack clusters. Three experiments are conducted in this study to explore the possibility of classifying crack-bearing materials (also referred as the sample) with machine learning models on sonic wavefront travel time. The dataset is generated as follows. First, different groups of crack-bearing models are generated using DFN method. DFN method and certain probability distributions are used to quantify the crack cluster in a sample. FMM simulation is then conducted on each of the generated crack-bearing materials to simulate the compressional wave propagation. The wavefront arrival time is measured by each of the 28 sensors as features in the training dataset. The arrival time and the group number of each crack-bearing material are recorded as training data for machine learning models. 10,000 data points are generated for each type of model. Each data point is a 28-dimensional vector. The values of the vector are the arrival time measured by the receivers in the simulation. Qualitative description of the crack orientation, dispersion, and spatial distribution as discrete categorical integers serve as the targets. 9 machine learning models are trained and tested on the training dataset. Grid search and 5-fold cross-validation are used when training the models. The goal is to see if machine learning models can differentiate crack-bearing material fracture system types solely based on simple wavefront arrival times. The classification accuracy of the models is compared. The importance of the 28 input values is analyzed using the feature permutation method. The workflow of each experiment is shown in Figure 4-15.



(a) Numerical model of crack-bearing materials. (b) FMM simulation and extract arrival times.



KNN	Decision Tree	Naïve Bayes
Linear SVM	Random Forest	ANN
RBF SVM	Adaboost	Voting Classifier

(c) Arrival times recorded at 28 sensors.(d) Machine learning models applied on the dataset.

Figure 4-15. Workflow for the experiments.

Three experiments are conducted in this study. The experiments are explained and analyzed in the following sections. The first experiment aims to apply machine learning models to differentiated crack-bearing models with different fracture dispersion. The second experiment is to classify crack-bearing models with different fracture orientations. The third experiment classifies crack-bearing models by fracture locations. By performing the experiments above, we can get a general analysis of the feasibility of applying machine learning models on the classification of materials with different kinds of fracture systems. The sensitivity and robusticity of different machine learning models can be analyzed and compared.

4.5 Workflow to Train and Test the Classifiers

4.5.1 Introduction of All the Classifiers

In this study, 9 classification machine learning models are applied to the data generated from the 2D models. The 9 models are K Nearest Neighbors classifier (KNN), linear Support Vector Machine (SVM), Radial Basis Function (RBF) SVM, Decision Tree classifier, Random Forest classifier, Adaboost classifier, Naïve Bayes classifier, ANN classifier, Voting classifier. The 9 models applied consists of the most popular classifiers in machine learning. Some of the models have been applied in the regression problems in Chapter 3, like SVM, KNN, ANN. The classifiers applied here are based on the same principles as those regression models in Chapter 3. I will briefly introduce the models that are not introduced in Chapter 3.

The decision tree method is a popular non-parametric method. The number of parameters used in the decision tree is not pre-defined. The basic idea behind the decision tree method is to split the training dataset in a way that each part of the split training dataset is purer than before. Usually, some criteria, like information gain, are calculated to indicate the purity of each part of the split. The method can be used both in regression and classification. One advantage of the decision is that it can visualize the decision-making process.

Based on the decision tree method, the random forest method is created by training multiple decision tree models in parallel. Random forest is an ensemble method. In a random forest model, multiple decision tree models are trained based on a different part of the training dataset. The features and data samples used to train one decision tree

are usually randomly selected. The random forest makes a prediction by gathering the decision of the decision trees.

AdaBoost is another ensemble method. Instead of training a set of models in parallel, AdaBoost models training a set of models in series. Usually, simple decision models are used as submodels in the AdaBoost model. Each decision tree model has limited prediction ability. By training a set of simple decision trees in series, the AdaBoost model can gain great prediction ability. Each of the simple decision tree model focus on the weakness of the previous model. The weight of a sample will be boosted if the sample is not classified correctly. By combing multiple simple decision trees in series, the decision boundary of the AdaBoost becomes smoother and more accurate.

Naïve Bayes classifier classifies data based on the Bayes' theorem. The theorem is called Naïve Bayes because it is based on the assumption that all features are independent within each class, which seems is a "Naïve" assumption. However, the model works very well in real-world problems like spam email detection. The theorem is about how to update the existing knowledge (Prior) when new observations appear. The method uses the Bayes rule to calculate the probability of new observations. The predicted class will be the class with the maximum probability.

The voting method used in this study is a simple ensemble method. The model ensembles a set of classifiers and make a prediction by summarizing the classifiers' prediction. Each classifier makes a vote, the majority of the prediction is taken as the final prediction. The prediction of each classifier can be weighted to make the prediction more accurate.

4.5.2 Model Evaluation and Improvement Procedure

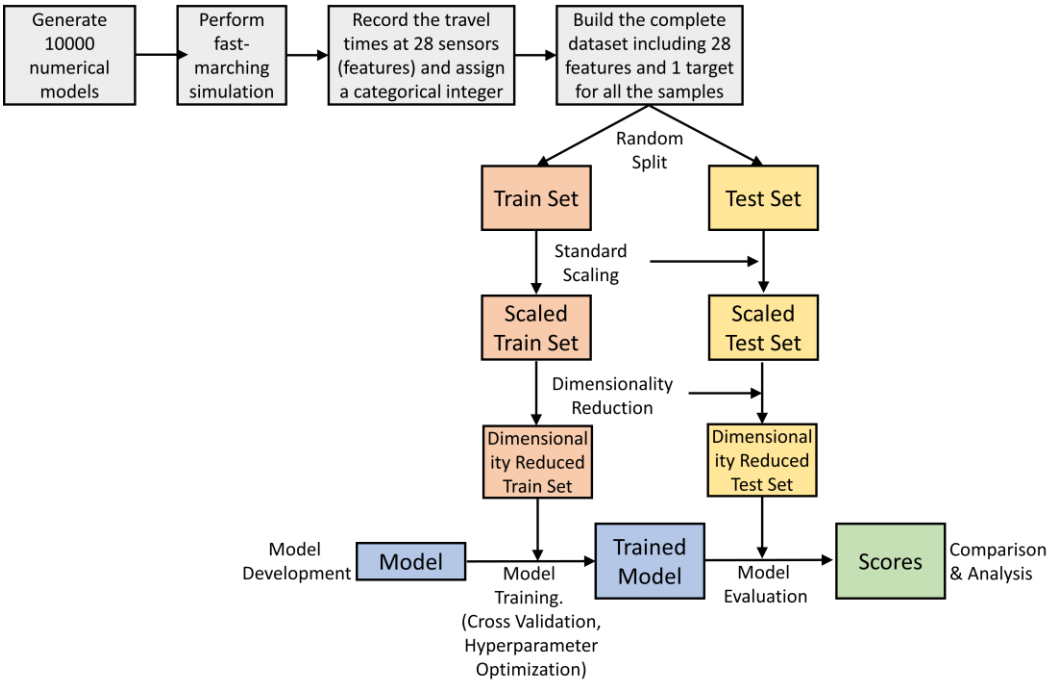


Figure 4-16. Data generation, Model development, and model evaluation process.

A detailed description of the workflow to train and test the classifiers can be seen in Figure 4-16. For each experiment in this study, different types of 2D crack-bearing models are built. Each type of model is embedded with fracture systems with different characteristics. 10,000 2D models are built for each type of fracture systems. FMM simulation will generate the front wave travel time as described in the previous section. After the dataset is acquired from FMM simulation, a standard classification procedure is performed on the dataset. The dataset is split into training and testing datasets with a ratio of 0.7 – 0.3. Classification models are trained on the training dataset. 5-fold cross-validation and grid search are used to tune the hyperparameters. The best model is tested on the testing dataset. The training and testing dataset are scaled to have zero mean and unit variance. The testing data are scaled with the same scaler as the training dataset. The experiment procedure is similar to the procedure in Chapter 3.

4.5.3 Statistical Relevance of The Predictions

The following procedures ensure the statistical relevance of the prediction. First, the dataset is large enough. As stated before. 10,000 simulation models are created for each type of model. For example, in the first experiment, three types of models are created with different fracture dispersion parameters. For each type of model, 10,000 simulation models are created. Totally 30,000 2D models are created for the experiments. This ensures that the training dataset is large enough for the machine learning models to learn from. The testing dataset is large enough to ensure the results are statistically stable. Second, the confidence interval is calculated for each of the models. Based on the prediction intervals, we can see if the prediction results of a model are statistical stable. Third, the prediction accuracy is considered high only if the prediction accuracy is much higher than the baseline. The baseline we use in this chapter is the theoretical prediction accuracy of a random classifier (a classifier that randomly predict the results). For a classification task with 4 classes, a random classifier has a theoretical prediction accuracy of 0.25. If a classifier learns some relationship between the features and the target, the classification accuracy should be higher than a random classifier. In this chapter, the baseline prediction accuracy is the prediction accuracy of a random classifier.

The correlation of the 28 measurements of experiments is calculated before used as train and test data. The analysis indicates that there is not a high correlation (>0.9) between the 28 features. This indicates that the wavefront arrival times measured by each sensor may contribute to the development of the models.

4.6 Data-Driven Modeling

4.6.1 Characterization of Crack Dispersion around a Dominant Orientation

4.6.1.1 Experiment Description

In this section, the machine learning models presented in previous sections are used to classify crack-bearing models based on the front wave arriving time. A set of 2D crack-bearing models is generated to simulate the sonic wave propagation process with the FMM algorithm. The fractures in the models are generated with a discrete fracture network modeling (DFN) method. The experiment in this section focus to investigate if machine learning models can classify crack-bearing models with different fracture orientation dispersions.

The fracture orientation is generated by the von-Mises distribution. The probability density function of the von-Mises distribution is (Mardia and Jupp, 2009):

$$\boldsymbol{f(x|\mu,k)} = \frac{\boldsymbol{e^{kcos(x-\mu)}}}{\boldsymbol{2\pi I_0(k)}} \tag{4-5}$$

where x is the main orientation, μ is a measure of the mode and k (kappa) is the dispersion factor. I_0 is the modified Bessel function of order 0. The mode μ and dispersion factor k are similar to the mean and standard deviation in the Gaussian distribution. The mode controls the mean of the distribution and the dispersion controls the spread of the distribution. von-Mises is a common distribution used to describe the orientation distribution of the fracture system.

I designed three crack-bearing models with different fracture dispersion. The dispersion (kappa) describes the randomness of the orientation from the main orientation. The three types of crack-bearing materials are shown in Figure 4-17.

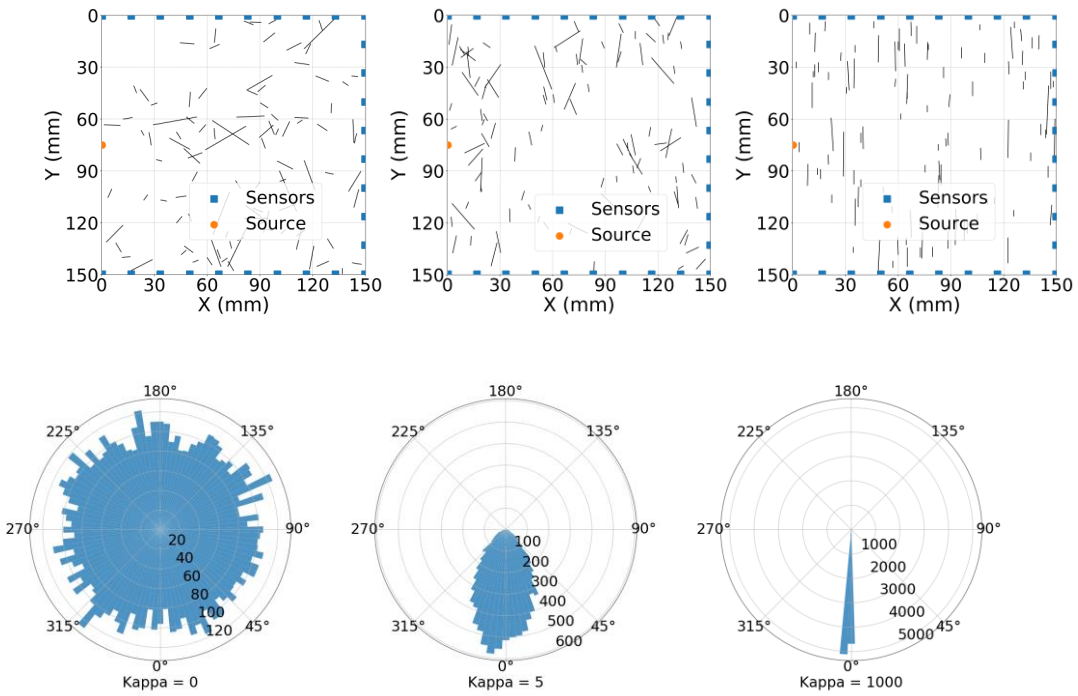


Figure 4-17. Crack-bearing materials (upper) and fracture distributions (lower) with different dispersion (kappa) factor. Left, kappa = 0, the fracture orientation is totally random; Middle, kappa = 5, the fracture orientation is centered at the 0 degrees and mostly between -50 and 50 degrees; Right, kappa = 1000, the fractures are almost parallel with the main orientation.

From Figure 4-17 we can see three different types of fracture systems with different orientation dispersions. The three different kinds of fracture systems have the same main orientation. The dispersion factor (kappa) of the fracture systems is set to 0, 5, 1000, respectively. The higher the kappa, the lower the dispersion. The lower three figures show the corresponding fracture orientation distributions of the three fracture systems. When the dispersion factor is set to 0, the orientations of the fractures are equally distributed in all directions. When dispersion is set to 5, the fractures orientations are concentrated around -50 – 50 degrees. When dispersion is set to 1000, the fracture orientations are almost parallel to each other. The sonic wave generation and measurement system are also shown in the figure. The compressional wave source is

located in the middle of the left boundary. The 28 receivers are equally located at the rest three boundaries.

The experiment is designed to investigate the machine learning models' sensitivity to fracture dispersions. Each type of fracture systems is common in the real-world. The fracture system may be random or parallel due to external forces, or partially parallel. For each type of the crack-bearing systems, 10,000 models are generated. Totally 30,000 models are generated. Each crack-bearing model is used to run an FMM simulation to generate a record of front wave arrival time. The record is basically a 28-dimensional vector that represents the arrival time measured by each sensor. The sonic wave propagation process is affected by the fracture system. We assume that by simply measuring the arrival times, we can acquire some information about the orientation dispersion of the fracture system. After the simulation, a data set with 30,000 records (rows) and 28 features (columns) are generated. Each record is labeled as one of the three types of the fracture system. A standard classification problem is solved by applying the 9 machine learning models.

4.6.1.2 Model Accuracy and Feature Importance

9 machine learning models are applied to the dataset. The models are KNN, SVM, RBF SVM, Decision Tree, Random Forest, Adaboost, Naïve Bayes, ANN, and Voting Classifier. These 9 models cover the most popular machine learning classifiers. The details about generating training and testing models, and about the training and testing the classifier are discussed in section 4.4 and 4.5. It is a standard and basic classification problem. The models are tuned with 5-fold cross-validation and grid search method. Test data is only used to test the best model.

The classification accuracy of the models on the testing dataset is shown in Figure 4-18. We can see that all 9 models have similar classification accuracy. The overall accuracy is around 0.6 for the 9 models. RBF SVM and Voting classifiers have the highest accuracy, which is 0.65. The confidence interval is also calculated. The 95% confidence interval indicates that there are 95% chance that the model's accuracy is in the given range. Since we tested the model on a large testing dataset, the confidence interval is small (around +/- 0005).

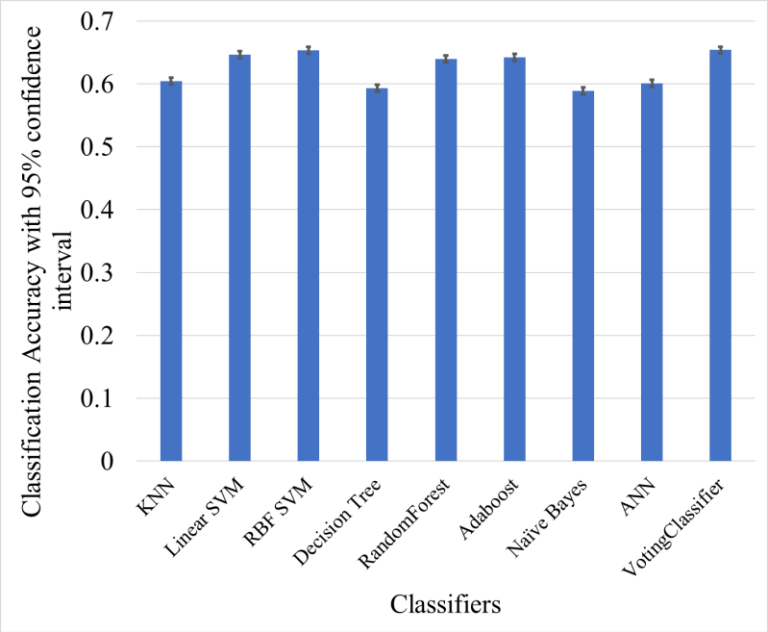


Figure 4-18. Classification accuracy with 95% confidence interval of the 9 classifiers.

The results show that most of the models do not perform well on classifying materials with different fracture dispersions. This is due to the fact that several cracks in these three categories of crack-bearing materials have similar orientations that result in loss of representative information between the three categories/types. The machine learning models distinguish different models by measuring the difference in the input

data. If the front wave travel times are similar for all the three types of fracture systems, the machine learning models can not classify them with high accuracy.

One way to answer the poor classification accuracy is to analyze the wave propagation route between the source and the sensors. If we draw a line between the source and each of the sensors, we will get 28 lines that represent the sonic wave propagation paths. Since the locations of the fractures in the three types of fracture systems are random, we can assume that the sonic wave propagation paths should intersect a similar number of fractures in the three types of fracture systems. This is especially true for the sensor located opposite to the source. This sensor located at the same location on three different types of fracture systems measures similar front wave arrival time. The dispersion of the fractures has a limited effect on arrival times. This can be further explained in the following sections.

I further examined the experiment results by calculating the precision, recall, and f1-score of the voting classifier. The voting classifier has the best performance. The results are shown in Table 4-2. The precision and recall are low for classes 1 and 2. The two cases are too similar to each other.

Table 4-2. Classification report of the voting classifier model.

	Precision	**Recall**	**F1-score**	**Support**
0	0.73	0.78	0.75	3003
1	0.53	0.4	0.46	2999
2	0.66	0.78	0.72	2998
avg / total	0.64	0.65	0.64	9000

I further investigated the disadvantage of the current experiment configuration by calculating the feature importance. The feature importance is calculated by the feature permutation method. It evaluates the feature importance by shuffling each feature and calculate the prediction accuracy drop. The assumption is that if a feature is important to a machine learning model, by replacing the feature, we should observe a high accuracy drop. The feature importance for the voting classifier is shown in Figure 4-19. It tells us which sensor is more important for the classification task.

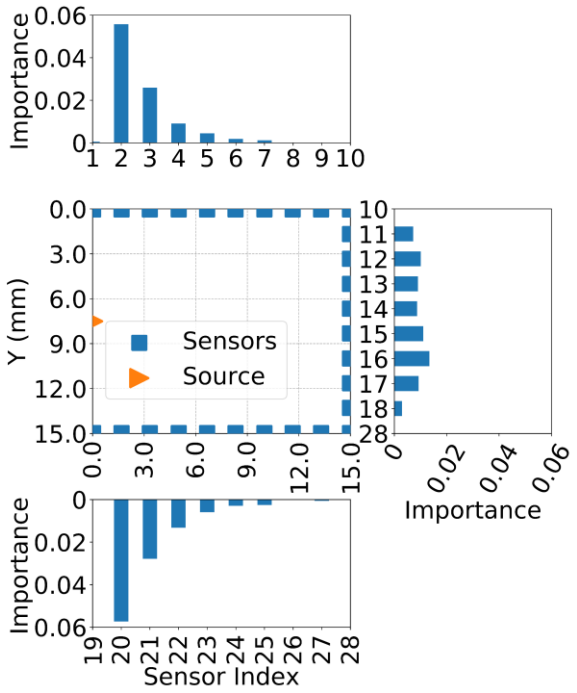


Figure 4-19. Feature importance for the voting classifier calculated by feature permutation.

The results are similar to our previous analysis. The sensors on the upper and lower boundary are more important. Figure 4-19 shows the importance of each sensor. The figure in the middle represents a model. The three subplots around the figure represent the importance of each sensor. The sensors are numbered from 1- 28. The sensor on the right boundary is less important because the arrival time measure by the sensors located on the right boundary is similar for the three types of crack-bearing

models. This is because the fractures' locations are random. When a sonic wave propagates from the source to the sensor on the right boundary, statistically, it should encounter a similar number of fractures. The dispersion of the fractures does not have a large effect on the arrival time on the right boundary.

Sensor #2 and #20 are the most important sensors. This is because the change of the dispersion is reflected in these two sensors. Take case 0 and case2 as examples, in case 0, the fractures are totally random in orientation. For case 2, the fractures are almost vertical. It is obvious that when a sonic wave propagates from source to sensor #2 or sensor #20, the probability of encountering a fracture is much smaller if the fracture orientation is parallel to the wave propagation direction. In case 2, the fracture orientation is almost parallel to the sonic wave propagation direction for sensor #2 and #20. The existence of fractures delays the propagation of the waves, which is the key for machine learning models to classify the crack-bearing models. This is why sensor #2 and #20 are important.

