APPLICATION OF DEEP LEARNING FOR UNDERSTANDING DYNAMIC WELL CONNECTIVITY

A Thesis

by

SHYAM KAREEPADATH SAJEEV

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Chair of Committee,	Akhil Datta-Gupta
Committee Members,	Michael J. King
	Debjyoti Banerjee
Head of Department,	Jeff Spath

August 2020

Major Subject: Petroleum Engineering

ABSTRACT

Artificial intelligence and machine learning have transformed many industries. However, the oil and gas industry is lagging in AI adaption. Currently, with the low oil prices and a considerable performance gap in the oil and gas industry, companies are looking for new ways to improve their operational efficiency. We have a promising proposition to apply state-of-the-art deep learning algorithms to reservoir management to understand the dynamic well-connectivity of reservoirs.

The deep learning algorithms, Long Short-Term Memory (LSTM) and Gated Recurrent Network (GRU) have a successful history in applying to many complex sequential and time series problems. In this thesis, we formulate the problem as a supervised deep-learning problem and use the LSTM and GRU algorithms to train a model that could identify well-connectivity. We model a single layer LSTM and GRU model with cell states (memory cells) to match the historical production rate by providing the input as the injection rate. For training purposes, we split the available data into training, validation, and testing datasets. We have also applied the Early Stopping criteria to prevent the underfitting and overfitting of the model. In the Early Stopping criteria, we monitor the error of the model in the validation dataset and select the model with minimum error in the validation set. The hyperparameters, cell size and window size, are optimized by the Grid Search method.

Although deep learning models work well, they are black-box models and do not provide any interpretability between the input features and the outputs. So, we have

applied the Permutation Feature Importance method for the model interpretability. This method calculates the reservoir connectivity by permuting or shuffling the inputs (water injection rates) one by one to the trained model and calculating the increase in the root mean square error (RMSE).

The deep learning workflow is applied to two cases: First, to a synthetic high permeability streak reservoir for proof of concept; second, to a field-scale model of the Brugge reservoir. The normalized streamline flux allocation factor validates the reservoir connectivity from the deep learning model.

DEDICATION

To my parents and brother

ACKNOWLEDGEMENTS

I would like to thank my committee chair, Dr. Akhil Datta-Gupta, and my committee members, Dr. Michael King and Dr. Debjyoti Banerjee for providing academic guidance and support throughout my study at TAMU.

I would also like to thank Dr. Srikanta Mishra from Battelle for valuable discussions related to my research.

Thanks also go to my friends and MCERI colleagues and the department faculty and staff for making my time at TAMU a great experience.

Finally, thanks to my mother, father and brother for their support and encouragement.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supervised by a thesis committee consisting of Professor Akhil Datta-Gupta [advisor] and Professor Michael J. King of the Department of Petroleum Engineering and Professor Debjyoti Banerjee of the Department of Mechanical Engineering.

All other work conducted for the thesis was completed by the student independently.

Funding Sources

This work was made possible by the financial support of the member companies of the Model Calibration and Efficient Reservoir Imaging (MCERI) consortium.

NOMENCLATURE

AI	Artificial Intelligence
ANN	Artificial neural network
CNN	Convolutional neural network
DL	Deep learning
GRU	Gated Recurrent Unit
LSTM	Long Short-Term Memory
ML	Machine learning
ReLU	Rectified linear unit
RNN	Recurrent neural network

TABLE OF CONTENTS

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

CHAPTER I

INTRODUCTION

With the advancement in learning algorithms, accelerated growth in data acquisition and storage capacity, and cheap and readily available on-demand (Cloud Computing) hardware (CPUs and GPUs) has fueled the growth of the Artificial Intelligence (AI) era. AI is transforming several industries, including financial, marketing, automotive, health care, and insurance. AI has become so crucial that many of us use it on a daily basis. These specialized algorithms have the power to analyze complex uninterpretable data and extract patterns into actionable intelligence. Some of the notable applications of AI include email spam filters, product recommendation systems, facial recognition, self-driving cars, language translation, voice search, and voice-activated assistants. (Barbounis et al. 2006; Hsieh et al. 2011; LeCun et al. 2015; Goodfellow et al. 2016; Chui et al. 2018)