4.6.2 Characterization of the Dominant Crack Orientation

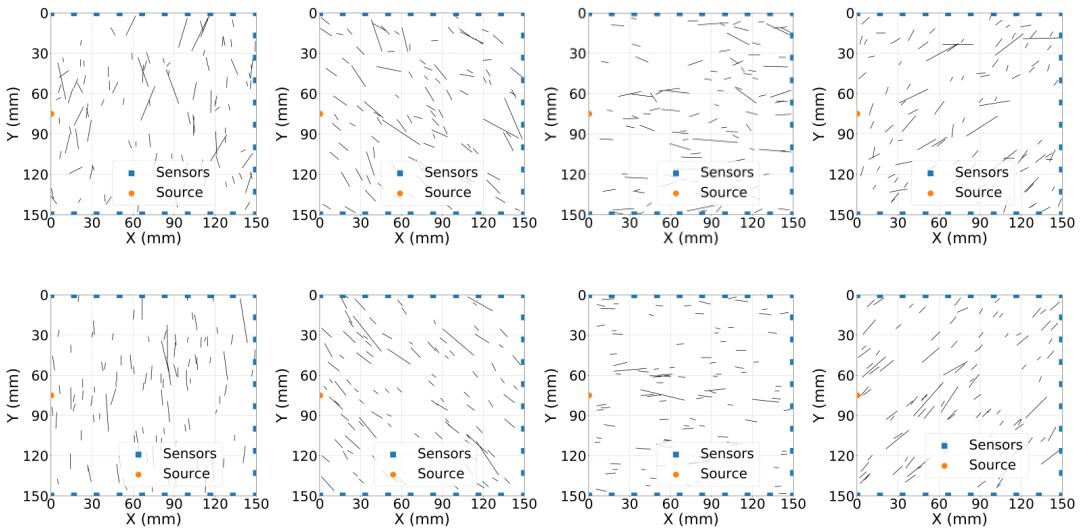
4.6.2.1 Description of The Experiment

In this section, I designed four numerical experiments to investigate the performance of the data-driven classifiers in identifying the dominant orientation of the crack cluster embedded in the material. The four experiments are created by changing the dominant crack orientation and the crack dispersion. The basic source-sensor configuration is the same as the previous experiment. The fracture systems in different groups are generated with the parameters in Table 4-3.

Table 4-3. Mode and dispersion (Kappa) of the von-Mises distribution parameters for the four numerical experiments to investigate the performance of the data-driven classifiers in identifying the dominant orientation of the crack cluster embedded in the material.

Experiments	Modes	Kappa (Orientation range)
#1	0°, 45°, 90°,135°	10 (+50°~-50°)
#2	0°, 45°, 90°,135°	50 (+20°~-20°)
#3	0°, 22.5°, 45°, 67.5°90°, 112.5°135°, 157.5°	10 (+50°~-50°)
#4	0°, 22.5°, 45°, 67.5°90°, 112.5°135°, 157.5°	50 (+20°~-20°)

The four experiments are similar to the experiment in the previous section. Each of the four experiments in this section has a different combination of main fracture orientation (mode) and the fracture dispersion factor (kappa). The first two experiments each contains 4 different types of fracture systems. The last two experiments contain 8 different types of the fracture system. The first experiment has four types of crack-bearing models. Each type of crack-bearing model is embedded with a fracture system with different main orientations. The dispersion factor of the fracture system is set to 10 for the first experiment. The second experiment is similar to the first one, but the dispersion factor is set to 50. The las two experiments are similar to the first two but with 8 different main orientations. Examples of the crack-bearing models are shown in Figure 4-20. The crack-bearing models of experiments #3 and #4 are shown in Figure A-1 and Figure A-2 in the Appendix section.



(a) Orientation = 0° (b) Orientation = 45° (c) Orientation = 90° (d) Orientation = 135°

Figure 4-20. Crack-bearing model examples from experiment #1 (upper) and experiment #2 (lower).

The same machine learning models are applied to the data generated from the 4 experiments as the experiment in the previous section. The model accuracy and fracture importance evaluation are investigated in the next section.

4.6.2.2 Model Accuracy and Feature Importance

For each of the 4 experiments, 10,000 simulations are conducted for each type of crack-bearing model. For experiment #1 and #2, 40,000 wavefront arrival time data points are gathered for the 4 types of crack-bearing models with different fracture orientations. For experiment #3 and #4, 80,000 data points are gathered. The 9 machine learning models applied are tuned using grid search and 5-fold cross-validation. The classification accuracy of the models is shown in Figure 4-21. A detailed classification accuracy and confidence interval data can be found in Table A-2.

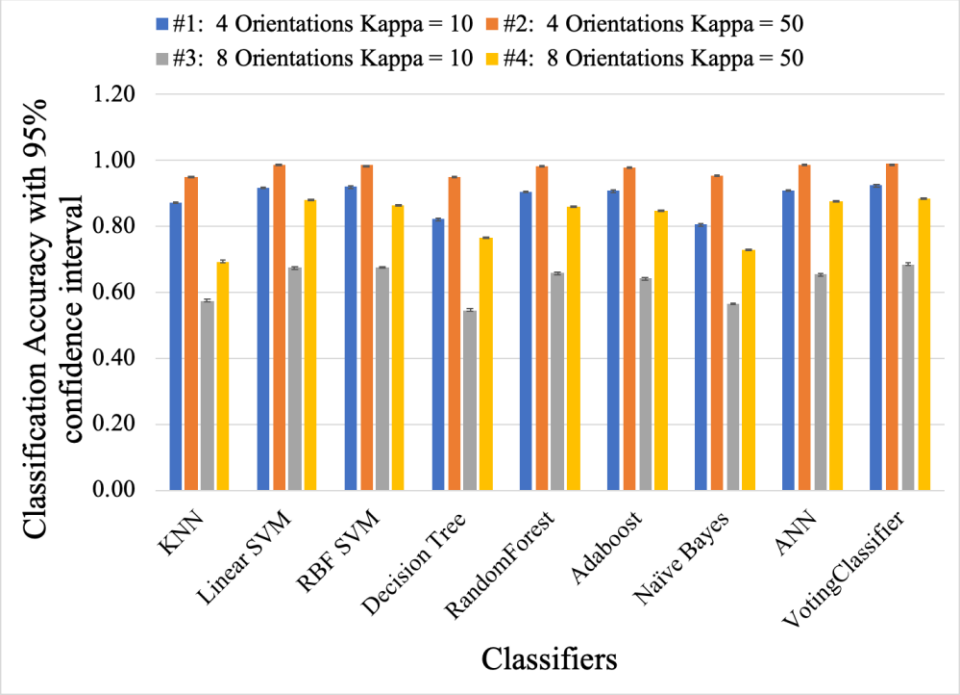
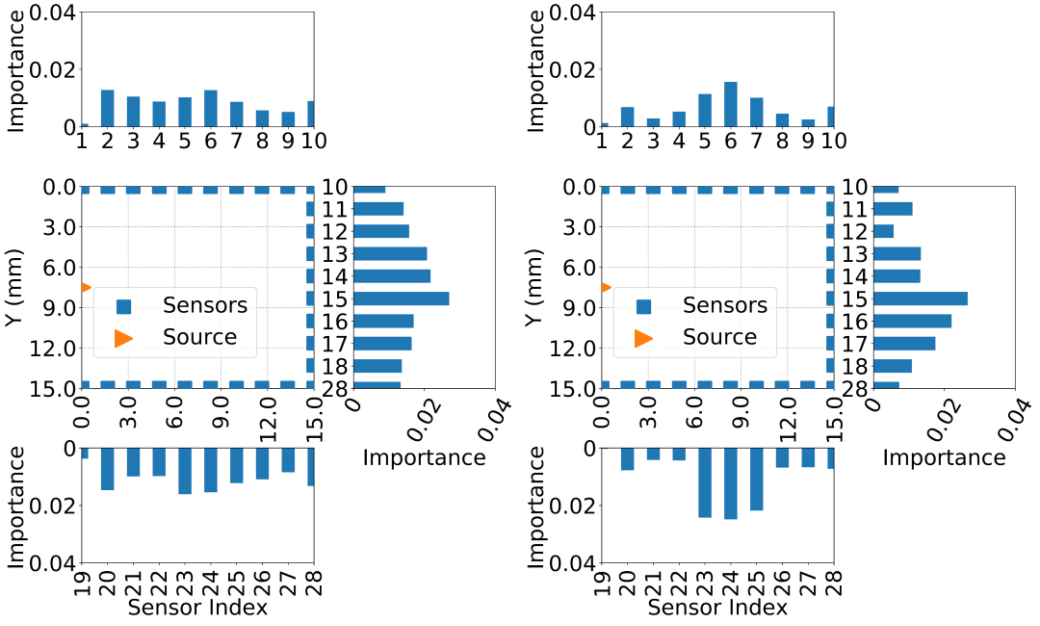


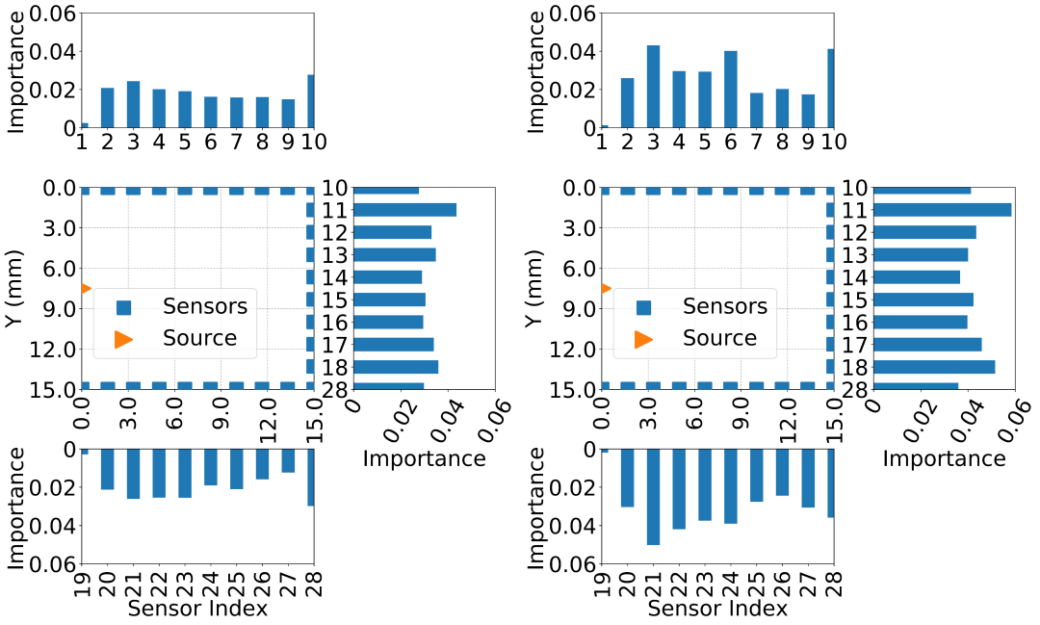
Figure 4-21. Classification accuracy with a 95% confidence interval (a small error bar can be observed on the top of each bar) for the 9 classifiers on the 4 experiments.

Similar to the experiment in the previous section, the SVM and voting classifier performs better than other models in all of the 4 experiments. We can also observe that the accuracy of experiment #1 and #2 are higher than #3 and #4. This is easy to understand since the first two experiments contain fewer types of fracture systems and thus the difference between two different cases is larger. Also, we can see that the classification accuracy is also affected by the dispersion factor (kappa). When the dispersion factor is higher, which corresponding to a more parallel fracture system, the classification accuracy is higher. This is also easy to understand. When fracture systems are parallel, the difference between different types of fracture systems is higher. An extreme case is when kappa is set to 0. In this case, the orientation will be totally random, and no model can distinguish the cases.

Feature importance is also calculated on the voting classifier. The feature importance calculated by feature permutation is shown in Figure 4-22.



(a) Experiment #1. 4 orientations, kappa=10. (b) Experiment #2. 4 orientations, kappa=50.



(c) Experiment #3. 8 orientations, kappa=10.(d) Experiment #4. 8 orientations, kappa=50.

Figure 4-22. Feature importance calculated by feature permutation for each of the 4 experiments with the voting classifier.

Compared to the experiment in the previous section, the 4 experiments in this section show different feature importance distributions. By comparing Figure 4-22 and Figure 4-19, we can see that sensors located on the right boundary are important for classification, and the upper and lower boundary sensors are also important. Compared to Figure 4-19, the difference shows that the sensors have a different level of importance for different kinds of measurement. For this experiment, the change of the fracture orientations affects the sonic wave arrival time on different sensors. When the crack cluster is oriented along the horizontal with low dispersion, the sensors on the boundary opposite to source will measure lower travel times and those on the upper and lower boundaries will measure larger travel times as compared to the case without any embedded cracks. An opposite trend will be observed when crack clusters are oriented in vertical direction parallel to the boundary on which the source is placed.

4.6.3 Characterization of the Spatial Distribution of Cracks

4.6.3.1 Description of The Experiment

In this section, an experiment is conducted to classify four different types of fracture systems with different fracture locations. Fracture locations are affected by the geomechanical properties as well as and the surrounding environment. Four different types of fracture systems are created by changing the fracture intensity function. The intensity function describes the possibility of the fractures' distribution in space. Given an intensity function, a fracture system can be generated using the inhomogeneous Poisson process (Fadakar Alghalandis, 2014; Illian et al., 2008).

Four different types of fracture systems are generated. Fracture intensity functions and the corresponding fractured models are shown in Figure 4-23. The inhomogeneous

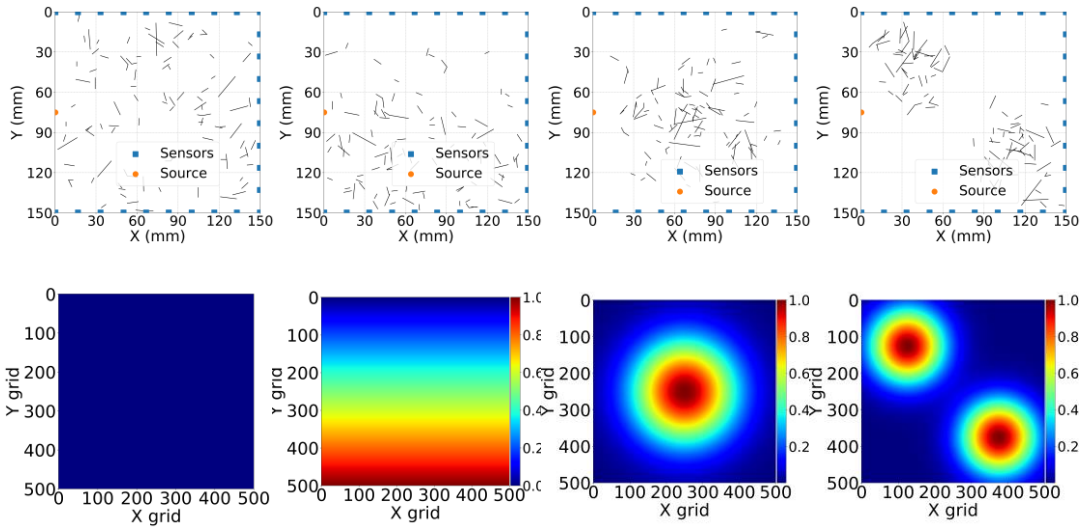
Poisson process is often called the 'rejection method'. Combined with the intensity function, the Poisson process can generate crack system according to the intensity function. The intensity function defines the probability of the cracks in the domain of investigation. An intensity function is defined in the domain of investigation *W*. The intensity function $\lambda(x)$ is normalized by divide the maximum value on the domain to generate an acceptance probability. The maximum value is defined as:

$$\boldsymbol{\lambda^* = \underset{x \in W}{max}\, \lambda(x)} \tag{4-6}$$

The inhomogeneous process is a random process that generates potential fracture locations. After a location is generated, the inhomogeneous Poisson process decided whether to reject or retain the point by the acceptance probability calculated as

$$\boldsymbol{p(x_i) = \lambda(x_i)/\lambda^*} \tag{4-7}$$

Sample fracture models generated from different fracture intensity functions are shown in Figure 4-23.



(a) Random (b) Linear (c) Single Gaussian (d) Double Gaussian

Figure 4-23. Four different intensity functions (lower) and corresponding crack-bearing models (upper).

Four different types of fracture system with different location distribution is shown in Figure 4-23. The goal of this experiment is to classify the four types of fracture systems. The four types of fracture locations are simple and easy to implement. The first one utilizes a constant intensity function, which means the fractures have an equal probability of appearance everywhere in the domain. The results are the same as a crack-bearing model where fractures are located randomly.

Figure 4-23b shows a crack-bearing model generated with linear probability function:

$$\boldsymbol{\lambda(x, y) = y} \tag{4-8}$$

The possibility of acceptance increases linearly with the value of the y-axis. The origin of the coordinate is located at the upper left corner of the crack-bearing model. The acceptance probability is only related to the y-axis. The fractures in this type of crack-bearing models have a higher probability located near the lower boundary.

Figure 4-23c shows a crack-bearing model generated with a Gaussian function as intensity function:

$$\boldsymbol{\lambda(x, y) = c * exp^{(-d*((x-x0)^2+(y-y0)^2))}} \tag{4-9}$$

where $x0$ and $y0$ is the center of the Gaussian distribution, d controls the variance of the distribution, c controls the minimum value of the intensity function. In this experiment, c is set to 1, d is set to 0.00005, $x0$ and $y0$ are both set to 250, which is the center of the grid. The acceptance possibility is high at the center of the crack-bearing models.

Figure 4-23d shows a crack-bearing model generated by adding two Gaussian functions together. The model is generated with the same Gaussian functions as Equation (4-9). The difference is that the center and the variance of the two Gaussian functions.

The experimental procedure is the same as in previous experiments. With 4 types of fracture systems, a total of 40,000 crack-bearing models are generated. 10,000 simulations are conducted for each of the four types of crack-bearing model. Each type of crack-bearing model is embedded with 100 fractures. The fracture orientations are totally random. The same 9 models are implemented in this section. Feature importance is also calculated in the following section.

4.6.3.2 Model Accuracy and Feature Importance

The classification accuracy of the 9 machine learning models is shown in Figure 4-24. Similar to previous experiments, the voting classifier is still the best performing classifier. The accuracy of the voting classifier is 0.86, which is good compared to other models. Adaboost and random forest also performs quite well. Naïve Bayes is the worst performing model.

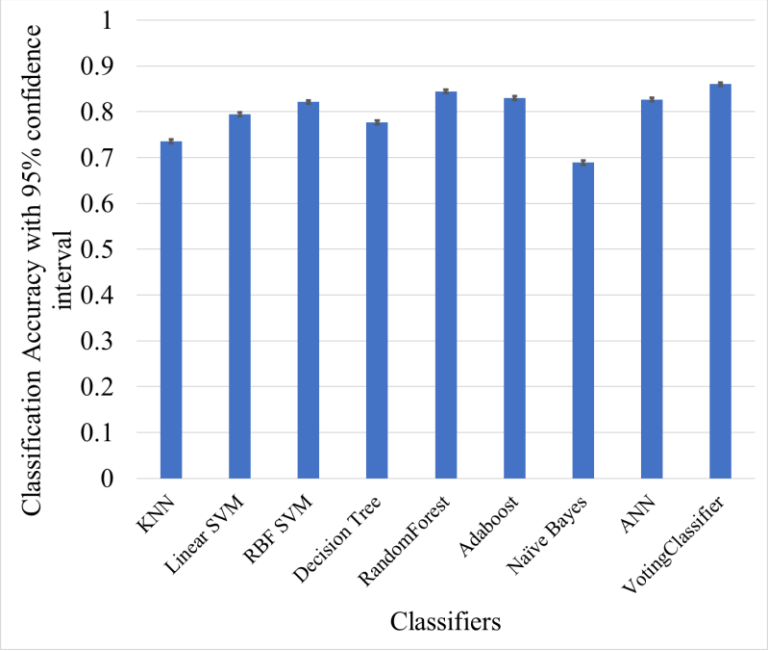


Figure 4-24. Classification accuracy with 95% confidence interval for the 9 classifiers on the arrival time generated from 4 types of fracture locations.

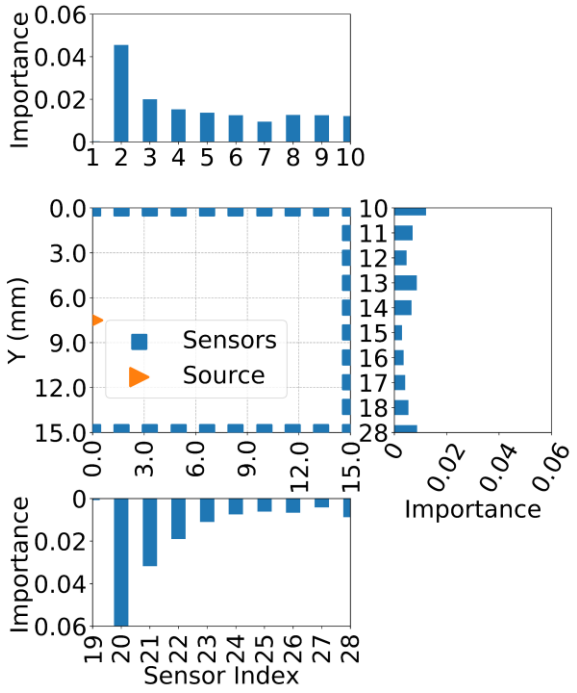


Figure 4-25. Feature importance calculated by feature permutation for with voting classifier.

The feature importance of this experiment is shown in Figure 4-25. We can see that the results resemble the feature importance in Figure 4-19. Sensor #2 and #20 are the two most important sensors, which means these two sensors provide the most important information for the machine learning models to distinguish the 4 types of fracture systems. Moreover, it seems sensor #20 is more important than sensor #2.

As I analyzed in previous sections, the feature importance can be explained from the perspective of the arrival time difference. By connecting the source to sensor #2 and #20, we get two lines that represent the sonic wave propagation routes. If no fractures intersect with these two lines, the arrival times measured by the two sensors should be exactly the same, and thus the sensors provide no useful information to distinguish the models. However, by comparing the location distribution of the four types of fracture

systems, we can easily see that the two sensors are useful for the models because the two propagation directions provide distinct arrival time signatures for different models.

For the first fracture system, the fractures are randomly located. Statistically, the two sensors should measure similar arrival times, but both arrival times are delayed by the existence of fractures. For the second type of fracture system, it is obvious that it takes longer for the sonic wave to propagate to the sensor #20 compared to sensor #2. For the third type of fracture system, the fractures are concentrated around the center. The propagation of sonic waves to the two sensors should encounter few fractures. The two sensors should measure similar arrival times. For the last type of fracture system, it is obvious the sensor #2 measure larger arrival time compared to sensor #20. For the four types of fracture systems, the two sensors can provide four distinct types of arrival times signatures. We have good reasons to believe the machine learning models can learn from the distinct signatures measured by the sensors to make the correct classification. The prediction processes inside machine learning models are more complex. Every sensor contributes to the final prediction. This analysis provides a possible explanation of the decision-making process of the models.

4.6.4 Effect of Noise on the Data-Driven Classifiers

In the previous section, I investigated the possibility of classifying materials with different fracture systems with machine learning models and FMM sonic wave simulations. In this section, experiments are conducted to analyze the models' responses to data with noise. Noise is a common occurrence in the measurements of the oil and gas industry. Well logging measurements are suffered from noise due to the complex downhole environment. Laboratory core measurements are suffered from the noise due to

the laboratory environment background, the equipment precision, and sample quality. The core sample characterization experiments with sonic waves may subject to environmental noise in a laboratory. Models' robusticity to noise should be considered when choosing the right machine learning model. To investigate the models' robustness to noisy data, I trained the 9 machine learning models with data that are corrupted with noise. Based on the experiments in this section, I found that different models exhibit different resistivity to noisy data, and training models with noisy data may improve the models' robusticity.

4.6.4.1 Description of The Experiment

The dataset used in this section is generated from the classification experiment with different fracture locations in section 4.6.3. The original dataset is scaled to have zero mean and unit variance. In the following section, I will call the original dataset without any noise the "clean data". To investigate the effect of noise on the models' performance, different level of noise is generated and added to the training and testing dataset. The noise is generated by Gaussian Distribution $\mathcal{N}(0, \theta)$. The level of noise is adjusted by changing the variance θ of the Gaussian distribution from 0 to 0.2 with a step of 0.01. Totally 20 different training and testing datasets are generated with different levels of noise. In the following section, I will call the 20 datasets that are added with noise the "noisy data". All the 9 models applied in previous sections are tested with noisy data.

Three types of experiments are conducted to generated comparison results on the models' resistivity to noise. First, the 9 models are trained with clean training data. Then the models' performance is calculated on the noisy testing dataset. Second, the 9 models

are trained with noisy training dataset, then the performance is tested on the noisy testing dataset that contains the same level of noise with the noisy training dataset. Third, the 9 models are trained with noise training dataset but tested on the clean testing dataset.

4.6.4.2 Model Performance Comparison

In the first experiment, the 9 models are trained on the clean training data and tested on the noisy testing data. Different levels of noise are added to the testing data to test the models' resistivity to noise. The classification accuracy of the 9 models on the testing dataset is shown in Figure 4-26.

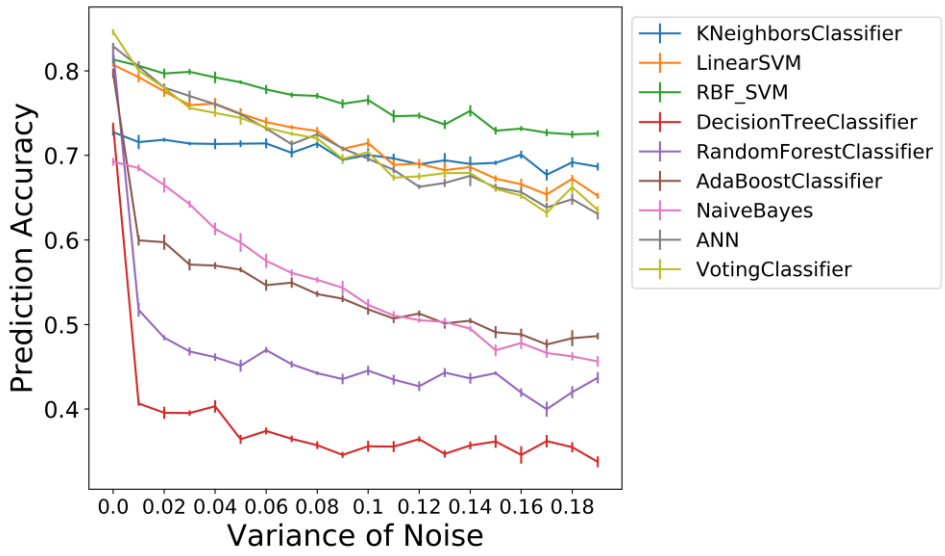


Figure 4-26. Experiment #1. Performance of models trained on the clean training dataset and tested on the noisy testing dataset with different levels of noise. The error bar represents the 25 percentiles and 75 percentiles of the prediction accuracies. The accuracy is calculated 20 times with resampled dataset.

Figure 4-26 shows the 9 models' classification accuracy on the testing dataset with respect to the variance of noise in the testing dataset. The 9 models are all trained with the clean training dataset. From Figure 4-26, we can see that different model exhibits different resistivity to noise. Based on intuition, the increase in the level of noise in the testing dataset results in deterioration of the performance of the classifiers.

However, some models show significant accuracy drop when the noise is added to the testing dataset. Decision tree classifier, gradient boosting classifier, Adaboost classifier, and random forest classifier show significant accuracy drop when noise with 0.01 variance is added. This demonstrates that these models are not robust to noise in the testing dataset.

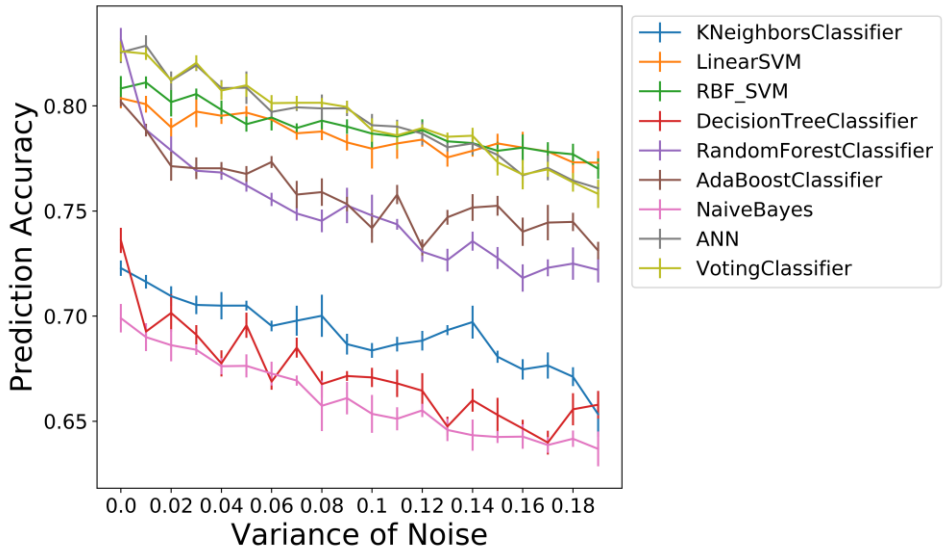


Figure 4-27. Experiment #2. Performance of models trained on the noisy training dataset and tested on the noisy testing dataset with different levels of noise. The error bar represents the 25 percentiles and 75 percentiles of the prediction accuracies. The accuracy is calculated 20 times with resampled dataset.

In the second experiment, I added noise in both the training dataset and the testing dataset. The models' performance is shown in Figure 4-27. Compared to previous experiments, we can see that the models' resistivity to noise in the testing dataset is improved. After adding noise with variance equals to 0.2 in both training and testing dataset, we observe around 5% accuracy drop for most of the models. Compared to the results shown in Figure 4-26, none of the classifier exhibit sharp drop in performance when small amount of noise is added to both the training and testing datasets. In summary, it is important to have little noise in the training dataset.

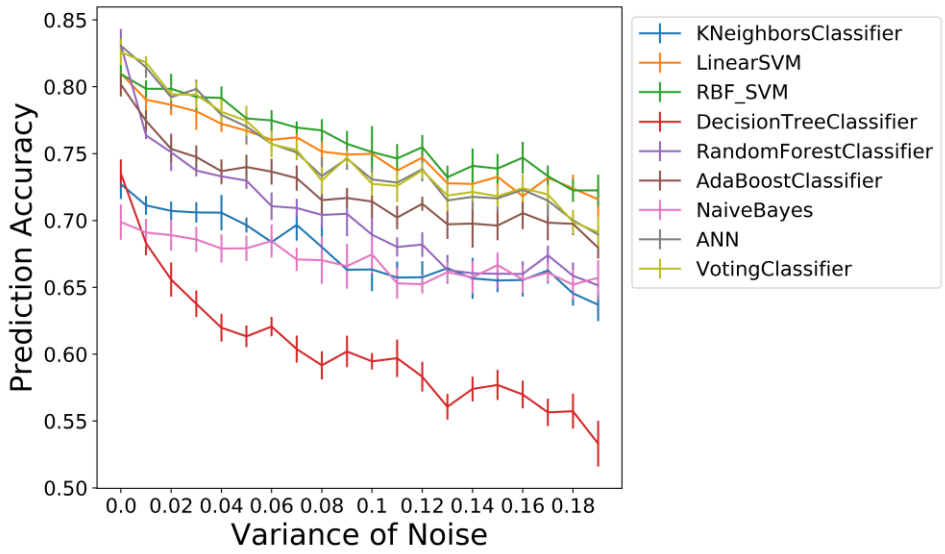


Figure 4-28. Experiment #3. Performance of models trained on the noisy training dataset and tested on the clean testing dataset. The error bar represents the 25 percentiles and 75 percentiles of the prediction accuracies. The accuracy is calculated 20 times with resampled dataset.

Figure 4-28 shows the models' performance when trained with a noisy training dataset and tested on the clean testing dataset. Compared to the first experiment, most of the models do not exhibit significant accuracy drop. This demonstrates that adding noise in the training dataset may improve the accuracy of the models on the noisy testing dataset. Compared to the second experiment, models in experiment 3 shows high accuracy drops on the testing dataset. For models in experiment 3, around 10% accuracy drop are observed when adding noise with 0.2 variances on the training dataset.

By comparing these three experiments, we can see that different machine learning models show different resistivity on noise in data. KNN, ANN SVC models show good resistivity to noise in the testing dataset. We observe that by adding noise in the training dataset, we can improve the models' resistivity to noise. The models have a higher prediction accuracy. However, adding noise in the training dataset also increased the variance of the accuracy. The error bars in both Figure 4-27 and Figure 4-28 are larger

compared to Figure 4-26. When applying machine learning models on the dataset that may suffer from different levels of noise, adding noise in a training dataset may be a possible way to improve the robusticity of the models.

4.7 Novelty of This Study

- The numerical model of crack-bearing models and simulations of traveltime measurements related to wavefront propagation are based on real-world laboratory experiments.
- A limited number of source-receiver pairs were used to generate the simulated data for training the classifier. In addition, each receiver measures very limited information related to wave propagation through a crack-bearing model, which is the sonic travel time (first arrival).
- Feature importance is performed using feature permutation importance to identify the most important receivers for accomplishing the desired classification task.
- The study shows that statistical parameters of the fracture system, like main orientation and locations intensity, can be acquired with simple source-receiver configurations.
- The study provides a way to design the best experiment configuration for crack-bearing model characterization with wavefront travel time information.

4.8 Assumptions and Limitations

The data-driven classifiers were trained and tested on simulated dataset generated for simple sonic wave propagation through simple crack-bearing materials. Extensive study is required to understand the generalization of the proposed data-driven workflow

for laboratory measurements of crack-bearing materials. Major assumptions and limitations of the proposed data-driven workflow include:

- The materials under investigation are simplified to 2D and the fractures are considered as linear mechanical discontinuities. Fracture systems are simulated using a discrete fracture network modeling (DFN) method.
- Machine learning methods are developed only on sonic compressional travel times. Effects of wave reflection, scattering, and dispersion are ignored.
- Sonic front travel time is simulated using the fast-marching method, which can have errors in the presence of sharp contrasts.
- The models trained in this study are not general enough to be applied in the real world.
- The sonic wave investigated in this study is assumed to be a compressional wave. The effect of the shear wave will be investigated in future studies.
- The source and receivers are assumed to be attached to the boundary of the material. It is assumed that no distance between the material and the sensors.
- The fractures and pores are assumed to be filled with air under laboratory conditions.

4.9 Recommendations for Future Work

- Understand the physical basis of decision making by the best performing classifiers. In doing so, identify signal trends/behavior/components that can be used for facilitating transfer learning on materials with different source-receiver configurations and the material specifications.

- Develop a robust data-driven model to characterize the fracture system as fracture evolves from start to coalescence.
- Use both compressional and shear full waveforms with multiple sources to improve the desired machine learning tasks.

4.10 Comparison and Conclusions

In this study, a noninvasive material characterization method by measuring the sonic wave arrival times is proposed. The method is inspired by a real-world experiment. The experiments generated 2D crack-bearing models with different types of fracture systems. The fracture systems are created by the DFN method. FMM simulation is implemented to simulate the sonic wave propagation process. A sonic wave generation and measurement experiment configuration are designed. 28 sensors are embedded in the crack-bearing models to measure the front wave arrival times. The arrival times are used as input to train and test 9 selected machine learning models. The following conclusion can be drawn from this study:

The current source-sensor experiment configuration is informative enough to classify crack-bearing models with different fracture orientations and locations, but it cannot provide enough information to classify crack-bearing models with different fracture dispersions.

Voting classifier outperforms all other models in this study.

The experiments show that machine learning models exhibit high classification accuracy on classifying models with different orientations. Four experiments with 4 and 8 different orientations, the voting classifier achieves accuracy of 0.99 and 0.89.

Machine learning models can differentiate crack-bearing models with random, unimodal, bimodal, and linear intensity fracture locations with high accuracy (0.86).

Different sensors show different importance in different experiments. The arrival times measured by the sensors on the upper and lower boundary are more important to classify fracture systems with different dispersion and location. On the contrary, the sensors on the right boundary are more important to classify fracture systems with different orientations.

The experiment source-sensor configuration provides a simple way to characterize fracture systems with limited information. Machine learning models, especially ensemble models show great ability in interpreting simple sonic arrival times. Full sonic waveforms may be needed to accurately describe the distribution and evolution of the fractures.

Different machine learning models exhibit a different level of resistivity to noise. When applying machine learning models on datasets that may suffer from different levels of noise, adding noise in a training dataset may be a possible way to improve the robusticity of the models.

Different machine learning models exhibit a different level of resistivity to noise. Adding noise to the training dataset improves the models' resistivity to noise.

4.11 Acknowledgements

The research findings in this chapter are based upon work supported by the U.S. Department of Energy, Office of Science, Office of Basic Energy Sciences, Chemical Sciences Geosciences, and Biosciences Division, under Award Number DE-SC-00019266.

Chapter 5 Reinforcement Learning for Automatic History Matching

History matching aims to construct a reservoir model that can reproduce historical reservoir performance. The purpose of history matching is to augment the reservoir model with information from production data so that forecasts are more reliable. This requires an engineer to modify select reservoir parameters to match the production history of the reservoir while integrating information from geologic analysis, geophysical surveys, petrophysics, and production data. Assisted or automatic history matching workflows require the selection of uncertain reservoir parameters, following which optimization algorithms explore the parameter spaces to decrease the misfit between the model predictions and the reservoir responses. In this study, I develop and apply a reinforcement learning technique to for history matching. Reinforcement learning utilizes a learning agent to learn interactively from an environment to maximize the expected reward. It has been applied in situations like game playing and self-driving cars to achieve human-level automatic control.

Instead of a conventional reservoir simulator, I use a fast-marching pressure propagation reservoir simulator to predict model response from a select set of inputs. A learning agent interacts with the reservoir simulator to match production history. I use the Discrete and continuous reinforcement learning algorithms, Deep Q Network (DQN) and Deep Deterministic Policy Gradients (DDPG), respectively as the learning agents. The learning process begins with the learning agent predicting the reservoir parameters, and then reservoir simulations uses the predicted reservoir parameters to calculate the misfit. The misfit is calculated using the predicted production history (pressure drop and pressure derivative) and the actual production history. The actual production history is

always known for a particular well. The predicted production history is generated by running reservoir simulations. Based on the misfit, the learning agent receives a reward that reinforces successful actions taken by the learning agent. In doing so, the agent learns to adjust its prediction of reservoir parameters to automate the history matching process. Reinforcement learning balances exploration and exploitation during the learning process (Sutton and Barto, 2018). Sufficient exploration of the parameter space provides a greater chance of finding the global optimum while exploitation applies learnings from prior exploration. However, the exploration process may also result in more training episodes.

I test the effectiveness of the proposed reinforcement learning algorithms in history matching on a circular single layer reservoir. I test various learning agents with 14 synthetic cases, with 7 cases assuming a uniform distribution of physical properties and the other 7 cases assuming a composite model with two concentric volumes. In this study, I observe that DDPG performs better than DQN. DDPG can potentially find a strategy to decrease the misfit using continuous action. DQN discretizes the action space and is therefore limited.

To test the reinforcement learning algorithms further, I test a more realistic implementation of the permeability map. The permeability map is generated using the exponential spatial covariance function. These results also indicate that the DDPG algorithm successfully finds a strategy to minimize the misfit.

5.1 Background

History matching integrates multiple types of data to build a reliable reservoir model capable of predicting the future performance of a reservoir. Geologic

interpretation, well log and core data as well as geophysical surveys are used to build an initial static model which is further refined using dynamic data such as production rates and pressure measurements (Avansi and Schiozer, 2015). The integration of dynamic data is known as history matching and involves fitting the historical data by adjusting the model parameters. The matching process is an optimization process, which involves calibration of several parameters to decrease the misfit between model predictions and production history. The misfit is usually evaluated by an objective function. Because of the limited number of measurements in comparison to model parameters, the solution to history matching is often non-unique.

A large number of techniques have been developed to facilitate the matching procedure. Reservoir characterization using limited sampled spatial data requires geostatistical methods like Kriging to predict reservoir properties throughout the whole reservoir (Korjani et al., 2016; Zhao et al., 2018). In a reservoir where high heterogeneity exists, a large number of uncertain parameters need to be adjusted (Zhang et al., 2017) and sometimes, parameterization is applied to reduce the number of unknows by transformation-domain or spatial methods (Khaninezhad and Jafarpour, 2014). Spatial methods divide grids into zones assuming homogenous reservoir properties, whereas transformation-domain methods transform high dimensional parameters into low dimension by recognizing correlations in features. Various dimensionality reduction algorithms are Principle Component Analysis (PCA), Kernel PCA, as well as deep learning algorithms of Autoencoder (Canchumuni et al., 2017).

The history matching process can be classified into different categories such as manual, assisted and automatic history matching. Manual history matching requires

engineers' experience to select and calibrate uncertain properties. The boundary between assisted and automatic history matching is not clear. Some researchers identify automatic history matching as those that utilize an objective function and view the process as an optimization problem and assisted history matching is human-centered and computers only facilitate the searching of the optimum point in the parameter space (Sanghyun and Stephen, 2018). Other researchers believe that assisted history matching is a more accurate term since engineers' knowledge and experience are more or less involved in the matching process (Cancelliere et al., 2011; Emerick, 2017).

Based on the matching algorithm applied, history matching can also be identified as deterministic (gradient-based), stochastic or probabilistic (Kazemi and Stephen, 2012). Gradient-based optimization algorithms like Gauss-Newton (GN) and the Levenberg-Marquardt (LM) algorithms, use sensitivities of model responses to reservoir parameters to find a local minimum in the objective function. Gradient-based algorithms scale well with large problems (Shirangi and Emerick, 2016) but can potentially be stuck in a local minima (Mitchell, 1997). For extremely large problems, calculation of the sensitivities can be computationally expensive. Compared to deterministic optimization algorithms, stochastic algorithms include randomness in the exploration of the parameter space and thus may enable the algorithm to escape a local minimum and search a larger area in the parameter space (Mitchell, 1997). Stochastic algorithms such as the Genetic Algorithm and Particle Swarm Optimization have been found efficient at dealing with history matching problems where large number of parameters are involved (Sanghyun and Stephen, 2018).

Probabilistic algorithms like Ensemble Kalman Filter (EnKF) and Maximum Likelihood rely on a Bayesian framework (Vink et al., 2015). Probabilistic algorithms aim to infer a set of models with combinations of parameters that match production history. EnKF is a gradient-free algorithm, which relies on the initial ensemble to generates multiple history-matched models (Ding et al., 2016). EnKF method is a Monte-Carlo implementation of the Bayesian update, where probability density function (PDF) is updated when new data is available (Mandel, 2009). Markov Chain Monte Carlo (MCMC) is another popular method in history matching for the purpose of uncertainty characterization (Elsakout et al., 2015; Olalotiti-Lawal and Datta-Gupta, 2015). MCMC provides a method to sample the posterior pdf for reservoir properties but it may require a large number of iterations for high dimensional problems (Emerick and Reynolds, 2011).

Irrespective of the method chosen, the history matching process requires a large number of forward runs of the simulator which can be computationally expensive. Proxy models and the Fast Marching algorithm have been applied to provide rapid, approximate solutions to a reservoir simulator. Proxy models interpolate between the input parameters and the corresponding misfits to reduce the computation resources to calculate a model response. Different data fitting and interpolation techniques of Kriging, radial basis function, etc. and machine learning algorithms of artificial neural networks, have been applied as proxy models (Shams et al., 2017).

Currently, there are only a few applications of reinforcement learning in the oil and gas industry. One application (Talavera et al., 2010) discusses optimal control policy for well production based on a reinforcement learning algorithm. Reinforcement learning has also been applied in directional drilling control, where the algorithm is trained

together with a drilling simulator to minimize the deviation from the planned wellbore trajectory (Pollock et al., 2018). To the best of my knowledge, RL has not been applied to the area of history matching. The motivation behind this study is to evaluate the applicability of RL to the history matching problem.

The history matching problem is not suitable to be solved using classification algorithms. This is because the target of the history matching problem is continuous values. The history matching problem is hard to be solve using supervised learning algorithms. This is because the targets of this task (permeability, porosity etc.) cannot be acquired in real world. In history matching, we can only predict the target by analyzing the oil well production history. In this chapter, I used reinforcement learning to learn to solve the history matching problem in a trial and error way.

This chapter aims to answer the following questions.

- Can reinforcement learning algorithms be applied for the purpose of automatic history matching?
- What are the advantages and limitations of the reinforcement learning algorithms of DQN and DDPG?
- Can automatic history matching be turned into a sequential decision-making problem in order to suite the reinforcement learning algorithm framework?
- How to turn a reservoir simulator into a reinforcement learning environment?
- Is DDPG algorithm suitable for the purpose of automatic history matching?
- How to generate a realistic reservoir permeability map using a small number of parameters for the purpose of automatic history matching?

- Can reinforcement learning algorithms be applied to generate permeability maps using exponential covariance function?