AI is the ability of computers to perform tasks that usually require human intelligence. One of the main reasons for the popularity of AI is the progress of Machine Learning, Deep Learning, and Reinforcement Learning algorithms. Deep learning is based on the specialized network called the neural network, which is inspired by the human brain and tries to mimic the human brain (McCulloch & Pitts 1943)

1.1 Motivation

According to the Mackenzie report, the oil and gas industry has a performance gap of $200 billion in annual revenue. With historic low oil prices, the oil and gas

companies are looking to improve their operational efficiency and enhance the return on asset investments. Artificial intelligence and machine learning provide ways to unlock production potential for complex reservoirs and improve facilities' operational efficiency. If properly applied to a field or facility, AI can yield as high as 30-50 times the investments in the first few months of implementation (Brun et al. 2017).

The oil and gas industries have a 69% potential incremental value from AI over other analytic techniques. Still, the current AI adaption to the oil and gas industry is lagging (Chui et al. 2018). There is a need for applications of machine learning and deep learning projects to the oil and gas industry to improve efficiency and reduce operational costs. The AI business value proposition and easiness of implementation vary from project to project. One of the challenging and high business value propositions of AI is in the application of reservoir management (Haroon, S. 2018). This thesis focuses on the application of state-of-the-art deep learning techniques to reservoirs under waterflooding.

This research provides a workflow that could quickly identify unground connectivity between the producer and the injector. The connectivity is crucial for the reservoir management of fields.

1.2 Literature review: Machine learning in Oil and Gas Industry

In this section, we give a literature review of some of the applications of machine learning in the oil and gas industry. Machine learning has been applied in several areas of the oil and gas industry, both as classification and regression problems.

In geoscience, the deep neural network model has been successfully applied to seismic datasets and well logs. The machine learning is formulated as a supervised classification problem for well log correlation and seismic fault interpretations (Maniar et al. 2018; Carpenter 2019)

In drilling, the use of machine learning models has helped automate and increase the efficiency of the drilling. Machine learning has been applied as both supervised and unsupervised learning in the drilling process. Data is collected in real-time from underground sensors, to model the pore pressure and to identify the lithology. They are enabling near real-time optimization of drilling parameters (mud weight, weight on bit, and rate of penetration) to enhance the performance. Machine learning has also been applied to the geo-steering process, which is analogous to a self-driving car. (Parshall 2018; Pollock et al. 2018)

In production, machine learning methods have been successfully applied to maintenance prediction, equipment failure, identifying well-events, slug monitoring, and has been able to save time and increase the efficiency of production operations (Sneed 2017; Parshall 2018; Jansen Van Rensburg et al. 2019). A recurrent neural network (RNN) and artificial neural network (ANN) were applied to production time-series data for slug monitoring, and they observed RNN performed better than ANN (Omrani et al. 2019). In well-event identification, long short term memory (LSTM) and convolutional neural network (CNN) were used as anomaly detection models in time-series data to identify the well-events (Elichev et al. 2019).

Ben et al. (2020), used deep learning methods to predict real-time hydraulic fracturing pressure. They use expanding windows as training methods for time series data. First, several minutes of data were used to train the machine learning model to predict a few minutes of wellhead pressure. When the observed wellhead pressure is available for the initial prediction part, they retrain the machine learning model to whole data (initial several minutes + actual data of initial prediction) to predict the next few minutes. This process is continued throughout the hydraulic fracturing process. A combination of CNN-RNN and stacked RNN is used as the deep learning model.

In reservoir engineering, machine learning is used to accelerate reservoir simulation and the history matching process. History matching of a reservoir model is the process of tuning the uncertain model parameters to match the observed field data. It is an ill-posed problem i.e., the number of uncertain model parameters required to be updated is more than the number of production data points available. Hence the solution is non-unique, and multiple equally good solutions are possible for different sets of model parameters. History matching is an iterative process and requires several forward simulations to match the data points. As the complexity of the reservoir model and the number of data points increases, the computational time increases drastically. Depending upon the model, a full physics reservoir simulation can take several hours to days, and the history matching process may take weeks to months to be completed.

Extensive research is performed in accelerating reservoir simulation: Streamline Simulation (Datta-Gupta & King, 2007), Upscaling and Multiscale method (Efendiev & Hou, 2009; King et al. 2005), and reduced-order modeling (Cardoso & Durlofsky, 2010).

Then there is another class of method known as Data-Driven Methods, which does not require prior knowledge of the geological model and is built from the observed production data. Capacitance Resistance Model (CRM) (Sayarpour et al. 2009) is a data-driven method proven successful in waterflood applications. In CRM, the productivity index and allocation factor are assumed to be a constant, while in reality it can change (Guo et al. 2019).

Machine learning is used as a surrogate or proxy model for the physics-based simulation to speed up the computational process. In the compositional simulation, machine learning is used for solving the phase equilibrium problem (Gaganis & Varotsis, 2012). Cao et al. (2016) have used ANN for the forecasting of unconventional wells. Sagheer & Kotb (2019) have used a stacked LSTM to forecast the production from a conventional reservoir. The time-series data is transformed into stationary data to remove the increasing or decreasing trend in the preprocessing step. The LSTM model's input is the oil production rate from previous time steps, and it is used to predict one-time step at a time (t+1). The genetic algorithm is used to tune the hyperparameters of the LSTM model.

Artun (2016), has used ANN to map the injection rates to a production rate for given producers. The producer-injector interaction is derived from the weight from the ANN network (Olden & Jackson, 2002). The ANN was able to history match the production data and gives connectivity between the injector and producer.

1.3 Research Objective and Thesis Outline

In this research, we aim to develop a deep learning-based workflow for reservoir under waterflooding that can capture the complex reservoir physics to identify the dynamic well connectivity accurately. The permutation feature importance, ML model interpretability method, is used to understand the dynamic well connectivity of the injector-producer pair. Finally, the results from the deep learning workflow is validated with the streamline simulation method.

In Chapter II, we provide the background and training methodology behind state-of-the-art deep learning algorithms used for time series data. We also cover model interpretation methods to interpret the black-box deep learning algorithm.

In Chapter III, the deep learning algorithms are applied to a synthetic reservoir case and demonstrate the capabilities of the algorithm for accurate future prediction and dynamic well-connectivity. The streamline simulation results validate the connectivity derived from deep learning workflow.

In Chapter IV, the deep learning algorithms are applied to the Brugge Reservoir and demonstrate the capabilities to understand dynamic well-connectivity. The connectivity derived from deep learning workflow is validated with streamline simulation results.

Finally, in Chapter V, the research is concluded with a summary of the key findings. Recommendations and proposals for further research are also presented.

CHAPTER II

METHODOLOGY AND FORMULATION

Machine learning is a subset of artificial intelligence (Chollet 2018). It can be generally classified as supervised learning, unsupervised learning, and reinforcement learning. Supervised learning problems involve training in a labeled dataset (input and output are already defined). They can be further classified as regression problems (continuous output function) and classification problems (discrete output function) based on the output. Unsupervised learning problem involves training without a pre-existing labeled dataset, and it finds unknown patterns in the dataset. Unsupervised learning problems can be classified into clustering and dimensionality reduction problems (Chollet 2018). Reinforcement learning involves AI agents finding the optimal way to accomplish a goal to maximize reward (Salian 2019). In Reinforcement learning, there is no label input-output dataset; instead, the AI agent learns from the trial and error process, and the agent is rewarded or penalized based on the outcome of the actions. The overall goal of the AI agent is to maximize the reward (Kaelbling et al. 1996).