5.2 RL-Based History Matching Process

5.2.1 Agent, Environment, Action, and Reward

Reinforcement learning algorithms aim to solve problems involving a sequential decision-making process. The learning process involves several key components and concepts of learning agent, environment, reward, state, action, policy and value function, etc. (Sutton and Barto, 1998). A learning agent algorithm mimics the human learning process and interacts with the environment. The interaction between the agent and the environment is implemented by states, actions, and rewards. The state describes the current state of the environment. The learning agent must exert an action on the environment based on the current state and will receive the corresponding reward and new state. The learning agent estimates the value function, which depicts the goodness of the current state in terms of cumulative future reward. The ultimate goal is to learn a policy that maximizes expected future rewards.

In the case of history matching, the environment is the reservoir simulator, the action is a change in some reservoir parameter, the state is the value of the reservoir parameter(s) and the reward is related to the misfit between actual and model-predicted data. History matching typically requires exhaustive exploration of the parameter space which can be prohibitively expensive for large dimensional parameter spaces (Cancelliere et al., 2011; Sanghyun and Stephen, 2018). To balance exploration and exploitation, reinforcement learning is defined as a sequential decision-making problem.

The process involves a number of episodes which in turn involve a number of iterations. In each episode, the environment starts with the initial guess of the reservoir parameters and the learning agent takes actions in each iteration to alter the reservoir parameters. Following this step, the environment (simulator) provides the predicted production values from which a misfit between the predicted and true production history is calculated using the Mean Squared Error (MSE) in this study. The reward is negative MSE, which will increase when misfit decreases.

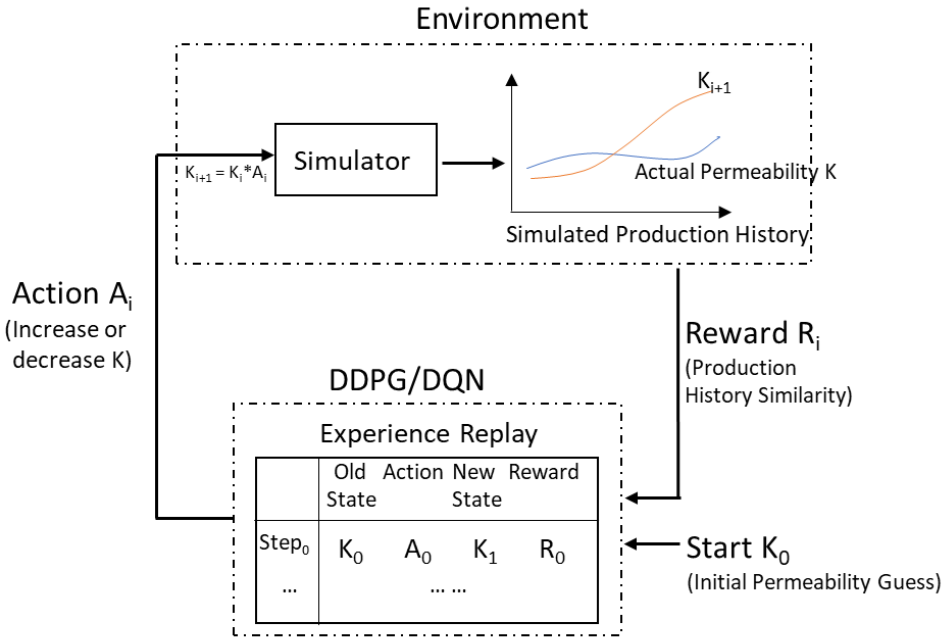


Figure 5-1. Flow chart of automatic history matching with the RL algorithms.

Figure 5-1 depicts the reinforcement learning process used in this study. The process starts with an initial guess of the reservoir permeability (initial state) at the beginning of each episode. Each episode is designed to have 10 iterations (10 actions in each episode to gain a reward). After a simulation run with the current state, the algorithms take specific actions to increase or decrease the permeability. The actions exerted on the environment lead to a new simulation run and a new calculated reward.

The transition experience $e_t = (s_t, a_t, r_t, s_{t+1})$, which is a combination of current state, action, reward, and new state at iteration *t+1*, will be stored for later learning.

The reinforcement learning algorithms that I use in this study are DQN and DDPG. DQN learns a mapping (usually a table) from state action pairs to the Q value, which is basically describes the cumulated reward of an action in a state. The DQN update the Q table based on the interaction history. DDPG algorithm is another popular reinforcement learning algorithm. DDPG accomplishes this task by introducing an actor-critic approach which is based on the Deterministic Policy Gradient (DPG) (Silver et al., 2014). DDPG can be used for problems with continuous actions.

5.3 RL Algorithms

DQN was originally developed to conquer complex game playing tasks with unprocessed, high-dimensional, and sensory input. DQN can learn policies from the unprocessed game image and reach a human-level performance (Mnih et al., 2015). However, DQN can only handle problems with discrete actions. DDPG overcomes this problem and is capable of addressing continuous and high dimensional action spaces (Lillicrap et al., 2015).

5.3.1 Introduction of the DQN Method

5.3.1.1 Basic Principles

The goal of reinforcement learning using the DQN method is to learn a good policy π from the sequence of states, actions, and rewards to maximize the cumulative future reward. A policy is a function that maps states to actions. For this study, a state is a set of permeabilities, an action is a set of permeability multipliers, and the reward is the negative mean squared error of the predicted and actual pressure drop and pressure

derivative. The cumulative future reward of an action at a specific state following a policy π is estimated by the action-value function (or Q function), which in our case is a neural network. The optimum action-value function is denoted as (Mnih et al., 2015):

$$\boldsymbol{Q^*(s,a) = \max_{\pi} \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots | s_t = s, a_t = a, \pi]} \quad (5\text{-}1)$$

where γ is a discount factor range from 0 to 1 and r is the reward at each iteration time. s, a represent state and action respectively. This equation indicates that the optimal action-value function is the one that maximizes the expected future cumulative reward given current state, action, and policy. The optimal action-value function tells us how much future reward it will accumulate in this game if it takes action a at state s and follow the policy π. The discount factor γ balances the current and future reward.

Experience replay is applied to address the instability when using a neural network as an action-value function. The agent's experience $e_t = (s_t, a_t, r_t, s_{t+1})$ at time *t* is stored as the experience dataset which is a combination of state, action, reward at time *t* and the new state at *t+1*. The action-value function was updated using a batch of randomly selected experiences. The loss function when updating the action-value function is denoted as follows (Mnih et al., 2015):

$$\boldsymbol{L_i(\theta_i) = \mathbb{E}_{(s,a,r,s')\sim U(D)} \left[\left(r + \gamma \max_{a'} Q(s', a'; \theta_i^-) - Q(s, a; \theta_i) \right)^2 \right]} \quad (5\text{-}2)$$

Equation (5-2) delineates the goal of our action-value function or the Q network developed by Mnih et al. (2015). $\boldsymbol{s, a}$ represent the state and action at the current step. $\boldsymbol{s', a'}$ represent the state and action at next step. $\boldsymbol{\theta_i}$ are the parameters of the Q network. θ_i^- are the parameters of the Q network used to calculate the results. The notation $(s, a, r, s') \sim U(D)$ means batch of randomly selected experiences are uniformly sampled from the experience dataset and θ_i is the weight of the network at iteration i. Since the Q

function is a mapping from state-action pairs to the expected future cumulative reward, the first two terms inside the bracket $r + \gamma \max_{a'} Q(s', a'; \theta_i^-)$ represent the updated target calculated using θ_i^- before iteration; and $Q(s, a; \theta_i)$ is the predicted target at iteration i. The Q network is updated by backpropagation of errors.

As we can see in Equation (5-1), the action-value function estimates the value for each discrete action. For problems with continuous actions, DQN does not have the ability to solve them efficiently.

5.3.1.2 Structures

In our study, I applied a neural network with two hidden layers, each hidden layer has 20 neurons. The input layer takes the state, which is the reservoir permeability, as input. The output layers output the predicted values for each action. The discount factor γ is set as 0.95. The 5 actions during each iteration are specified as 5 permeability multipliers of values 0.5, 0.7, 1, 1.3 and 2. In each episode, the algorithm takes 10 iterations. In each iteration, the algorithm can take one of the five actions and receive a reward from the environment. After an action is taken, the reservoir permeability is updated using the multiplier operating on the current permeability. The algorithm finds actions to maximize expected future reward by training the models using prior experience (previously visited states or experience replay). As the Q network experiences more and more cases, it will gradually find the correct state to maximize the expected reward. A good reward means the RL predicted reservoir parameters are very close to the actual reservoir parameters.

5.3.2 Introduction of the DDPG Method

5.3.2.1 Basic Principles

DDPG algorithm is another popular reinforcement learning algorithm. It overcomes some the disadvantages of DQN. Unlike DQN, DDPG is suited for problems involving continuous actions, which in this study represents continuous permeability multiplier. DDPG accomplishes this task by introducing an actor-critic approach which is based on the Deterministic Policy Gradient (DPG) (Silver et al., 2014). Compared with DQN where one network is used to predict action values and the action with the largest value is selected, the DDPG algorithm uses two neural networks to represent the policy and action-value function separately as shown in Figure 5-2.

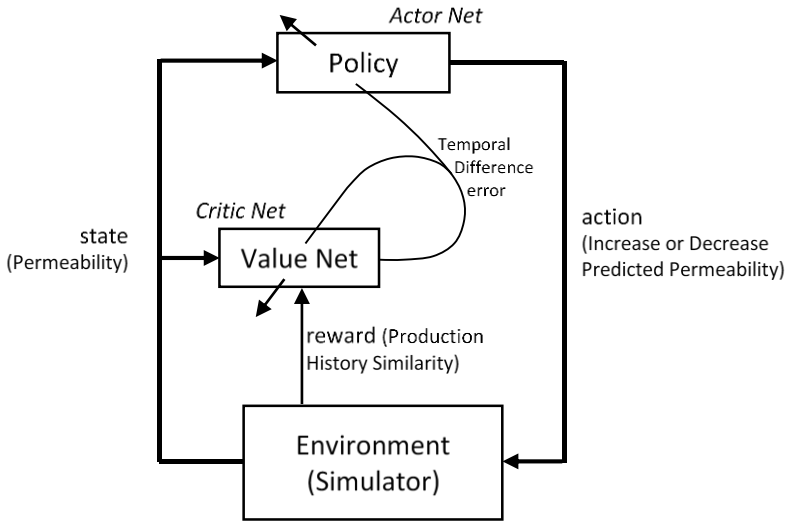


Figure 5-2. Actor-Critic algorithm adopted in the Deep Deterministic Policy Gradient algorithm, adapted from (Sutton and Barto, 1998).

Figure 5-2 explains how the DDPG algorithm learns to interact with an environment. DDPG consists of two neural networks, the actor net, and the critic net. The actor-network predicts the next action (permeability multipliers) and the policy network predicts the value of action at the current state (current permeabilities). The policy

function, or the actor-network, is a mapping from states to actions. The action-value function (Q function), or the critic-network, is a mapping from state-action pairs to the estimated cumulative future reward. The critic-network is updated using the same principle as Equation (5-1). The actor-network, or the policy, is updated using the following gradient (Lillicrap et al., 2015; Silver et al., 2014):

$$\nabla_{\theta^{\mu}} J \approx \mathbb{E}_{s_t \sim \rho^{\beta}}\left[\nabla_{\theta^{\mu}} Q(s, a|\theta^{Q})|_{s=s_t, a=\mu(s_t|\theta^{\mu})}\right]$$

$$= \mathbb{E}_{s_t \sim \rho^{\beta}}\left[\nabla_{a} Q(s, a|\theta^{Q})|_{s=s_t, a=\mu(s_t)} \nabla_{\theta_{\mu}} \mu(s|\theta^{\mu})|_{s=s_t}\right] \quad (5\text{-}3)$$

where μ_θ represents a deterministic policy parameterized by θ (the weight of the actor-network), $J(\mu_\theta)$ is the expected performance objective of the actor, $\mathbb{E}_{s_t \sim \rho^{\beta}}$ denotes the expected value when the state follows the distribution of ρ^{β}, which is the state distribution ρ generated from stochastic behavior policy β.

Equation (5-3) represents the gradient of the policy's performance, hence it is call policy gradient (Lillicrap et al., 2015; Silver et al., 2014). Equation (5-3) simply means that the gradient used to update the deterministic policy, or the actor-network, should be the gradient of expected cumulative reward with respect to the parameters (weight) in the actor-network. The second line of Equation (5-3) applied the chain rule to the first line of the equation.

5.3.2.2 Structures

The DDPG parameters in this study are adapted from the study by Lillicrap et. al (Lillicrap et al., 2015). The basic structures are the same, but I change a few parameters based on our experiments. The discount factor γ is 0.99, Adam optimizer (Kingma and Ba, 2014) is used with a learning rate of 0.0005 for both actor and critic network. The soft target update factor τ is 0.001. Both the actor-network and critic-network have 4

hidden layers, each containing 100 neurons. Similar to the original DDPG research, the action inputs were not included in the critic-network until the 2nd hidden layer. The actor-network contains 3 hidden layers with 100 neurons in each layer. The critic-network contains 4 hidden layers with 100 neurons in each layer.

5.4 Simulation Model and Environment

5.4.1 Introduction to The MFM Reservoir Simulation

History matching problems generally necessitate a large number of reservoir simulation runs which can make the process prohibitively expensive for large field studies. In order to decrease simulation time, I use the fast-marching method (FMM) to solve for reservoir/well pressures with reasonable accuracy. FMM relies on tracking the spatial location of the pressure front within the reservoir. (Karlsen et al., 2000). Multistencils Fast Marching (MFM), initially proposed for computer vision problems has now been applied for reservoir simulation problems to further improve simulation speed (Hassouna and Farag, 2007; Yoon, 2017).

Although extremely rapid, the class of models that can be investigated using the FMM are limited (Leem et al., 2015; Li and King, 2016; Onishi et al., 2019; Terada, 2019). FMM is usually used to simulate the wave front. FMM is not widely used in complex reservoir simulations problems that involves multi-phase flow, complex fracture systems etc.

5.4.2 Description of Reservoir Properties

In this study, I use an isotropic single-layered circular, dead oil model. I construct two set of cases that differ in permeability values and reservoir zonation. The first set of cases consist of a homogeneous and isotropic reservoir with a single permeability value,

referred to as the single-zone model. The reinforcement learning algorithm needs to estimate only one permeability value for the single-zone models. For the second set of cases, the model comprises of two concentric zones with different permeabilities; this will be referred to as the dual-zone model. Therefore, two values of permeability are to be matched. The detailed reservoir and well properties are shown in the following tables and figures.

5.4.2.1 Basic Reservoir and Well Properties

Table 5-1. Default reservoir properties for all cases

Reservoir Radius (ft)	Grid Size (ft^2)	Initial Pressure (psi)	Reservoir Thickness (ft)	Porosity	Permeability (md)
4040	40 × 40	6000	10	0.25	Differ for each case

Table 5-2. Default well reservoir fluid properties for all cases

Well property	Fluid property		
Well radius (ft)	Total Compressibility (psi^{-1})	Flow Rate (bbl/day)	Viscosity (cp)
0.25	0.000006	1	0.4

5.4.2.2 Actual Values and Initial Guess of the Reservoir Permeability for the Single-Zone Models

Table 5-3. Initial guess and actual permeabilities of the single-zone models

Case number	Initial guess of permeability (md)	Actual permeability (md)
1	100	1
2	100	5
3	100	10
4	100	50
5	1	100
6	1	50
7	1	10

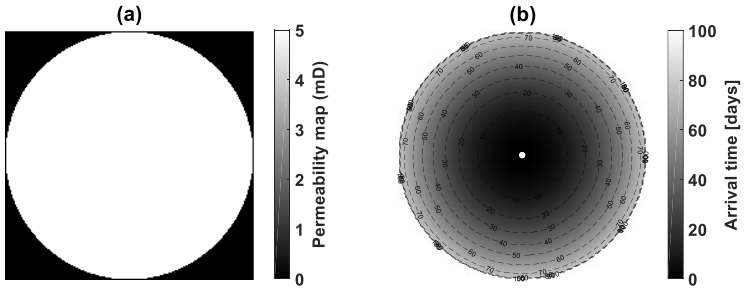


Figure 5-3. Reservoir permeability map (left) and pressure arrival time (right) for case 2 in Table 5-3.

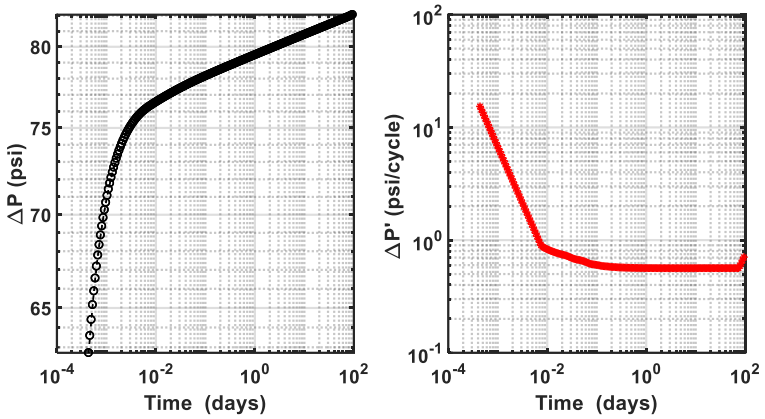


Figure 5-4. The diagnostic plot of pressure changes and Bourdet-type pressure derivative responses for 100 days for case 2 in Table 5-3.

Figure 5-3 shows the permeability map and pressure arrival time simulated with default reservoir parameters (Table 5-1) and actual permeability of 5md. The corresponding pressure changes and pressure derivative responses for the first 100 days are shown in Figure 5-4. The pressure derivative is the derivative of the pressure drop.

I design 7 different cases as shown in Table 5-3 with different initial permeability values and the goal is to estimate permeability that is close to the actual permeability values by matching the pressure and pressure derivative responses.

5.4.2.3 Actual Values and Reservoir Permeability Map and Initial Guesses of the Reservoir Permeability for the Dual-Zone Models

Table 5-4. initial guess and actual permeabilities of the dual-zone models.

Case number	Initial guess of permeability (md)		Actual permeability (md)	
	Inner	Outer	Inner	Outer
1	100	100	1	5
2	100	100	1	10
3	100	100	1	50

4	100	100	50	1
5	50	50	100	1
6	10	10	20	50
7	1	1	5	20

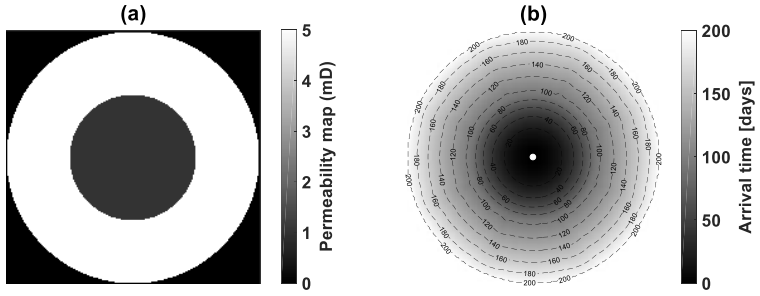


Figure 5-5. Reservoir permeability map (left) and pressure arrival time (right) with the permeability of 1 md and 5 md for case 1 in Table 5-4.

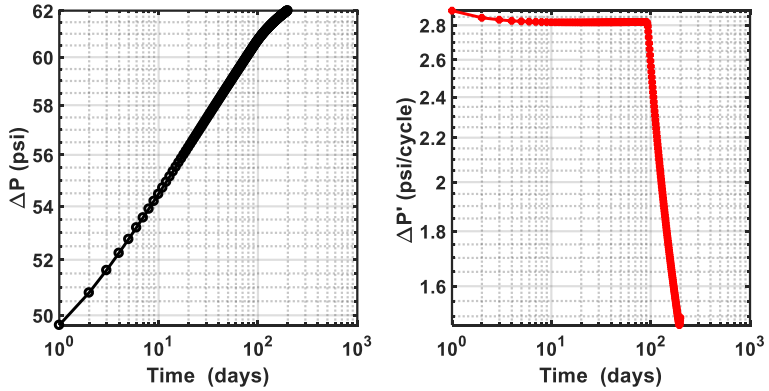


Figure 5-6. The diagnostic plot of pressure changes and Bourdet-type pressure derivative responses for 100 days with the permeability of 1 md and 5 md for case 1 in Table 5-4.

Table 5-5 is the actual and initial guess of inner and outer zone permeabilities of the dual-zone models. The seven cases include various of permeability changes from the initial guesses to the actual permeability. An example of the permeability map of the dual-zone model is shown in Figure 5-5. Figure 5-6 shows the pressure and pressure derivative of the dual-zone model #1. The inner zone permeability is 1md, and the outer zone permeability is 5md.

5.5 Results and Comparison

5.5.1 Application of the DQN Method to Automated History Matching

5.5.1.1 Single-Zone Models

7 single-zone models shown in Table 5-3 are tested using the DQN algorithm with reservoir and well parameters in Table 5-1 and Table 5-2. The 7 cases start with different initial permeability guesses and the goal is to estimate permeability that is close to the actual permeability values by matching the pressure and pressure derivative responses.

The prediction results are shown in Table 5-5. DQN use discrete actions (permeability multipliers) of [0.5, 0.7, 1, 1.3, 2], and thus can only predict a finite number of permeabilities within 10 iterations of 1 episode. For all the 7 cases, case 4 is the best. The predicted permeability equals the actual permeability exactly. In other cases, the errors are around 10 percent.

The error between predicted and the actual pressure history is evaluated using normalized root mean squared error (NRMSE), which is the normalized root mean square error using the mean of the actual pressure. The NRMSE is formally defined as:

$$\boldsymbol{NRMSE = RMSE/\bar{y} = (\Sigma(\hat{y} - y)^2/n)/\bar{y}} \quad (5\text{-}4)$$

where y is the actual pressure, $\bar{y}$ is the mean of the actual pressure, $\hat{y}$ is the predicted pressure, n is the number of the data point.