2.1 Background of Artificial Neural Network

Deep learning is a subset of machine learning based on the artificial neural network, which emphasizes the learning of feature or representation through successive layers. The neural network is a powerful feature extractor and learns hierarchically. For example, in an image recognition problem, the first layer of neural networks would detect edges from the input pixel data. The second layer learns to detect parts of the

faces from the detected edges. The detected parts of the face are passed on to the next layer as input. In the final layer, using the different parts of the faces as input, it can try to recognize the faces. In this example, the neural network's hidden layer tries to extract a more prominent feature than the input and pass it on to the next layer. The learning hierarchy is from simple features to sophisticated features (Ng 2017).

2.1.1 Architecture of Neural Network

Figure 1 shows a biological neuron, the artificial neural network is inspired by biological neurons and tries to mimic the human brain (McCulloch & Pitts, 1943). It consists of several connected weighted neurons that try to learn the output-input relationship with the help of activation functions. ANN is essentially a function approximator, which can learn any complex function. The non-linearity of the ANN is introduced by the activation function. The common activation function used in deep learning are sigmoid, tanh, and rectified linear unit (ReLU).

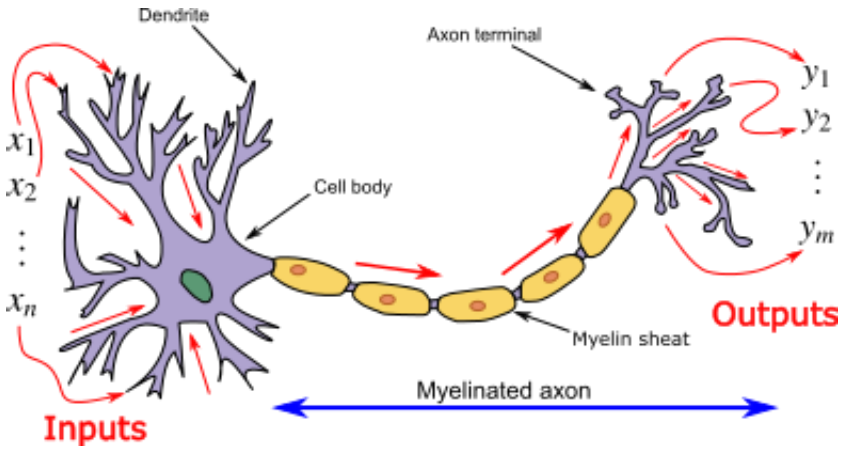


Figure 1 Biological Neuron: Signal flow from Dendrite (Inputs) to Axon (Output) (Vu-Quoc, 2018)

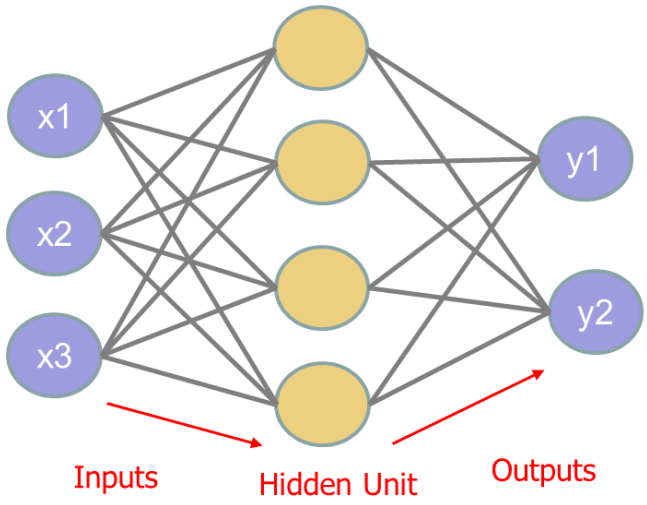


Figure 2 Architecture of an Artificial Neural Network

A simple example of a two-layer (one hidden layer and one output layer) artificial neural network is shown in Figure 2. The x is the input feature vector, and y is the output vector. The $W^{[1]}$and $b^{[1]}$ are the weights matrix and bias vector of the hidden layer. The $W^{[2]}$ and $b^{[2]}$ are the weights and bias of the output layer. Weights and bias are the trainable parameters of the network. They are adjusted in the training process to minimize the misfit between $\hat{y}$ and the actual output.

$$z^{[1]} = W^{[1]}x + b^{[1]} \quad (1)$$

$$a^{[1]} = g(z^{[1]}) \quad (2)$$

$$z^{[2]} = W^{[2]}a^{[1]} + b^{[2]} \quad (3)$$

$$a^{[2]} = g(z^{[2]}) \quad (4)$$

$$\hat{y} = a^{[2]} \quad (5)$$

The g(x) is the activation function, $a^{[1]}$ is the output of the hidden layer and $a^{[2]}$ is the output of the output layer. The equation 1-5 shows the forward pass of the neural

network. The dimension of trainable parameters of the model are $W^{[1]} \in \mathbb{R}^{4\times3}$, $b^{[1]} \in \mathbb{R}^{4\times1}$, $W^{[2]} \in \mathbb{R}^{2\times4}$ and $b^{[2]} \in \mathbb{R}^{2\times1}$

The activation function is one of the hyperparameters used in the neural network. It is essential to use a proper activation function for efficient learning of the network. Table 1 shows some of the popular activation functions the sigmoidal function, tanh, and rectified linear unit (ReLU). The activation function used depends upon the problem itself. We use the sigmoidal function for a binary classification problem because its outputs are between 0 and 1. But it is not commonly used in linear regression because, if we look at Figure 3 the derivative plot of the sigmoid function, the derivative is very small and converges to zero at both ends. Hence, the problem of vanishing gradient appears, and it would be hard to train. (Chollet 2018; Géron 2019; Kızrak 2020).

Activation Functions	**Equations**	**Range**
Sigmoidal Function	$\sigma(x) = \frac{1}{1+e^{-x}}$	[0,1]
Hyperbolic Tangent	$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} \mp e^{-x}}$	[-1,1]
Rectified linear unit (ReLU)	$f(x) = \begin{cases} 0 \; for \; x < 0 \\ x \; for \; x \geq 0 \end{cases}$	[0,∞)