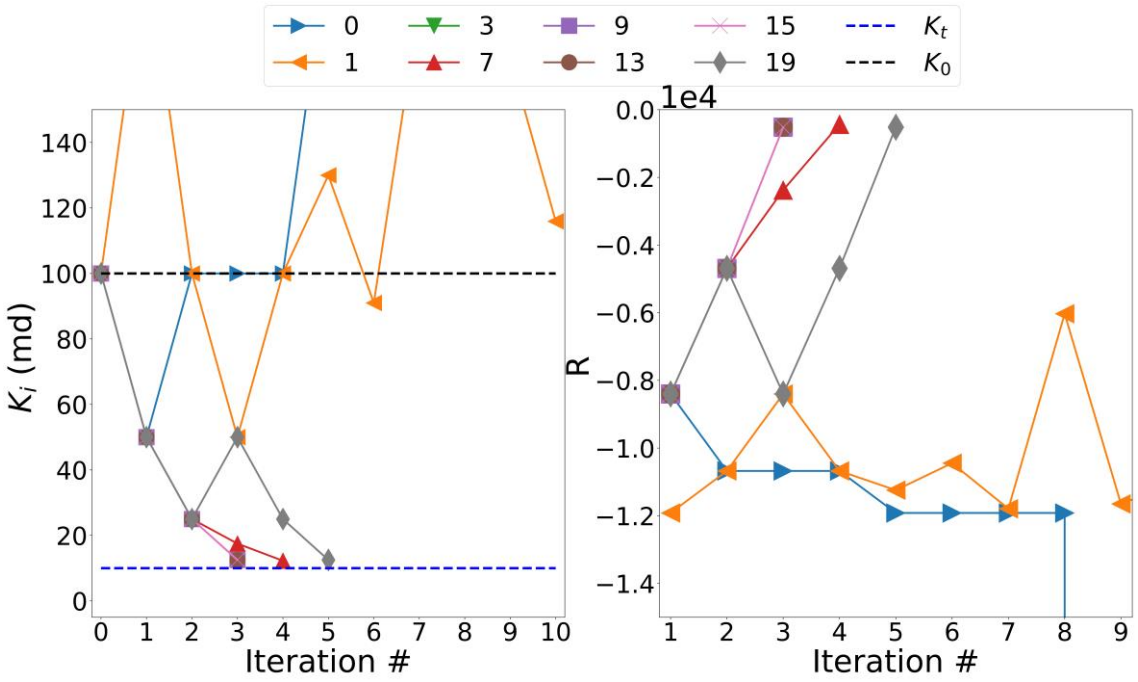
Table 5-5. Predicted permeability of the single-zone models.

Case number	Initial guessed permeability (md)	Actual permeability (md)	Predicted permeability (md)	NRMSE
1	100	1	1.07	0.0693

2	100	5	5.57	0.0968
3	100	10	11.37	0.1154
4	100	50	50	0
5	1	100	89.6	0.1093
6	1	50	44.8	0.1106
7	1	10	10.4	0.0371

Case 3 is the worst performing case in this example. Figure 5-7 shows the predicted permeability (Figure 5-7a) and corresponding reward (Figure 5-7b) for each episode and iteration of case 3. Each line represents an episode and each dot represent the predicted permeability (left) or the corresponding reward (right) for each iteration. K_t and K_0 represent true permeability and the initial guess of permeability.

The permeability is set to the initial permeability (100md) at the beginning of each episode. Due to the discrete actions, lines from different episodes may overlap with each other when the same actions are selected. At the beginning stage of training, the DQN randomly select actions to explore the action space. After several explorations, the DQN learns that by selecting the actions that decease the permeability it will gain a higher reward. In episodes 7-19 the algorithms show a constant performance, the DQN tries to lower the permeability to receive a higher reward. At the end of each episode, the DQN gets close enough to K_t, but due to the limitation of discrete actions, it only reaches an approximate solution.



(a) **(b)**

Figure 5-7. (a) Predicted permeability (k_i) and (b) reward (R) at each iteration of each episode for the history matching in case 3. The legend above the figure refers to a few selected episodes (0, 1, 3, 7, 9, 13, 15 and 19). Iterations within each episode are denoted on the x-axis.

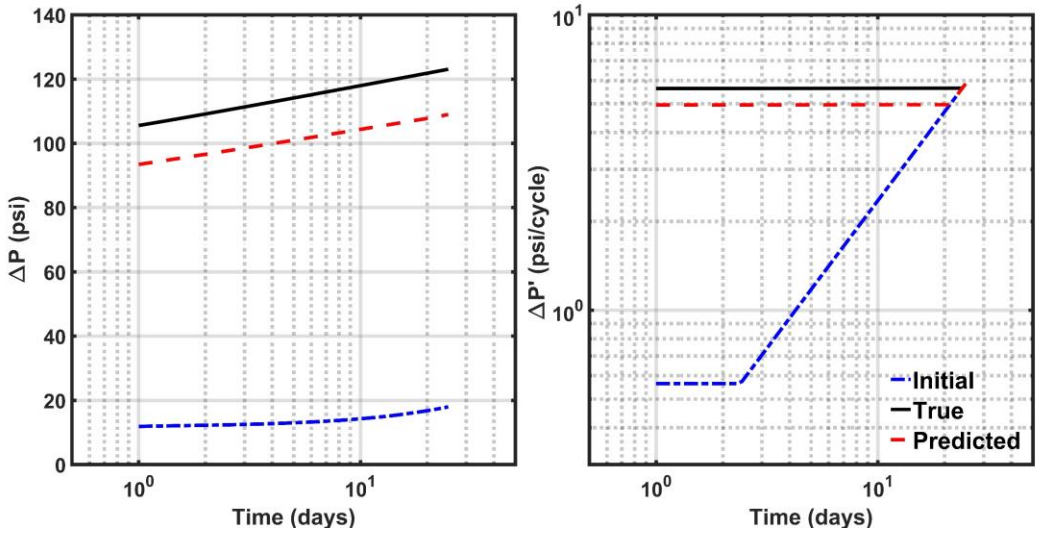


Figure 5-8. Comparison of the actual and predicted pressure (left) and pressure derivative (right) for case 3.

Figure 5-8 shows the result of case 3 which is the worst of the 7 cases studied. The blue line is the pressure and pressure derivative generated with initial guess

permeability, which is 100 for this case. The black line is the true pressure history to be matched. The red line is the predicted pressure and pressure derivative. Clearly the algorithm was unable to match production history.

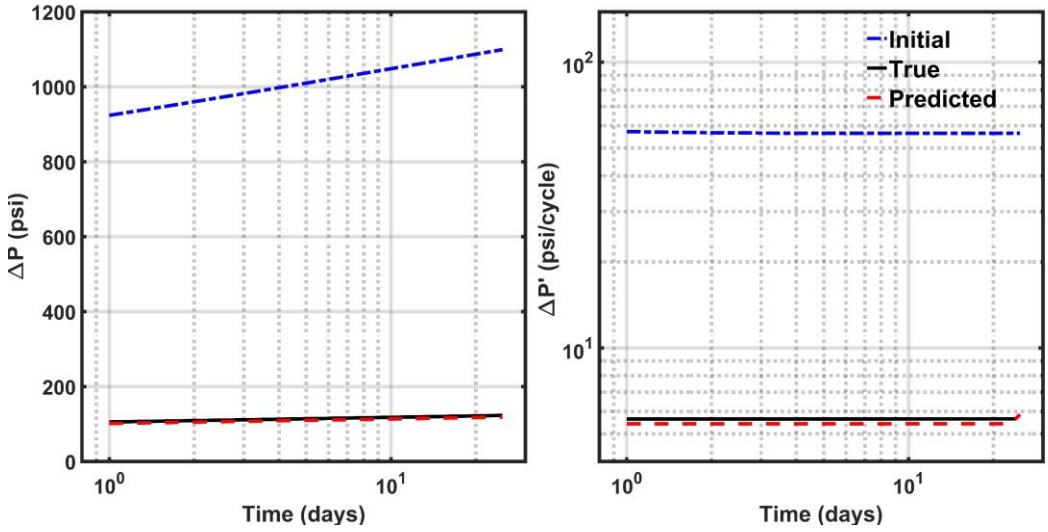


Figure 5-9. Comparison of the actual and predicted pressure (left) and pressure derivative (right) of case 7.

To give a comparison, Figure 5-9 shows the prediction result of case 7. Case 7 achieved an NRMSE of 0.0371. Compared with case 3, the predicted pressure and pressure derivative exhibit a great match with each other for case 7.

5.5.1.2 Dual-Zone Models

Due to the two permeabilities to be predicted, the action space for the learning task is larger compared to the single-zone models. The actions include a choice of permeability multiplier from a set of 3 values: 0.7, 1.0 and 1.3. So for the two concentric permeability rings, there are a total of 9 possible combinations of permeability multipliers. The prediction results are summarized in Table 5-6.

Table 5-6. Initial guess and actual permeabilities of the dual-zone models.

Case number	Initial guessed permeability (md)		Actual permeability (md)		Predicted permeability		NRMSE
	Inner	Outer	Inner	Outer	Inner	Outer	
1	100	100	1	5	2.8	33.6	0.6327
2	100	100	1	10	2.8	2.8	0.6247
3	100	100	1	50	2.8	24	0.6296
4	100	100	50	1	49	91	0.1830
5	50	50	100	1	76.8	168	0.8155
6	10	10	20	50	18.2	5.8	0.1195
7	1	1	5	20	4.8	2.6	0.0447

In general, the performance of RL appears to be worse in comparison to the simpler model previously discussed. This can be attributed to several factors. First, the chosen dimensions of the action space may be small and in Cases 2 and 3, the inner zone permeability decreases down to 2.8md. More iterations may have provided better convergence in this example. Another reason is the combined effect of the inner zone permeability and width on the predicted pressure values. For example, in Cases 4 and 7, the DQN predicts the inner zone permeability accurately, but because the misfit is less sensitive to the outer zone permeability, the final NRMSE is small.

5.5.2 Application of the DDPG Continuous Method to Automated History Matching

DDPG may solve the problems occurring in section 5.5.1. Continuous actions are used in DDPG, and thus the algorithm can explore an infinite number of permeabilities

by adjusting actions. The same cases as in section 5.5.1 are tested using the DDPG algorithm.

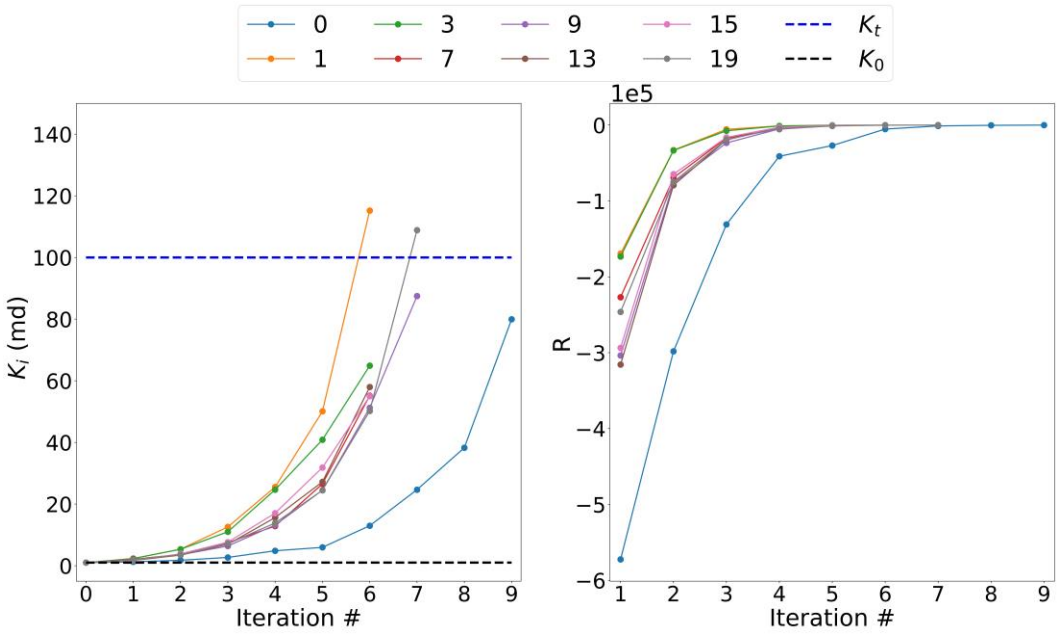
5.5.2.1 Single-zone Models

Table 5-7 summarizes the prediction performance of the DDPG algorithm. It shows great improvement compared with the DQN algorithm. All values of the NRMSE are below 0.05 as shown in the last column of Table 5-7. The DDPG algorithm uses continuous action to alter the permeability and thus can predict continuous variables instead of discrete points. (b)

Figure 5-10 shows the predicted permeability in each episode and the corresponding reward for case 5. Figure 5-11 shows the pressure history generated using the predicted permeability and actual permeability, respectively.

Table 5-7. Predicted permeability of single-zone cases.

Case number	Initial guessed permeability (md)	Actual permeability (md)	Predicted permeability (md)	NRMSE (DQN)	NRMSE (DDPG)	DDPG Improvement (%)
1	100	1	1.001	0.0693	0.0009	98.70
2	100	5	4.98	0.0968	0.0040	95.87
3	100	10	9.84	0.1154	0.0155	86.57
4	100	50	49.8	0	0.0041	-
5	1	100	96.9	0.1093	0.0304	72.19
6	1	50	48.52	0.1106	0.0286	74.14
7	1	10	9.955	0.0371	0.0041	88.95



(a) **(b)**

Figure 5-10. (a) Predicted permeability (k_i) and (b) reward (R) at each iteration of each episode for the history matching of case 5. Episodes are denoted using different colors and labels on top. Iterations are denoted on the x-axis.

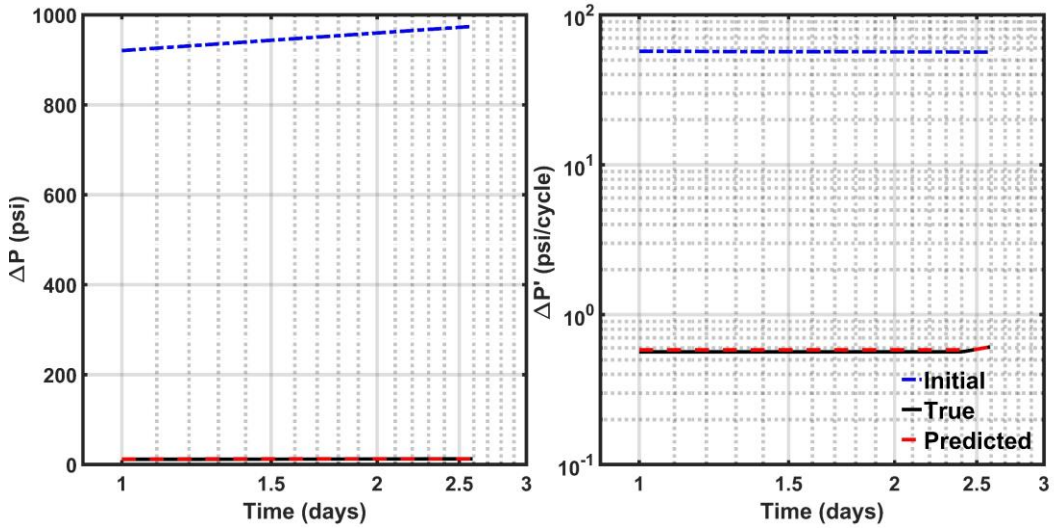


Figure 5-11. Comparison of the actual and predicted pressure (left) and pressure derivative (right) of case 5.

Compared to DQN, the performance of DDPG in single-zone models show a reduced misfit. As shown in Figure 5-11, even for the worst performing example, the predicted pressure response almost overlaps the true response. In (b)

Figure 5-10 we can interpret some learning processes of the DDPG algorithm. In episode 0, the DDPG algorithm makes a choice of increasing the permeability. After episode 0, the DDPG algorithm exhibits a consistent strategy of increasing the reward by increasing the permeability. At the end of episode 19, the predicted permeability is around 110, which is very close to the target permeability (100md).

5.5.2.2 Dual-zone models

Table 5-8. initial guess and actual permeabilities of the Dual-zone models.

Case number	Initial guessed permeability (md)		Actual permeability (md)		Predicted permeability		NRMSE (DQN)	NRMSE (DDPG)	DDPG Improvement (%)
	Inner	Outer	Inner	Outer	Inner	Outer			
1	100	100	1	5	1	0.34	0.6327	0.0064	98.99
2	100	100	1	10	0.976	0.56	0.6247	0.0267	95.73
3	100	100	1	50	1.02	0.56	0.6296	0.0180	97.14
4	100	100	50	1	37	65	0.1830	0.0697	61.91
5	50	50	100	1	97.6	0.9	0.8155	0.0346	95.76
6	10	10	20	50	20.1	8.7	0.1195	0.0038	96.82
7	1	1	5	20	5	0.007	0.0447	0.0138	69.13

The same 7 cases tested previously are tested using the DDPG algorithm and the results again show a much better performance in comparison to the DQN. Table 5-8 shows that the NRMSE is much smaller and the largest value is the NRMSE of case 4.

From the predicted permeability values, I observe that the DDPG predicts the inner permeability more accurately in comparison to the outer permeability. As explained earlier, the pressure response at the well is more sensitive to the inner permeability value and so this result makes sense.

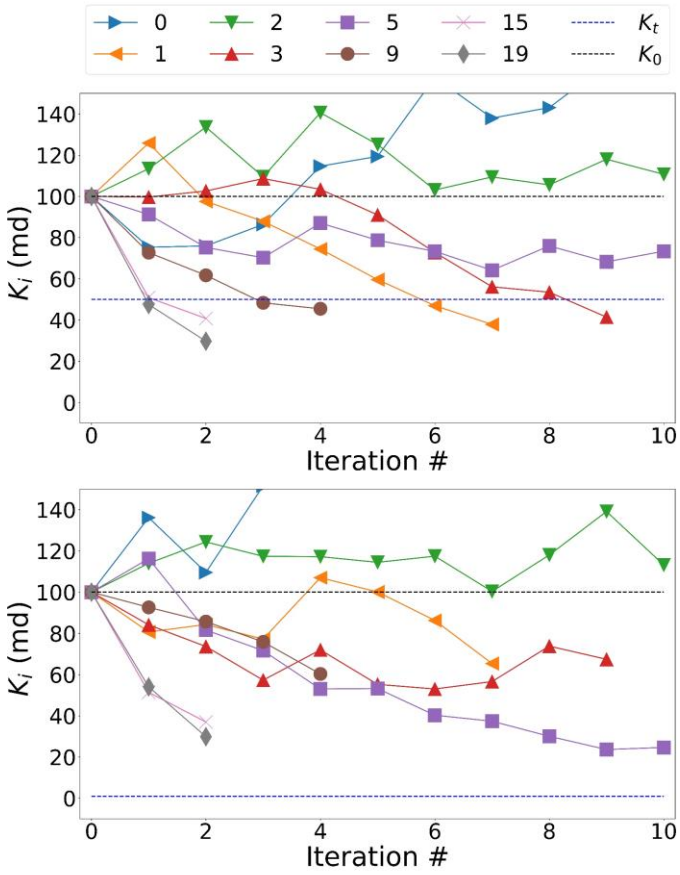


Figure 5-12. Predicted permeabilities at each iteration of 8 of case 4. The upper figure shows the search process with the target permeability of 50md; the lower figure with the target permeability of 1md.

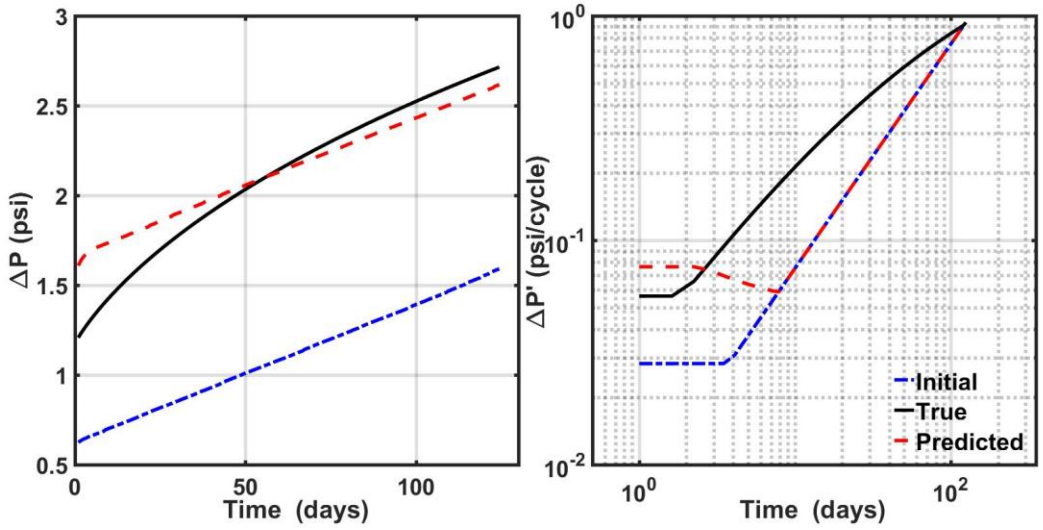


Figure 5-13. Comparison of the actual and predicted pressure on a linear scale (left) and pressure derivative (right) of case 4.

Figure 5-12 and Figure 5-13 show the result of case 4. Case 4 has the highest NRMSE in all the 7 test cases. Figure 5-12 shows how DDPG learns to take actions to reach the two target permeabilities. Initially, DDPG explores the action space randomly. The predicted permeability shows a random change in episodes 1-3. After episode 3, the algorithm gradually learns that reducing both permeabilities will result in a higher reward. In episode 15 and 19, the DDPG algorithm reaches the target within two steps. Episodes 15 and 19 stop at iteration 2 where the misfit is lower than a threshold.

Figure 5-13 is the predicted and actual pressure and pressure derivative. The true inner permeability is 50md and the pressure front hit the boundary of the inner zone at about 2 days. The permeability of the inner zone is high compared to cases 1-4, and thus the pressure drop is small. The predicted pressure profile has a poor match in the first half and has a good match in the second half. Although the DDPG missed the outer zone permeability entirely, it still has a better match in the second half. This is largely due to the limitation of the reward function. Our reward function is designed to be a negative

mismatch of the pressure. The DDPG relies only on the misfit between the predicted and actual pressure to evaluate the goodness of the prediction. However, from the derivative plot, we can see that that the pressure front with the predicted permeabilities hit the reservoir boundary in about 7 days, and there is a large misfit between the predicted and actual pressure derivative after day 7. The reward function would perform better if it combines misfit from both pressure and pressure derivative.