Table 1 Activation Functions

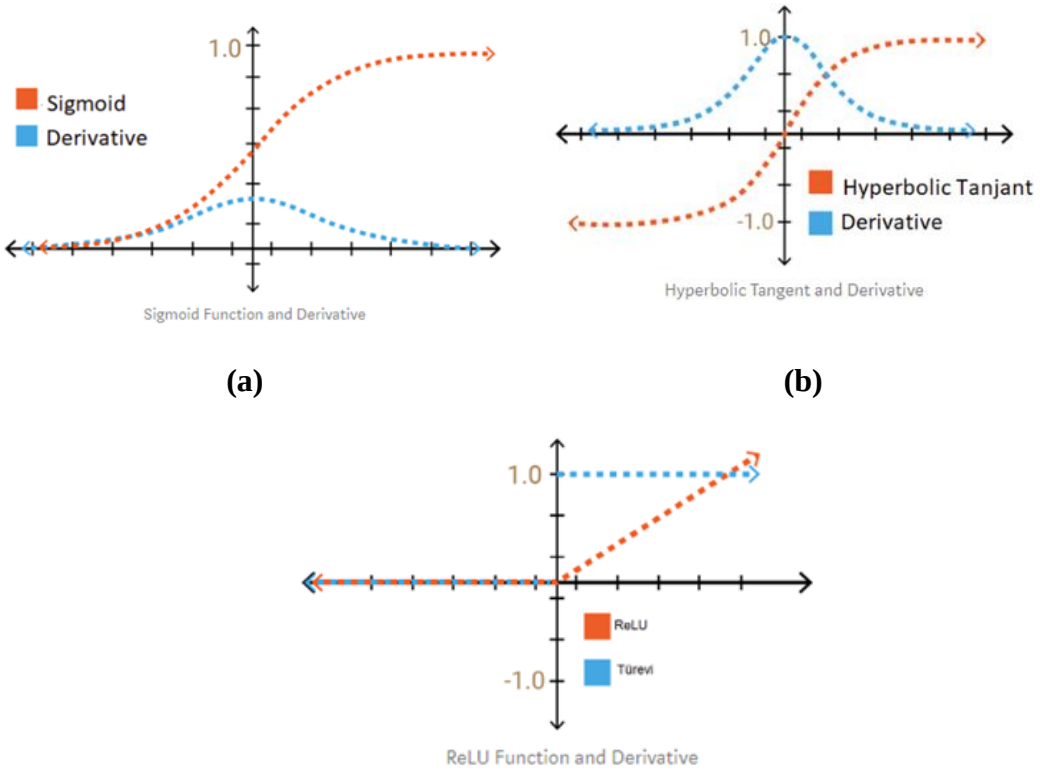


Figure 3 Activation functions and their derivates (a) Sigmoidal Function (b) Sigmoidal Function (c) ReLU Function (Kızrak, 2020)

2.1.2 Training Methodology: Backpropagation

The neural network is trained by the backward propagation of error, which is an efficient way of calculating the partial derivative of cost function w.r.t to the model parameter. The cost function is defined as the misfit between the observed and predicted values. Then, an optimization algorithm is used to update the weights and bias of the model. Figure 4 shows the flow chart for ANN training.