Figure 5-14 shows the best-performed case. Case 6 has an inner permeability of 20md and outer permeability of 50md. From the pressure derivative plot, we can see that the pressure front hit the outer boundary of the inner zone at day 5.5, and hit the boundary of the reservoir at day 6. The outer zone has higher permeability and thus pressure front goes faster through the outer zone. Even though the outer zone permeability is not accurately predicted, the resulting pressure profiles still can match with good accuracy.

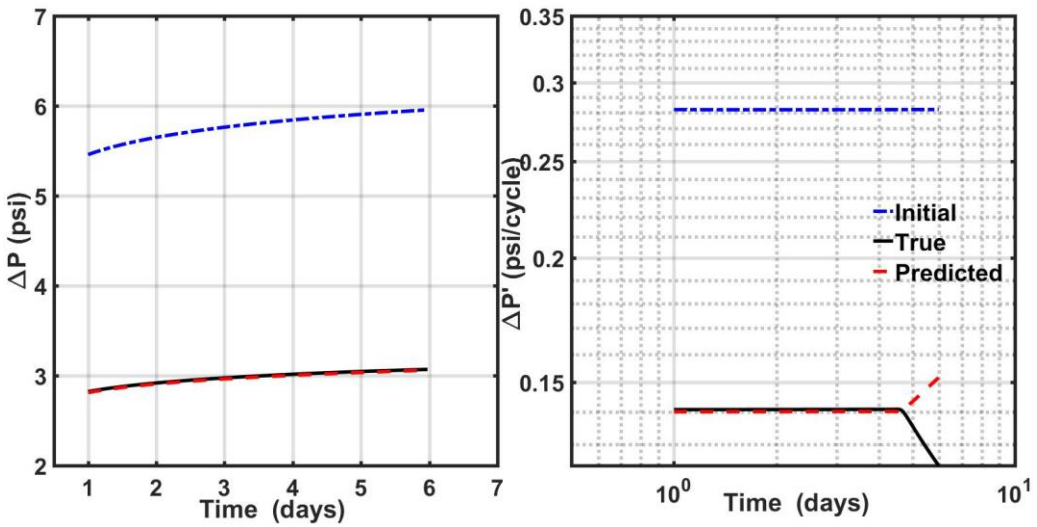


Figure 5-14. Comparison of the actual and predicted pressure on a linear scale (left) and pressure derivative (right) of case 6.

I have tried to combine the misfit from pressure and pressure derivative into the reward function. However, several obstacles result in worse performance. First, as shown in Figure 5-13, the absolute value is usually several orders smaller than the pressure drop. The right method should be selected to normalize the two misfits on the same scale. Second, how to combine the two misfits needs more study. Our study shows simple normalization and a weighted sum of the two sums results in deteriorated prediction performance.

5.6 DDPG Model on Reservoir Permeability Map Generated by Exponential Covariance Function

In this section, more realistic permeability maps are generated with an exponential spatial covariance function.

5.6.1.1 Permeability Maps Generated by the Exponential Covariance Function

One disadvantage of the simulations in the previous sections is the simplicity of the reservoir permeability maps. In a real reservoir, the permeability and other parameters of the reservoir are usually correlated in space. In this section, I define a square reservoir on a 41x41 grid. Each grid is 100ft by 100ft. The reservoir parameters of porosity, oil composition, etc. are the same as the oil reservoir introduced in the previous section. The permeability maps are generated using the exponential covariance function expressed as:

$$\boldsymbol{C_{ij}} = \boldsymbol{C_0 exp}\left(-\frac{\boldsymbol{h_{ij}}^2}{\boldsymbol{a_0^2}}\right) \quad (5\text{-}5)$$

where *i* and *j* represents the grid index, $\boldsymbol{C_{ij}}$ represents the covariance between the grid *i* and grid j, $\boldsymbol{h_{ij}}$ represents the distance between the grid i and grid j. $\boldsymbol{C_0}$ and a_0 are scaling factors.

However, the covariance matrix generated by Equation

(5-5) is semi-positive definite and is transformed to the closest positive definite matrix. For our experiment, the error caused by transforming the covariance matrix to the closest positive definite matrix is small. The covariance matrix $\boldsymbol{C}$ is decomposed as:

$$\boldsymbol{C} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^{-1} \tag{5-6}$$

where $\boldsymbol{\Lambda}$ is the diagonal matrix contains eigenvalues, and $\boldsymbol{Q}$ is the corresponding matrix contains eigenvectors. To convert the covariance matrix $\boldsymbol{C}$ to the nearest positive definite matrix, the matrix $\boldsymbol{\Lambda}$ is processed to remove the negative eigenvalues.

$$\boldsymbol{\Lambda}' = \boldsymbol{diag}(\boldsymbol{\lambda}) = \begin{cases} \boldsymbol{\lambda_i}, & \boldsymbol{\lambda_i} \geq \boldsymbol{0} \\ \boldsymbol{\varepsilon}, & \boldsymbol{\lambda_i} < \boldsymbol{0} \end{cases} \tag{5-7}$$

where $\boldsymbol{\lambda_i}$ is the eigenvalues in the matrix $\boldsymbol{\Lambda}$, $\boldsymbol{\varepsilon}$ is a small positive number.

The transformed eigenvalue matrix $\boldsymbol{\Lambda}'$ is then used to generate the new covariance matrix $\overline{\boldsymbol{C}}$:

$$\overline{\boldsymbol{C}} = \boldsymbol{Q}\boldsymbol{\Lambda}'\boldsymbol{Q}^{-1} \tag{5-8}$$

The new covariance matrix $\overline{\boldsymbol{C}}$ is guaranteed to be positive definite. The covariance matrix $\overline{\boldsymbol{C}}$ is then decomposed with the Cholesky decomposition:

$$\overline{\boldsymbol{C}} = \boldsymbol{\Sigma}\boldsymbol{\Sigma}^{\boldsymbol{T}} \tag{5-9}$$

The permeability map is generated with the matrix $\boldsymbol{\Sigma}$ and Gaussian distribution:

$$\boldsymbol{X} = \overline{\boldsymbol{X}} + \boldsymbol{\Sigma}\boldsymbol{N} \tag{5-10}$$

where $\overline{\boldsymbol{X}}$ a vector with size 1681 that controls the mean of the permeability map, $\boldsymbol{\Sigma}$ is a matrix of 1681 by 1681, $\boldsymbol{N}$ is a vector with size 1681 that generated from Gaussian distribution $\mathbb{N}(0,1)$. The matrix $\boldsymbol{X}$ is then transformed into a 41x41 matrix, which is the permeability map. An example of the actual permeability map generated with the aforementioned process is shown in Figure 5-15. The permeability map generated with

the exponential covariance function is more realistic than the permeability map used in the previous section.

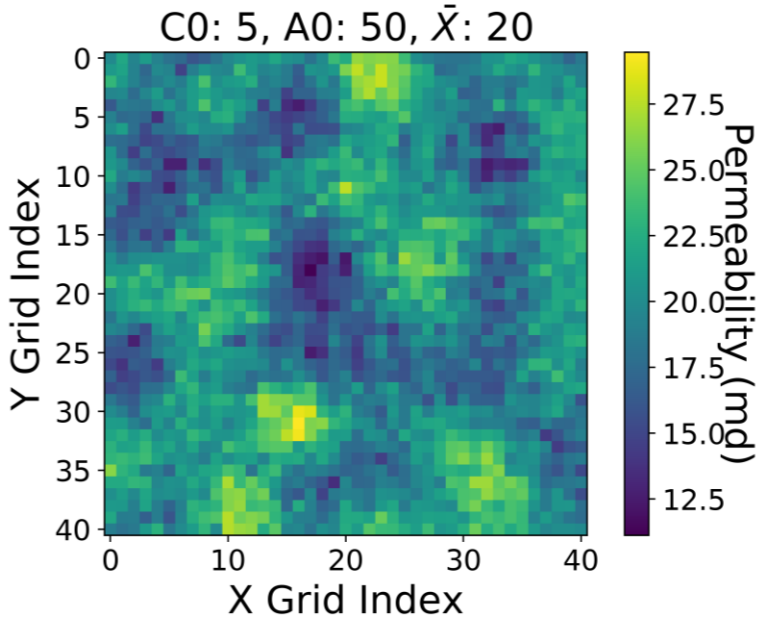


Figure 5-15. Permeability map generated with the actual parameters mentioned in Table 5-9.

With this method, the permeability map is controlled by the covariance matrix. The three key parameters controlling the permeability distribution are the scaling factors $\boldsymbol{C_0}$, a_0, and the mean permeability $\overline{\boldsymbol{X}}$. The goal of the DDPG algorithm applied in this section is to find the correct combination of the three parameters that reproduces production history. The true parameters and the initial guess of the parameters are shown in Table 5-9.

Table 5-9. Initial guess, actual, and predicted values of parameters of the exponential covariance function.

	$\boldsymbol{C_0}$	a_0	$\bar{\boldsymbol{X}}$
Initial guessed parameters	1	200	10
Actual parameters	5	50	20
Predicted parameters	1.213	57.741	20.379

The initial guess and the actual parameters are selected to ensure that the permeability maps generated are positive, and the pressure wave takes long enough to reach the reservoir boundary. The DDPG algorithm is applied to estimate the parameters of the connectivity function.

5.6.1.2 Results

Table 5-9 listed the initial guess, actual, and predicted values of the three parameters of connectivity function. During the experiment, I find that $\bar{X}$ is more important than the other two parameters, $\boldsymbol{C_0}$ and a_0 which control the variance of the permeability map. However, the effect of the permeability variance on the production history is much smaller than the mean of the permeability. The mean of the permeability directly affects the pressure wave transition time, whereas the variance of the permeability affects local pressure wave transition. The DDPG algorithm learns to minimize the prediction error by predicting the $\bar{X}$ with high accuracy, but the algorithm neglected the other two parameters as you can see in Table 5-9 and Figure 5-16. From

Table 5-9 we can see that there is a large error between the predicted $\boldsymbol{C_0}$, a_0 and the actual $\boldsymbol{C_0}$, a_0. By changing the $\bar{X}$ alone, the algorithm obtains a good reward.

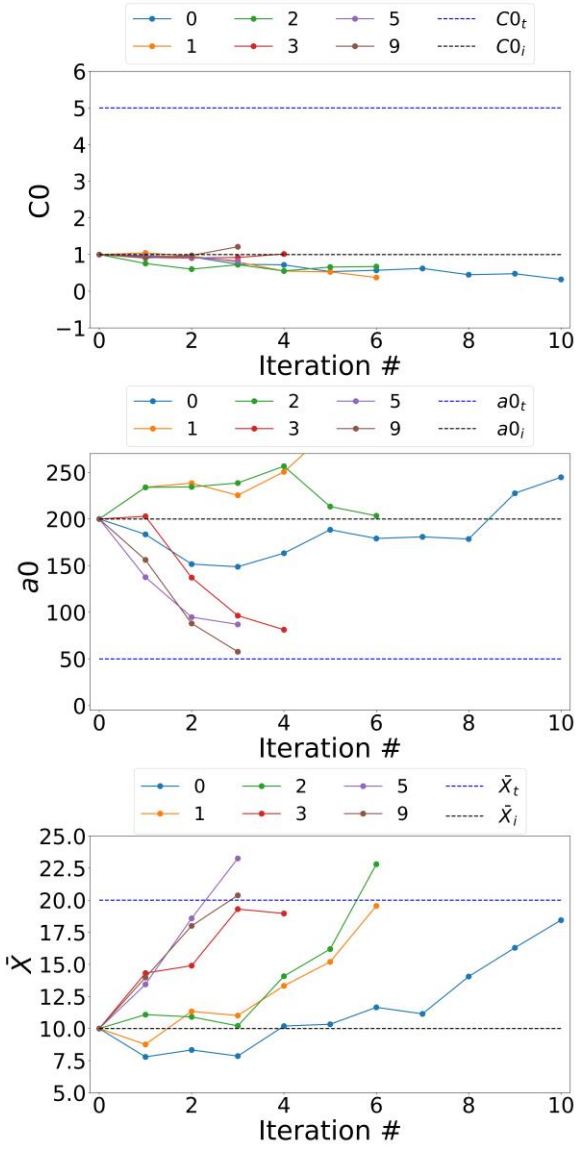


Figure 5-16. (a) Predicted $\boldsymbol{C_0}$ (b) $\boldsymbol{a_0}$ and (b) $\bar{X}$ at each iteration of each episodes for the history matching. Episodes are denoted using different colors and labels on top. Iterations are denoted on the x-axis.

Figure 5-16 shows an example of the DDPG predicted three parameters during the first 10 episodes. From Figure 5-16 we can see that DDPG seems to neglect the parameter $\boldsymbol{C_0}$. The $\boldsymbol{C_0}$ estimates stay constant during the first 10 episodes. For the other

two parameters, DDPG finds the correct strategy. The DDPG algorithm learned to decrease parameter a_0 after several episodes. The predicted a_0 almost equals the actual a_0 at the last step of the 9th episode. The DDPG algorithm increases the parameter $\bar{X}$ from the initial guess (10) to the actual value (20) after several episodes. The DDPG algorithm reaches the actual value of the parameter within fewer steps with the increase of episodes. This indicates the DDPG algorithm learns the correct strategy for parameter a_0 and $\bar{X}$. The parameter $\boldsymbol{C_0}$ since it does not have large impact on the production history.

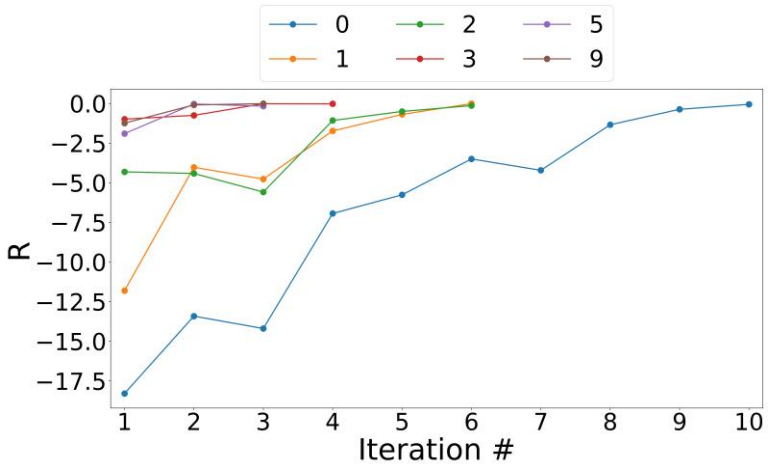


Figure 5-17. Rewards of each episode of the DDPG algorithm.

Figure 5-17 shows the rewards the DDPG algorithm gets at each step during 6 episodes. The reward is the negative of the error between the actual production history and the predicted production history. We can see that in episode 0, the DDPG algorithm takes 10 steps to increase the reward. However, the DDPG algorithm improves after each episode. It takes the algorithm 6 steps before the algorithm reaches the target for episodes 1 and 2. For episodes 5 and 9, the algorithm only takes 3 steps to reach the target. The reward almost equals 0 (which is the largest reward possible) when the episodes terminate.

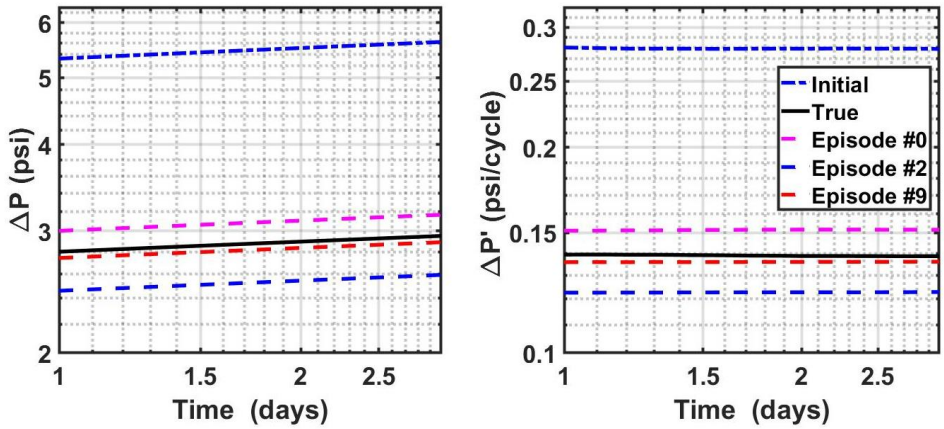
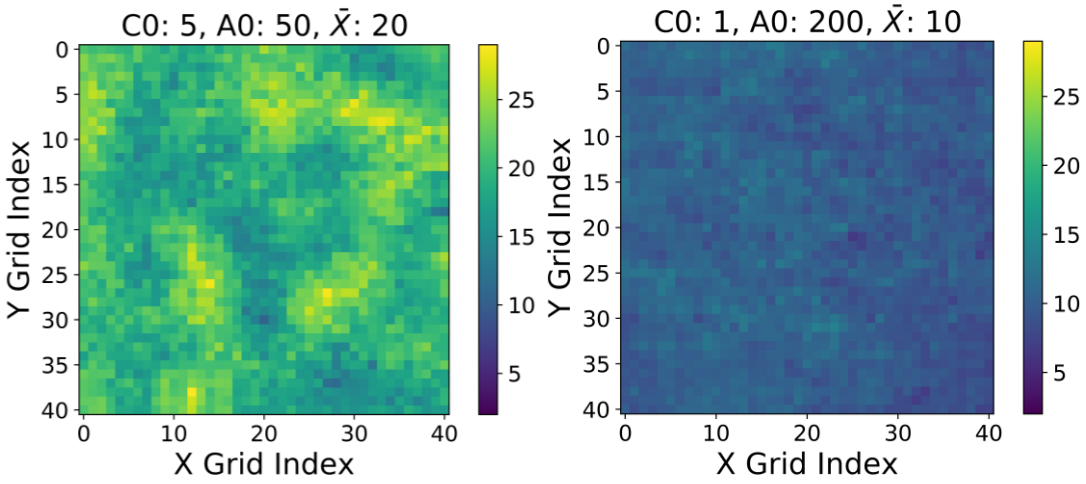
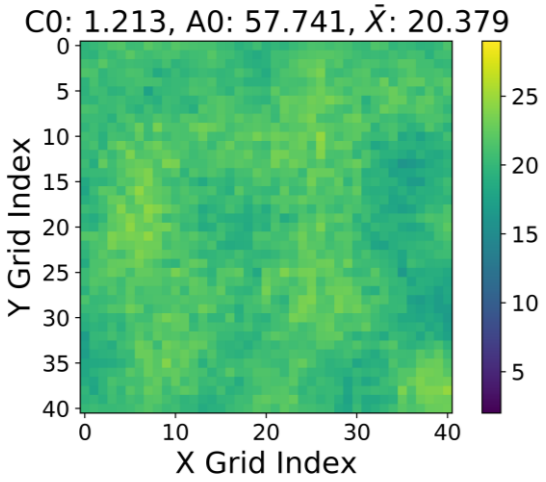


Figure 5-18. Comparison of the actual and predicted pressure (left) and pressure derivative (right).

Figure 5-18 shows the pressure and pressure derivative for the actual, initial guessed, and DDPG predicted reservoirs. The DDPG predicted results of episodes #0, #2, and #9 are plotted for comparison. The blue dashed line is the pressure and pressure derivative from the initial guessed reservoir. The solid black line is the pressure and pressure derivative from the actual (or target) reservoir. The goal of the DDPG algorithm is to find the actual reservoir permeability map from the initial guessed reservoir permeability map. From Figure 5-18 we can see that the pressure and pressure derivative of the predicted and the actual reservoir are quite close. This indicates that the DDPG algorithm manages to find a strategy to minimize the difference (or error) between the predicted and actual pressure history.



(a). Actual (b) Initial



(c). Predicted

Figure 5-19. Permeability map generated using the (a) actual parameters, (b) initial guess parameters, and (c) DDPG predicted parameters of the connectivity function as listed in Table 5-9.

Figure 5-19 shows three permeability map realizations generated using the actual, initial guessed, and DDPG predicted reservoir parameters. The three parameters control the average permeability and the spatial variation of the permeability. From the figure, we can see that the permeability map generated with the initial parameter values are quite different from the actual permeability map. Even though the DDPG algorithm fails to

predict the C_0, the permeability map generated with the three DDPG predicted parameters is still similar to the actual permeability map. The reward acquired by the DDPG algorithm increases at each iteration. This indicates that the DDPG algorithm learns to neglect the parameter of C_0. By focusing on the other two parameters, DDPG managed to generate permeability maps that are close to the target permeability map.

5.7 Assumptions and Limitations

This study is based on the following assumptions. The reservoir is simulated based on a simple reservoir simulator. The simulator uses the fast-marching algorithm to increase the simulation process. A fictitious isotropic single-layer circular reservoir is built with a dead oil model to simplify the problem and reduce the parameter to be matched. I assume all parameters of the reservoir are known except permeability and no fractures exist. The RL environment is modeled as a deterministic MDP.

The proposed automated history matching process has limitations in the following aspects. First, the reservoir model is simple. Further experiments are needed to test a more complex reservoir model on the commercial simulator. Second, a single objective function is used to control learning. However, the error calculated by the objective function between predicted and true production history varies with the reservoir permeability. Different objective functions need to be tested to find the most suitable one for history matching problems. Third, the program is still primitive, and some functions (hyperparameters selection etc.) need to be performed manually. Third, for the DDPG algorithm, some of the parameters need to be adjusted to achieve the best performance. DDPG algorithm includes complex processes of initialization, exploration, weight update, etc. DDPG needs some experimental simulation to adjust the parameters for

different problems. Lastly, the RL algorithms can only predict a single set of parameters. The solution of the history matching problem is non-unique. However, the RL can only predict one unique set of values. Instead of finding the parameters, the RL algorithms are actually trying to find a policy, or how to change the current values of the parameters. A possible solution to solve the non-uniqueness of the history matching with RL algorithm is to use the value function acquired to evaluate the value of each state (parameters), and instead of selecting the best set of parameters, we can select a range of parameters. More research is needed for this limitation.

5.8 Conclusions

In this study, two reinforcement learning algorithms are applied to achieve automatic history matching. A reservoir simulator using the MFM algorithm is encapsulated in an interactable environment. The reinforcement learning algorithms use the sequential decision-making process to adjust the initial reservoir permeability to approach the target reservoir permeability. Principles of existing history matching methods and the proposed history matching method using reinforcement learning algorithms are explained and compared in detail.

Reservoir permeability is the target variable to be predicted by the RL algorithms. 14 different reservoir models are developed to test the performance of both DQN and DDPG algorithms. 7 cases with a single-zone and 7 cases with 2 targets are used to test both algorithms. Both algorithms can find a strategy to decrease the misfit for all cases. However, due to the limitation of the discrete action, DQN can not match the production history with great accuracy, especially for dual-zone models. Whereas DDPG can achieve relatively higher accuracy with NRMSE less than 0.07 for all test cases. We

observe a minimum of 60% improvement of NRMSE of the DDPG algorithm compared to the DQN algorithm. The DDPG algorithm is also tested on reservoirs with more realistic permeability maps. The permeability maps are generated using the exponential covariance function. The results indicate that the DDPG algorithm successfully reduced the misfit between the predicted reservoir production history and the target reservoir production history. However, DDPG fails to predict the $\boldsymbol{C_0}$ parameter.

The reinforcement learning algorithm provides a new solution for the existing history matching problem. Our study shows the promising result of using deep reinforcement learning algorithms to achieve automatic history matching. This study proposed a new history matching process. Further studies are needed to optimize the parameters in the deep networks and improve the reward function.

Chapter 6 Conclusions

In this study, various shallow- and deep-machine learning methods are applied and evaluated on different characterization tasks related to geological materials. The tasks include synthetic logs generation for pore-scale characterization and geo-mechanical characterization, core-scale characterization of crack network embedded in materials, and automatic history matching for reservoir-scale permeability characterization. The following conclusions can be drawn from these investigations.

Supervised-learning using MARS model and shallow neural networks exhibit good sonic log-synthesis performance. In blind test on a separate well, MARS model achieves an R2 of 0.63 and 0.59 when synthesizing DTC and DTS logs, respectively.

Deep learning models can be used to generate the NMR T2 distribution responses of the near-wellbore region by processing easy-to-acquire well logs under data constraints with high accuracy and good smoothness.

Shallow models require less computation to generate the NMR T2 log, K-neighbors perform best. OLS, LASSO, ElasticNet models have similar performance. NMR T2 generated by the ANN model is not as smooth as real NMR T2.

A source-sensor experiment configuration is developed for the purpose of crack-bearing material characterization based on the measurement of sonic wavefront measurements. The results indicate that the source-sensor experiment configuration can provide enough information to classify crack-bearing models with different fracture orientations and locations.

The experiments show that machine learning models exhibit high classification accuracy on classifying models with different orientations based on the wavefront

measurements. For experiments with 4 and 8 different orientations, the voting classifier achieves an accuracy of 0.99 and 0.89.

In the study of automatic history matching, both DQN and DDPG algorithms can find a strategy to decrease the misfit for all cases. However, due to the limitation of the discrete action, DQN can not match the production history with great accuracy, especially for dual-zone models. Whereas DDPG can achieve relatively higher accuracy with NRMSE less than 0.07 for all test cases.

References

Abdi, H. (2010). Partial least squares regression and projection on latent structure regression (PLS Regression). *Wiley interdisciplinary reviews: computational statistics, 2*(1), 97-106.

Aggelis, D. G. (2011). Classification of cracking mode in concrete by acoustic emission parameters. *Mechanics Research Communications, 38*(3), 153-157.

Aggelis, D. G., Shiotani, T., & Terazawa, M. (2009). Assessment of construction joint effect in full-scale concrete beams by acoustic emission activity. *Journal of engineering mechanics, 136*(7), 906-912.

Akinnikawe, O., Lyne, S., & Roberts, J. (2018). *Synthetic Well Log Generation Using Machine Learning Techniques*. Paper presented at the SPE/AAPG/SEG Unconventional Resources Technology Conference, Houston, Texas, USA. https://doi.org/10.15530/URTEC-2018-2877021

Alexeyev, A., Ostadhassan, M., Mohammed, R. A., Bubach, B., Khatibi, S., Li, C., & Kong, L. (2017). *Well Log Based Geomechanical and Petrophysical Analysis of the Bakken Formation*. Paper presented at the 51st U.S. Rock Mechanics/Geomechanics Symposium, San Francisco, California, USA. https://doi.org/

Asoodeh, M., & Bagheripour, P. (2012). Prediction of compressional, shear, and stoneley wave velocities from conventional well log data using a committee machine with intelligent systems. *Rock Mechanics and Rock Engineering, 45*(1), 45-63.

Avansi, G. D., & Schiozer, D. J. (2015). *A New Approach to History Matching Using Reservoir Characterization and Reservoir Simulation Integrated Studies*. Paper presented at the Offshore Technology Conference, Houston, Texas, USA.

Baines, V., Bootle, R., Pritchard, T., Macintyre, H., & Lovell, M. (2008). *Predicting Shear And Compressional Velocities In Thin Beds.* Paper presented at the 49th Annual Logging Symposium.

Baron, L., & Holliger, K. (2011). Constraints on the permeability structure of alluvial aquifers from the poro-elastic inversion of multifrequency P-wave sonic velocity logs. *IEEE Transactions on Geoscience and Remote Sensing, 49*(6), 1937-1948.

Beale, M. H., Hagan, M. T., & Demuth, H. B. (2012). *Neural network toolbox™ user's guide.* Paper presented at the R2012a, The MathWorks, Inc., 3 Apple Hill Drive Natick, MA 01760-2098,, www. mathworks. com.

Bhoumick, P. (2018). Mapping Hydraulic Fracture Using Shear Wave.

Bhoumick, P., Sondergeld, C., & Rai, C. S. (2018). *Mapping Hydraulic Fracture in Pyrophyllite Using Shear Wave*. Paper presented at the 52nd U.S. Rock Mechanics/Geomechanics Symposium, Seattle, Washington. https://doi.org/

Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., & Bengio, S. (2015). Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*.

Cai, Y., Liu, D., Mathews, J. P., Pan, Z., Elsworth, D., Yao, Y., . . . Guo, X. (2014). Permeability evolution in fractured coal — Combining triaxial confinement with X-ray computed tomography, acoustic emission and ultrasonic techniques. *International Journal of Coal Geology, 122*, 91-104. doi:https://doi.org/10.1016/j.coal.2013.12.012

Cancelliere, M., Verga, F., & Viberti, D. (2011). *Benefits and Limitations of Assisted History Matching*. Paper presented at the Offshore Europe, Aberdeen, UK.

Canchumuni, S. A., Emerick, A. A., & Pacheco, M. A. (2017). *Integration of Ensemble Data Assimilation and Deep Learning for History Matching Facies Models*. Paper presented at the OTC Brasil, Rio de Janeiro, Brazil.

Castagna, J. P., Batzle, M. L., & Eastwood, R. L. (1985). Relationships between compressional-wave and shear-wave velocities in clastic silicate rocks. *Geophysics, 50*(4), 571-581. doi:10.1190/1.1441933

Chang, H.-C., Chen, H.-C., & Fang, J.-H. (1997). Lithology determination from well logs with fuzzy associative memory neural network. *IEEE Transactions on Geoscience and Remote Sensing, 35*(3), 773-780.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Chumney, E. C. (2006). *Methods and designs for outcomes research*: ASHP.

Ding, S., Jiang, H., Liu, G., Sun, L., Lu, X. a., & Zhao, L. (2016). Determining the levels and parameters of thief zone based on automatic history matching and fuzzy method. *Journal of Petroleum Science and Engineering, 138*, 138-152. doi:https://doi.org/10.1016/j.petrol.2015.09.010

Doersch, C. (2016). Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*.

Dosovitskiy, A., & Brox, T. (2016). *Generating images with perceptual similarity metrics based on deep networks.* Paper presented at the Advances in Neural Information Processing Systems.

Eberhart-Phillips, D., Han, D.-H., & Zoback, M. D. (1989). Empirical relationships among seismic velocity, effective pressure, porosity, and clay content in sandstone. *Geophysics, 54*(1), 82-89. doi:10.1190/1.1442580

Elkatatny, S. M., Zeeshan, T., Mahmoud, M., Abdulazeez, A., & Mohamed, I. M. (2016). *Application of Artificial Intelligent Techniques to Determine Sonic Time from Well Logs*. Paper presented at the 50th U.S. Rock Mechanics/Geomechanics Symposium, Houston, Texas.

Elsakout, D., Christie, M., & Lord, G. (2015). *Multilevel Markov Chain Monte Carlo (MLMCMC) For Uncertainty Quantification*. Paper presented at the SPE North Africa Technical Conference and Exhibition, Cairo, Egypt. https://doi.org/10.2118/175870-MS

Emerick, A. (2017). Technology Focus: History Matching and Forecasting (April 2017). doi:10.2118/0417-0089-JPT

Emerick, A. A., & Reynolds, A. C. (2011). *Combining the Ensemble Kalman Filter with Markov Chain Monte Carlo for Improved History Matching and Uncertainty Characterization*. Paper presented at the SPE Reservoir Simulation Symposium, The Woodlands, Texas, USA. https://doi.org/10.2118/141336-MS

Fadakar Alghalandis, Y. (2014). *Stochastic modelling of fractures in rock masses.*

Farhidzadeh, A., Mpalaskas, A. C., Matikas, T. E., Farhidzadeh, H., & Aggelis, D. G. (2014). Fracture mode identification in cementitious materials using supervised pattern recognition of acoustic emission features. *Construction and Building Materials, 67*, 129-138. doi:https://doi.org/10.1016/j.conbuildmat.2014.05.015

Freedman, R. (2006). Advances in NMR Logging. doi:10.2118/89177-JPT
Friedman, J. H. (1991). Multivariate adaptive regression splines. *The annals of statistics*, 1-67.
Furtney, J. (2015). Scikit-fmm software. *https://github.com/scikit-fmm*.
Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*: MIT Press.
Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., . . . Bengio, Y. (2014). *Generative adversarial nets.* Paper presented at the Advances in neural information processing systems.
Greenberg, M., & Castagna, J. (1992a). SHEAR-WAVE VELOCITY ESTIMATION IN POROUS ROCKS: THEORETICAL FORMULATION, PRELIMINARY VERIFICATION AND APPLICATIONS. *Geophysical Prospecting, 40*(2), 195-209.
Greenberg, M. L., & Castagna, J. P. (1992b). SHEAR-WAVE VELOCITY ESTIMATION IN POROUS ROCKS: THEORETICAL FORMULATION, PRELIMINARY VERIFICATION AND APPLICATIONS1. *Geophysical Prospecting, 40*(2), 195-209. doi:10.1111/j.1365-2478.1992.tb00371.x
Gui, G., Pan, H., Lin, Z., Li, Y., & Yuan, Z. (2017). Data-driven support vector machine with optimization techniques for structural health monitoring and damage detection. *KSCE Journal of Civil Engineering, 21*(2), 523-534. doi:10.1007/s12205-017-1518-5
Guo, C., & Liu, R. C. (2010). A borehole imaging method using electromagnetic short pulse in oil-based mud. *IEEE Geoscience and Remote Sensing Letters, 7*(4), 856-860.
Han, D.-H. (1987). *Effects of porosity and clay content on acoustic properties of sandstones and unconsolidated sediments*. Retrieved from
Han, D. h., Nur, A., & Morgan, D. (1986). Effects of porosity and clay content on wave velocities in sandstones. *Geophysics, 51*(11), 2093-2107. doi:10.1190/1.1442062
Hassouna, M. S., & Farag, A. A. (2007). Multistencils fast marching methods: A highly accurate solution to the eikonal equation on cartesian domains. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 29*(9).
He, M., Miao, J., & Feng, J. (2010). Rock burst process of limestone and its acoustic emission characteristics under true-triaxial unloading conditions. *International Journal of Rock Mechanics and Mining Sciences, 47*(2), 286-298.
Hong, D., Yang, S., & Yang, S. (2014). A Separately Determining Anisotropic Formation Parameter Method for Triaxial Induction Data. *IEEE Geoscience and Remote Sensing Letters, 11*(5), 1015-1018.
Hossain, Z., Mukerji, T., & Fabricius, I. L. (2012). Vp-Vs relationship and amplitude variation with offset modeling of glauconitic greensand. *Geophysical Prospecting, 60*(1), 117-137.
Huang, K.-Y., Shen, L.-C., Chen, K.-J., & Huang, M.-C. (2013). *Multilayer perceptron with genetic algorithm for well log data inversion.* Paper presented at the Geoscience and Remote Sensing Symposium (IGARSS), 2013 IEEE International.
Illian, J., Penttinen, A., Stoyan, H., & Stoyan, D. (2008). *Statistical analysis and modelling of spatial point patterns* (Vol. 70): John Wiley & Sons.

Iverson, W. P., & Walker, J. N. (1988). Shear and compressional logs derived from nuclear logs *SEG Technical Program Expanded Abstracts 1988* (pp. 111-113): Society of Exploration Geophysicists.

Jingrong, Z., Ke, W., & Yang, G. (2010, 16-18 April 2010). *Acoustic emission signals classification based on support vector machine.* Paper presented at the 2010 2nd International Conference on Computer Engineering and Technology.

Johnston, J. E., & Christensen, N. I. (1993). Compressional to shear velocity ratios in sedimentary rocks. *International Journal of Rock Mechanics and Mining Sciences & Geomechanics Abstracts, 30*(7), 751-754. doi:https://doi.org/10.1016/0148-9062(93)90018-9

Karlsen, K. H., Lie, K. A., & Risebro, N. H. (2000). A fast marching method for reservoir simulation. *Computational Geosciences, 4*(2), 185-206. doi:10.1023/A:1011564017218

Kazemi, A., & Stephen, K. D. (2012). Schemes for automatic history matching of reservoir modeling: A case of Nelson oilfield in UK. *Petroleum Exploration and Development, 39*(3), 349-361.

Keys, R. G., & Xu, S. (2002). An approximation for the Xu-White velocity model. *Geophysics, 67*(5), 1406-1414.

Khaninezhad, M., & Jafarpour, B. (2014). Hybrid parameterization for robust history matching. *SPE Journal, 19*(03), 487-499.

Kim, S.-J., Koh, K., Lustig, M., Boyd, S., & Gorinevsky, D. (2007). An Interior-Point Method for Large-Scale $\ell_1 $-Regularized Least Squares. *IEEE journal of selected topics in signal processing, 1*(4), 606-617.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Korjani, M. M., Popa, A. S., Grijalva, E., Cassidy, S., & Ershaghi, I. (2016). *Reservoir Characterization Using Fuzzy Kriging and Deep Learning Neural Networks*. Paper presented at the SPE Annual Technical Conference and Exhibition, Dubai, UAE.

Leem, J., Lee, K., Kang, J. M., Park, Y., & Park, J. (2015). *History Matching with Ensemble Kalman Filter Using Fast Marching Method in Shale Gas Reservoir*. Paper presented at the SPE/IATMI Asia Pacific Oil & Gas Conference and Exhibition, Nusa Dua, Bali, Indonesia. https://doi.org/10.2118/176164-MS

Li, C., & King, M. J. (2016). *Integration of Pressure Transient Data Into Reservoir Models Using the Fast Marching Method*. Paper presented at the SPE Europec featured at 78th EAGE Conference and Exhibition, Vienna, Austria. https://doi.org/10.2118/180148-MS

Li, H., & Misra, S. (2017a). Prediction of subsurface NMR T2 distribution from formation-mineral composition using variational autoencoder. *SEG Technical Program Expanded Abstracts 2017*, 3350-3354. doi:https://doi.org/10.1190/segam2017-17798488.1

Li, H., & Misra, S. (2017b). Prediction of Subsurface NMR T2 Distributions in a Shale Petroleum System Using Variational Autoencoder-Based Neural Networks. *IEEE*

Geoscience and Remote Sensing Letters, PP(99), 1-3. doi:https://doi.org/10.1109/lgrs.2017.2766130

Li, H., & Misra, S. (2018). Long Short-Term Memory and Variational Autoencoder With Convolutional Neural Networks for Generating NMR T2 Distributions. *IEEE Geoscience and Remote Sensing Letters*.

Li, H., Misra, S., & He, J. (2019). Neural network modeling of in situ fluid-filled pore size distributions in subsurface shale reservoirs under data constraints. *Neural Computing and Applications*. doi:10.1007/s00521-019-04124-w

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., . . . Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Lingjaerde, O. C., & Christophersen, N. (2000). Shrinkage structure of partial least squares. *Scandinavian Journal of Statistics, 27*(3), 459-473.

Liu, X., Liang, Z., Zhang, Y., Wu, X., & Liao, Z. (2015). Acoustic emission signal recognition of different rocks using wavelet transform and artificial neural network. *Shock and Vibration, 2015*.

Loutas, T., Eleftheroglou, N., & Zarouchas, D. (2017). A data-driven probabilistic framework towards the in-situ prognostics of fatigue life of composites based on acoustic emission data. *Composite Structures, 161*, 522-529. doi:https://doi.org/10.1016/j.compstruct.2016.10.109

Maleki, S., Moradzadeh, A., Riabi, R. G., Gholami, R., & Sadeghzadeh, F. (2014). Prediction of shear wave velocity using empirical correlations and artificial intelligence methods. *NRIAG Journal of Astronomy and Geophysics, 3*(1), 70-81.

Mandel, J. (2009). A brief tutorial on the ensemble Kalman filter. *arXiv preprint arXiv:0901.3725*.

Mardia, K. V., & Jupp, P. E. (2009). *Directional statistics* (Vol. 494): John Wiley & Sons.

Miller, R. L., Moore, B., Viswanathan, H., & Srinivasan, G. (2017, 18-21 Nov. 2017). *Image Analysis Using Convolutional Neural Networks for Modeling 2D Fracture Propagation.* Paper presented at the 2017 IEEE International Conference on Data Mining Workshops (ICDMW).

Misra, S., Chakravarty, A., Bhoumick, P., & Rai, C. S. (2019a). Unsupervised clustering methods for noninvasive characterization of fracture-induced geomechanical alterations. *Machine Learning for Subsurface Characterization*, 39.

Misra, S., Ganguly, E., & Wu, Y. (2019b). Generalization of machine learning assisted segmentation of scanning electron microscopy images of organic-rich shales. *Machine Learning for Subsurface Characterization*, 315.

Misra, S., & He, J. (2019a). Shallow neural networks and classification methods for approximating the subsurface in situ fluid-filled pore size distribution. *Machine Learning for Subsurface Characterization*, 65.

Misra, S., & He, J. (2019b). Stacked neural network architecture to model the multifrequency conductivity/permittivity responses of subsurface shale formations. *Machine Learning for Subsurface Characterization*, 103.

Misra, S., & Li, H. (2019). Deep neural network architectures to approximate the fluid-filled pore size distributions of subsurface geological formations. *Machine Learning for Subsurface Characterization*, 183.

Misra, S., Osogba, O., & Powers, M. (2019c). Unsupervised outlier detection techniques for well logs and geophysical data. *Machine Learning for Subsurface Characterization*, 1.

Mitchell, T. M. (1997). Machine learning: McGraw-hill New York.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., . . . Ostrovski, G. (2015). Human-level control through deep reinforcement learning. *Nature, 518*(7540), 529-533.

Moore, B. A., Rougier, E., O'Malley, D., Srinivasan, G., Hunter, A., & Viswanathan, H. (2018). Predictive modeling of dynamic fracture growth in brittle materials with machine learning. *Computational Materials Science, 148*, 46-53. doi:https://doi.org/10.1016/j.commatsci.2018.01.056

Olalotiti-Lawal, F., & Datta-Gupta, A. (2015). *A Multi-Objective Markov Chain Monte Carlo Approach for History Matching and Uncertainty Quantification*. Paper presented at the SPE Annual Technical Conference and Exhibition, Houston, Texas, USA. https://doi.org/10.2118/175144-MS

Onalo, D., Adedigba, S., Khan, F., James, L. A., & Butt, S. (2018). Data driven model for sonic well log prediction. *Journal of Petroleum Science and Engineering, 170*, 1022-1037. doi:https://doi.org/10.1016/j.petrol.2018.06.072

Onalo, D., Oloruntobi, O., Adedigba, S., Khan, F., James, L., & Butt, S. (2019). Dynamic data driven sonic well log model for formation evaluation. *Journal of Petroleum Science and Engineering, 175*, 1049-1062. doi:https://doi.org/10.1016/j.petrol.2019.01.042

Onishi, T., Iino, A., Jung, H. Y., & Datta-Gupta, A. (2019). *Fast Marching Method Based Rapid Simulation Accounting for Gravity*. Paper presented at the SPE/AAPG/SEG Asia Pacific Unconventional Resources Technology Conference, Brisbane, Australia. https://doi.org/10.15530/AP-URTEC-2019-198249

Perol, T., Gharbi, M., & Denolle, M. (2018). Convolutional neural network for earthquake detection and location. *Science Advances, 4*(2), e1700578.

Pierson, K. D., Hochhalter, J. D., & Spear, A. D. (2018). Data-Driven Correlation Analysis Between Observed 3D Fatigue-Crack Path and Computed Fields from High-Fidelity, Crystal-Plasticity, Finite-Element Simulations. *JOM*, 1-9.

Pollock, J., Stoecker-Sylvia, Z., Veedu, V., Panchal, N., & Elshahawi, H. (2018). *Machine Learning for Improved Directional Drilling*. Paper presented at the Offshore Technology Conference, Houston, Texas, USA.

Radford, A., Metz, L., & Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

Rawlinson, N., & Sambridge, M. (2004). Wave front evolution in strongly heterogeneous layered media using the fast marching method. *Geophysical Journal International, 156*(3), 631-647. doi:10.1111/j.1365-246X.2004.02153.x

Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., & Lee, H. (2016). Generative adversarial text to image synthesis. *arXiv preprint arXiv:1605.05396*.

Rezaee, M. R., Ilkhchi, A. K., & Barabadi, A. (2007). Prediction of shear wave velocity from petrophysical data utilizing intelligent systems: An example from a

sandstone reservoir of Carnarvon Basin, Australia. *Journal of Petroleum Science and Engineering, 55*(3-4), 201-212.

Rovinelli, A., Sangid, M. D., Proudhon, H., & Ludwig, W. (2018). Using machine learning and a data-driven approach to identify the small fatigue crack driving force in polycrystalline materials. *npj Computational Materials, 4*(1), 35.

Sanghyun, L., & Stephen, K. D. (2018). *Optimizing Automatic History Matching for Field Application Using Genetic Algorithm and Particle Swarm Optimization*. Paper presented at the Offshore Technology Conference Asia, Kuala Lumpur, Malaysia.