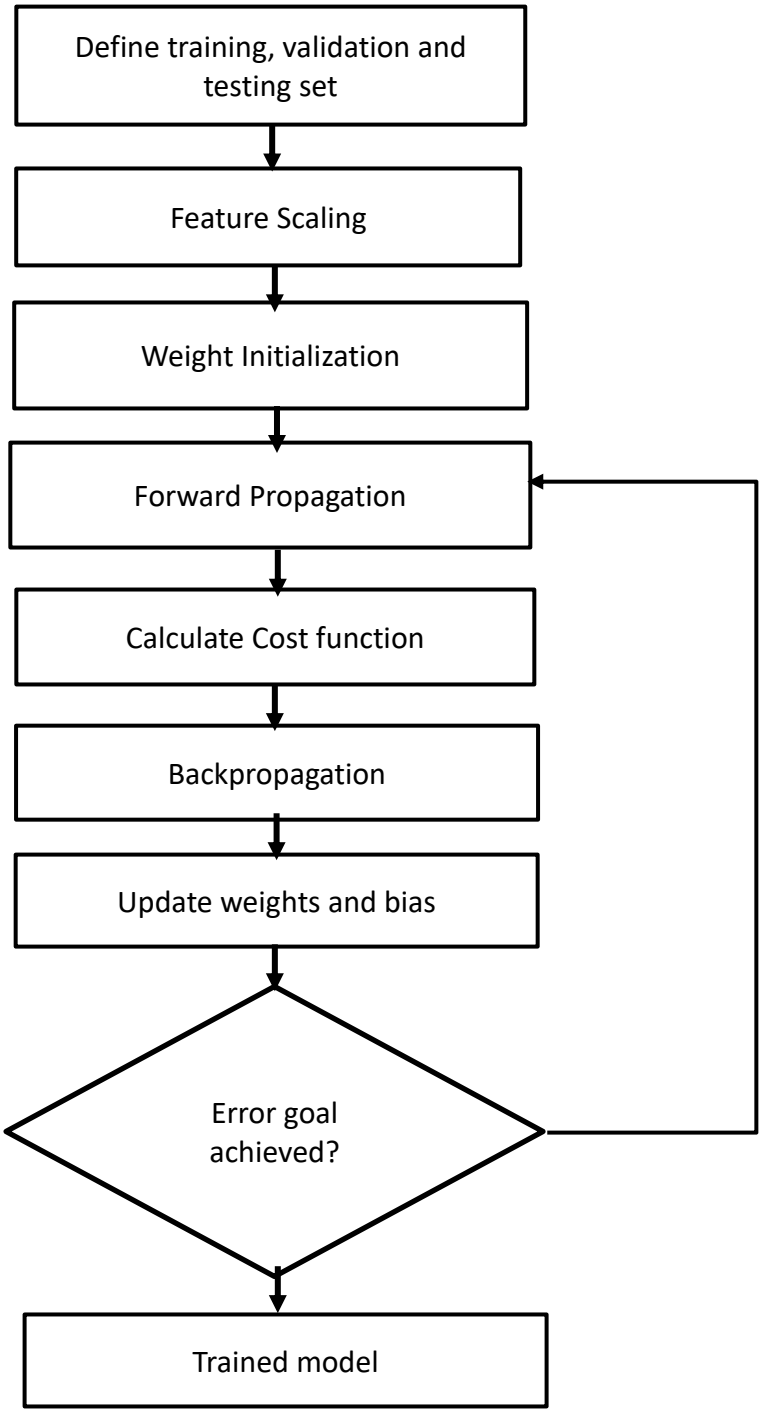


Figure 4 Flow Chart of ANN training

In a supervised learning process, the data set is divided into training, validation, and test datasets. The data is then normalized to a mean of zero and a standard deviation of one. The normalization of the data is done for the efficient optimization of neural network training. The weights are initialized by Xavier's initialization (Glorot & Bengio 2010). The weights of a layer are initialized from Gaussian distribution of mean zero and variance of 1/N, where N is the number input for the previous layer. The correct initialization of weight is essential for the convergence in a reasonable number of iterations. Once weights are initialized, we calculate the model's prediction using the input features and calculate the loss function. The loss function is selected based on the problem. For classification problems, the cross-entropy loss function is generally used. For regression problems, Mean Square Error (MSE), Root Mean Square Error (RMSE), and Mean Absolute Error (MAE) are commonly used. The cost function is defined as the average of the loss function over all the training examples. Equation 6 shows an example of a cost function. The main objective during the training process is to minimize the cost function. The backpropagation is an efficient algorithm to calculate the partial derivatives of the cost function with respect to the model parameters (weights and bias) (Rumelhart et al. 1986). After calculating the required partial derivatives, an optimization algorithm, such as Gradient Descent or Adaptive Moment Estimation (Adam), is used for minimizing the cost function.

$$C(X,\theta) = \frac{1}{2N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2 \quad (6)$$

2.1.3 Overfitting and Underfitting

Overfitting occurs when the model learns the training data so well that it captures the noise in the data. It leads to high variance and low bias for the model. Overfitting occurs when the model is excessively complicated. Overfitting could be overcome by

- Getting more training data
- Removing redundant features from the model
- Adding Regularization to the cost function
- Dropout
- Early stopping of the training

Underfitting occurs when the model is too simple to learn the training data. It leads to low variance and high bias for the model. Underfitting can be overcome by

- Adding more features to the model
- Increasing the complexity of the model

Both underfitting and overfitting are bad for the model and have poor performance on the new data set.

2.1.3.1 Early Stopping

One of the challenges in training a neural network is to determine how long the model needs to train or the number of epochs (iterations). If we under-train the model or use a small number of epochs, the model will be underfitting. Similarly, if we over-train the model or use a large number of epochs, the model will be overfitting. One of the ways to overcome this problem is by implementing a stopping criterion for the training. Early stopping is a criterion used to stop the model training when the model starts to

overfit. For early stopping, the input dataset is divided into three subsets: a training set, a validation set, and a testing set. We initialize the number of epochs to be a very large number. While the model is training in the training dataset, we monitor the misfit, or the error, in the validation set. We stop training when the error starts to increase in the validation set, i.e., the model training is stopped when the model performance on the validation set stops improving (Chollet 2018; Brownlee 2019).