Shams, M., El-Banbi, A. H., & Sayyouh, H. (2017). *A Comparative Study of Proxy Modeling Techniques in Assisted History Matching*. Paper presented at the SPE Kingdom of Saudi Arabia Annual Technical Symposium and Exhibition, Dammam, Saudi Arabia.

Shao, W., Chen, S., Eid, M., & Hursan, G. (2019). *Carbonate Log Interpretation Models Based on Machine Learning Techniques*. Paper presented at the SPWLA 60th Annual Logging Symposium, The Woodlands, Texas, USA. https://doi.org/

Shirangi, M. G., & Emerick, A. A. (2016). An improved TSVD-based Levenberg–Marquardt algorithm for history matching and comparison with Gauss–Newton. *Journal of Petroleum Science and Engineering, 143*, 258-271. doi:https://doi.org/10.1016/j.petrol.2016.02.026

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., & Riedmiller, M. (2014). *Deterministic policy gradient algorithms*. Paper presented at the Proceedings of the 31st International Conference on Machine Learning (ICML-14).

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). *Sequence to sequence learning with neural networks.* Paper presented at the Advances in neural information processing systems.

Sutton, R. S., & Barto, A. G. (1998). *Introduction to reinforcement learning* (Vol. 135): MIT press Cambridge.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*: MIT press.

Talavera, A. G., Tupac, Y. J., & Vellasco, M. M. B. R. (2010). *Controlling Oil Production in Smart Wells by MPC Strategy With Reinforcement Learning*. Paper presented at the SPE Latin American and Caribbean Petroleum Engineering Conference, Lima, Peru.

Tariq, Z., Elkatatny, S., Mahmoud, M., & Abdulraheem, A. (2016). *A New Artificial Intelligence Based Empirical Correlation to Predict Sonic Travel Time*. Paper presented at the International Petroleum Technology Conference, Bangkok, Thailand.

Terada, K. (2019). *Rapid Coupled Flow and Geomechanics Simulation using the Fast Marching Method*. Paper presented at the SPE Annual Technical Conference and Exhibition, Calgary, Alberta, Canada. https://doi.org/10.2118/199785-STU

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 267-288.

Treeby, B. E., & Cox, B. T. (2010). k-Wave: MATLAB toolbox for the simulation and reconstruction of photoacoustic wave fields. *Journal of biomedical optics, 15*(2), 021314.

Van Vliet, L. J., & Verbeek, P. W. (1993). *Curvature and bending energy in digitized 2D and 3D images.* Paper presented at the Proceedings of the Scandinavian Conference on Image Analysis.

Venugopalan, S., Xu, H., Donahue, J., Rohrbach, M., Mooney, R., & Saenko, K. (2014). Translating videos to natural language using deep recurrent neural networks. *arXiv preprint arXiv:1412.4729*.

Vink, J. C., Gao, G., & Chen, C. (2015). *Bayesian Style History Matching: Another Way to Under-Estimate Forecast Uncertainty?* Paper presented at the SPE Annual Technical Conference and Exhibition, Houston, Texas, USA.

Vinyals, O., Toshev, A., Bengio, S., & Erhan, D. (2015). *Show and tell: A neural image caption generator.* Paper presented at the Proceedings of the IEEE conference on computer vision and pattern recognition.

Wang, B., Zhou, F., Zou, Y., Liang, T., Wang, D., Xue, Y., & Gao, L. (2019). Quantitative investigation of fracture interaction by evaluating fracture curvature during temporarily plugging staged fracturing. *Journal of Petroleum Science and Engineering, 172*, 559-571.

Wang, Y., Li, C., & Hu, Y. (2018). Experimental investigation on the fracture behaviour of black shale by acoustic emission monitoring and CT image analysis during uniaxial compression. *Geophysical Journal International, 213*(1), 660-675.

Watanabe, N., Ishibashi, T., Hirano, N., Ohsaki, Y., Tsuchiya, Y., Tamagawa, T., . . . Tsuchiya, N. (2011). Precise 3D Numerical Modeling of Fracture Flow Coupled With X-Ray Computed Tomography for Reservoir Core Samples. *SPE Journal, 16*(03), 683-691. doi:10.2118/146643-PA

Wen, T.-H., Gasic, M., Mrksic, N., Su, P.-H., Vandyke, D., & Young, S. (2015). Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *arXiv preprint arXiv:1508.01745*.

Wold, H. (2004). Partial least squares. *Encyclopedia of statistical sciences, 9*.

Wong, P. M., Gedeon, T. D., & Taggart, I. J. (1995). An improved technique in porosity prediction: a neural network approach. *IEEE Transactions on Geoscience and Remote Sensing, 33*(4), 971-980.

Wu, P.-Y., Jain, V., Kulkarni, M. S., & Abubakar, A. (2018). *Machine learning-based method for automated well-log processing and interpretation*. Paper presented at the 2018 SEG International Exposition and Annual Meeting, Anaheim, California, USA. https://doi.org/

Wu, Y., Misra, S., Sondergeld, C., Curtis, M., & Jernigen, J. (2019). Machine learning for locating organic matter and pores in scanning electron microscopy images of organic-rich shales. *Fuel, 253*, 662-676.

Yang, S.-Q., Jing, H.-W., & Wang, S.-Y. (2012). Experimental Investigation on the Strength, Deformability, Failure Behavior and Acoustic Emission Locations of Red Sandstone Under Triaxial Compression. *Rock Mechanics and Rock Engineering, 45*(4), 583-606. doi:10.1007/s00603-011-0208-8

Yegnanarayana, B. (2009). *Artificial neural networks*: PHI Learning Pvt. Ltd.

Yoon, S. (2017). Numerical Simulation of the Effects of Reservoir Heterogeneity, Fractures, and Multi-Well Interference on Pressure Transient Responses Using Multistencils Fast Marching Method.

Zhang, D., Shen, A., Jiang, X., & Kang, Z. (2017). Efficient history matching with dimensionality reduction methods for reservoir simulations. *SIMULATION*, 0037549717742963. doi:10.1177/0037549717742963

Zhao, X., Popa, A. S., Ershaghi, I., Aminzadeh, F., Li, Y., & Cassidy, S. D. (2018). *Reservoir Geostatistical Estimation of Imprecise Information Using Fuzzy Kriging Approach*. Paper presented at the SPE Western Regional Meeting, Garden Grove, California, USA.

Zhou, J., Chen, M., Jin, Y., & Zhang, G.-q. (2008). Analysis of fracture propagation behavior and fracture geometry using a tri-axial fracturing system in naturally fractured reservoirs. *International Journal of Rock Mechanics and Mining Sciences, 45*(7), 1143-1152. doi:https://doi.org/10.1016/j.ijrmms.2008.01.001

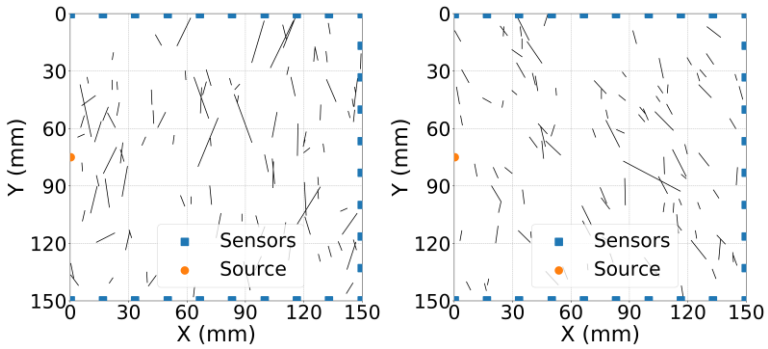
Zhou, M., & Yang, J. (2017). *Effect of the layer orientation on fracture propagation of Longmaxi Shale under uniaxial compression using micro-CT scanning*. Paper presented at the 2017 SEG International Exposition and Annual Meeting, Houston, Texas. https://doi.org/

Zhou, Z., Cheng, R., Cai, X., Ma, D., & Jiang, C. (2018). Discrimination of Rock Fracture and Blast Events Based on Signal Complexity and Machine Learning. *Shock and Vibration, 2018*, 10. doi:10.1155/2018/9753028

Zhu, Z., Burns, D. R., Brown, S., & Fehler, M. (2015). Laboratory experimental studies of seismic scattering from fractures. *Geophysical Journal International, 201*(1), 291-303. doi:10.1093/gji/ggu399

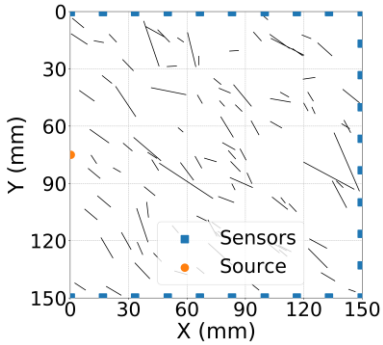
Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology), 67*(2), 301-320.

Appendix A:

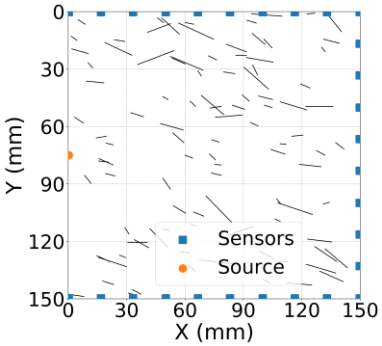


(a) Orientation = 0°

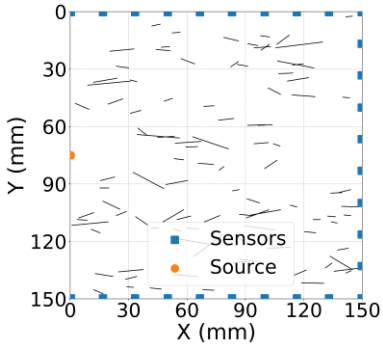
(b) Orientation = 22.5°



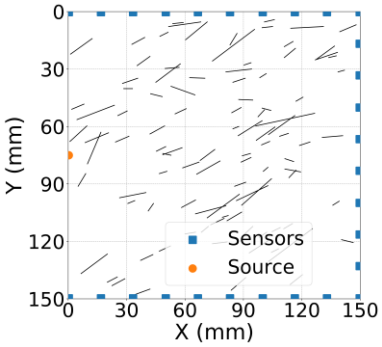
(c) Orientation = 45°



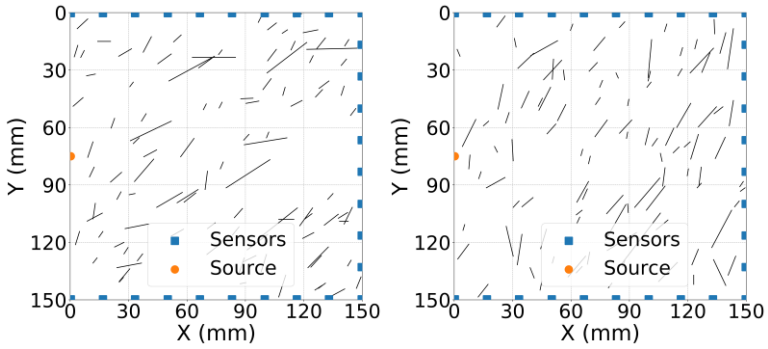
(d) Orientation = 67.5°



(e) Orientation = 90°

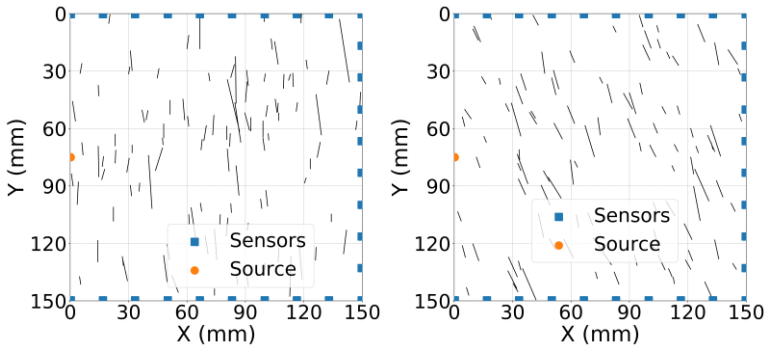


(f) Orientation = 112.5°

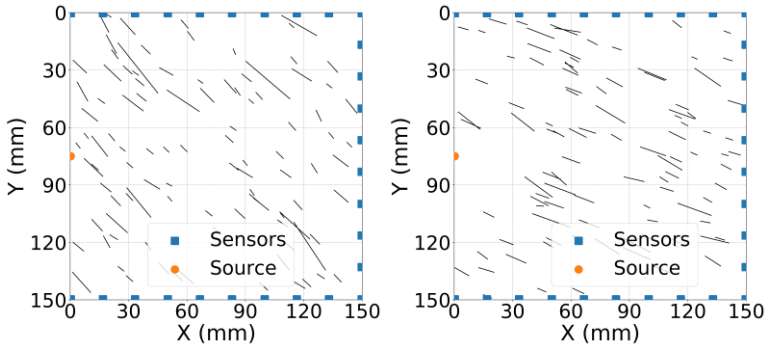


(g) Orientation = 135° (h) Orientation = 157.5°

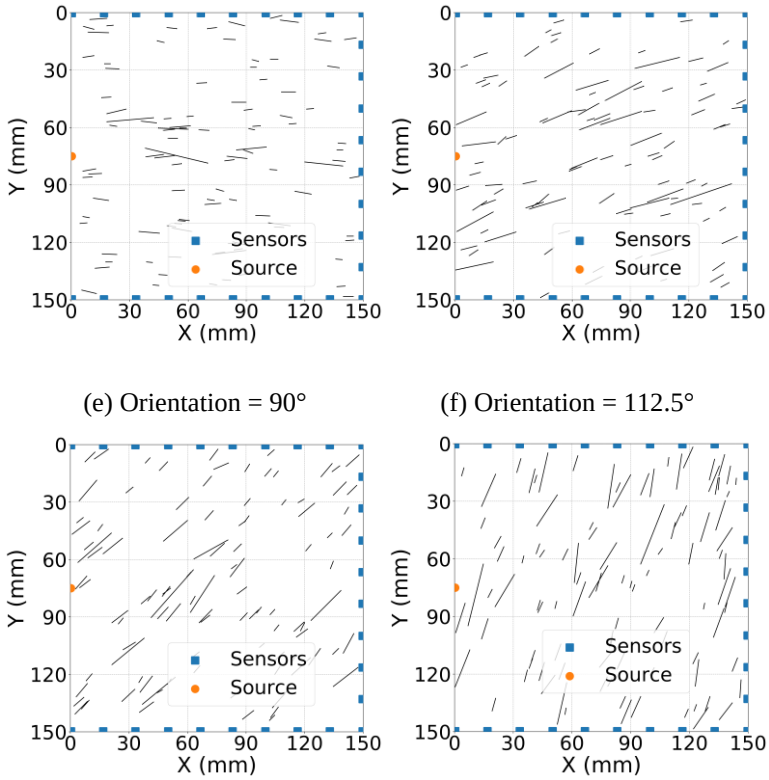
Figure A-1. Fractured models examples from experiment #3, Kappa = 10.



(a) Orientation = 0° (b) Orientation = 22.5°



(c) Orientation = 45° (d) Orientation = 67.5°



(e) Orientation = 90° (f) Orientation = 112.5°

(g) Orientation = 135° (h) Orientation = 157.5°

Figure A-2. Fractured models examples from experiment #3, Kappa = 50.

Table A-1. Prediction performance of the 8 models evaluated with 4 metrics with 95% confidence interval. The table shows the testing performance on both testing dataset from Well #1 and all depth on Well #2.

Metrics	Test Set		OLS	PLS	LASSO	ElasticNet
R^2	Well 1	DTC	0.78+/- 8.34E-05	0.78+/- 7.40E-05	0.78+/- 8.76E-05	0.78+/- 7.70E-05
		DTS	0.73+/- 1.33E-04	0.74+/- 1.24E-04	0.74+/- 1.31E-04	0.74+/- 1.31E-04
	Well 2	DTC	0.54+/- 9.27E-03	0.48+/- 9.43E-03	0.53+/- 8.67E-03	0.53+/- 1.06E-02
		DTS	0.22+/- 1.47E-02	0.20+/- 1.67E-02	0.22+/- 1.29E-02	0.21+/- 1.78E-02
MSE	Well 1	DTC	15.05+/- 5.77E-03	15.05+/- 5.12E-03	15.06+/- 6.06E-03	15.05+/- 5.33E-03
		DTS	31.99+/- 1.63E-02	31.98+/- 1.51E-02	31.98+/- 1.60E-02	31.96+/- 1.60E-02
	Well 2	DTC	10.72+/- 2.15E-01	10.71+/- 2.19E-01	10.91+/- 2.01E-01	10.86+/- 2.47E-01
		DTS	30.25+/- 5.71E-01	31.12+/- 6.47E-01	30.50+/- 5.01E-01	30.86+/- 6.90E-01
AIC	Well 1	DTC	6845.80+/- 9.64E-01	6845.55+/- 8.57E-01	6847.36+/- 1.01E+00	6845.98+/- 8.91E-01
		DTS	8745.05+/-	8744.27+/-	8744.66+/-	8743.08+/-

			1.28E+00	1.19E+00	1.26E+00	1.26E+00
	Well 2	**DTC**	6902.42+/-5.77E+01	6900.04+/-5.83E+01	6954.24+/-5.37E+01	6935.55+/-6.53E+01
		DTS	9920.64+/-5.52E+01	10000.95+/-6.07E+01	9948.03+/-4.82E+01	9974.61+/-6.40E+01
BIC	**Well 1**	**DTC**	6892.45+/-9.64E-01	6892.20+/-8.57E-01	6894.01+/-1.01E+00	6892.63+/-8.91E-01
		DTS	8791.71+/-1.28E+00	8790.93+/-1.19E+00	8791.31+/-1.26E+00	8789.73+/-1.26E+00
	Well 2	**DTC**	6950.23+/-5.77E+01	6947.85+/-5.83E+01	7002.04+/-5.37E+01	6983.36+/-6.53E+01
		DTS	9968.44+/-5.52E+01	10048.75+/-6.07E+01	9995.84+/-4.82E+01	10022.42+/-6.40E+01
Metrics	**Test Set**		**MARS**	**ANN**	**ANN Single-depth**	**ANN Multi-depth**
R^2	**Well 1**	**DTC**	0.80+/-7.76E-04	0.79+/-2.15E-03	0.64+/-3.34E-02	0.48+/-3.01E-02
		DTS	0.77+/-7.75E-04	0.76+/-3.31E-03	0.62+/-3.17E-02	0.29+/-2.32E-02
	Well 2	**DTC**	0.63+/-1.05E-02	0.50+/-2.92E-02	0.19+/-5.80E-02	0.41+/-5.75E-02
		DTS	0.59+/-1.32E-02	0.02+/-4.39E-02	0.16+/-6.53E-02	0.29+/-8.25E-02
MSE	**Well 1**	**DTC**	14.00+/-5.36E-02	14.25+/-1.48E-01	18.34+/-1.72E+00	26.67+/-1.54E+00
		DTS	28.57+/-9.46E-02	29.88+/-4.03E-01	32.74+/-2.75E+00	61.61+/-2.01E+00
	Well 2	**DTC**	8.67+/-2.43E-01	11.71+/-6.78E-01	19.15+/-1.37E+00	13.90+/-1.54E+00
		DTS	15.91+/-5.12E-01	38.08+/-1.71E+00	32.90+/-2.25E+00	27.88+/-3.24E+00
AIC	**Well 1**	**DTC**	6650.11+/-9.57E+00	6850.35+/-2.66E+01	9014.41+/-1.55E+02	9405.49+/-1.88E+02
		DTS	8446.14+/-8.37E+00	8875.71+/-3.47E+01	10561.87+/-1.13E+02	10610.17+/-1.49E+02
	Well 2	**DTC**	6254.34+/-8.25E+01	7248.21+/-1.63E+02	10239.18+/-2.90E+02	8712.78+/-1.93E+02
		DTS	8013.84+/-9.40E+01	10767.70+/-1.97E+02	11763.06+/-2.62E+02	9796.73+/-2.17E+02
BIC	**Well 1**	**DTC**	6655.94+/-9.57E+00	7322.71+/-2.66E+01	14271.40+/-1.55E+02	12793.79+/-1.88E+02
		DTS	8451.97+/-8.37E+00	9820.43+/-3.47E+01	15818.86+/-1.13E+02	11135.94+/-1.49E+02
	Well 2	**DTC**	6260.32+/-8.25E+01	7732.23+/-1.63E+02	15620.27+/-2.90E+02	12181.00+/-1.93E+02
		DTS	8019.82+/-9.40E+01	11735.74+/-1.97E+02	17144.16+/-2.62E+02	10334.90+/-2.17E+02

Table A-2. Classification accuracy with a 95% confidence interval for the 9 classifiers on the 4 experiments.

Classifiers	Experiment #1: 4 Orientations Kappa = 10	Experiment #2: 4 Orientations Kappa = 50	Experiment #3: 8 Orientations Kappa = 10	Experiment #4: 8 Orientations Kappa = 50
KNN	0.87266 +/- 0.0033	0.95058 +/- 0.0021	0.57495 +/- 0.0034	0.69308 +/- 0.0032
Linear SVM	0.91666 +/- 0.0027	0.98766 +/- 0.0011	0.67441 +/- 0.0032	0.88216 +/- 0.0022
RBF SVM	0.91933 +/- 0.0027	0.98525 +/- 0.0012	0.67512 +/- 0.0032	0.8642 +/- 0.0024
Decision Tree	0.822 +/- 0.0037	0.94983 +/- 0.0021	0.54641 +/- 0.0035	0.76512 +/- 0.0029
RandomForest	0.90483 +/- 0.0029	0.98233 +/- 0.0013	0.65766 +/- 0.0033	0.86108 +/- 0.0024
Adaboost	0.90741 +/- 0.0028	0.98 +/- 0.0014	0.64187 +/- 0.0033	0.84916 +/- 0.0025
Naïve Bayes	0.80591 +/- 0.0039	0.953 +/- 0.0021	0.56466 +/- 0.0034	0.72787 +/- 0.0031
ANN	0.90991 +/- 0.0028	0.98675 +/- 0.0011	0.65341 +/- 0.0033	0.87512 +/- 0.0023
VotingClassifier	0.92341 +/- 0.0026	0.98891 +/- 0.0010	0.68516 +/- 0.0032	0.88545 +/- 0.0022