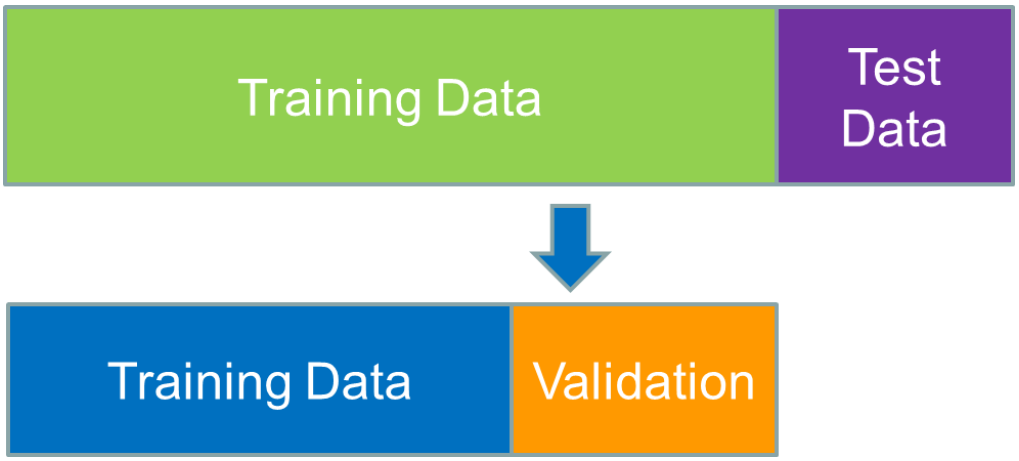


Figure 5 Training dataset, Validation dataset and Test dataset

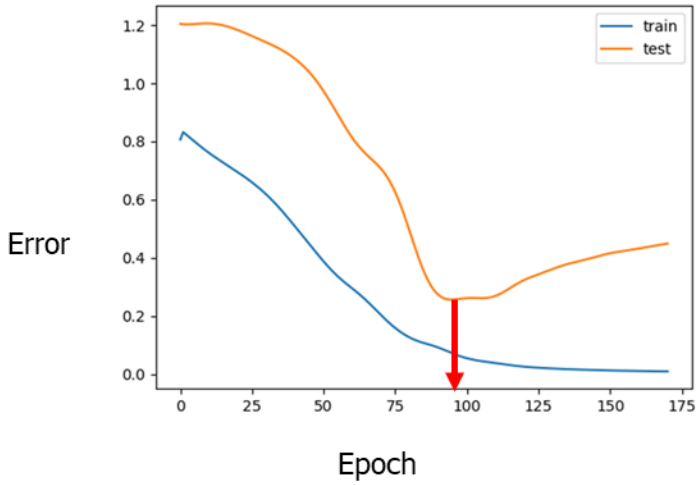


Figure 6 Error of Training dataset (Blue) and Validation dataset (Orange)

Figure 5 shows how the input dataset is split into training, validation, and testing. In Figure 6, the blue line represents the error of training set vs epoch, and the orange line represents the error of validation set vs epoch. Initially, both the training and validation error decreases show that the model is learning. Around epoch 100, the validation error starts to increase, which shows the model is trying to overfit the training dataset. So, we stop the training process and use the model at which the validation loss error starts to increase.

2.3.1.2 Dropout

Dropout is a regularization technique in which nodes/units of neural network layers are randomly removed during training. Dropout is used to prevent overfitting of a neural network by reducing the dependency of inputs for each neuron in the network. So, neurons cannot rely on a single input, and the weights are spread out like the L2

regularization technique. (Srivastava et al. 2014). Figure 7 shows an illustration of Dropout.

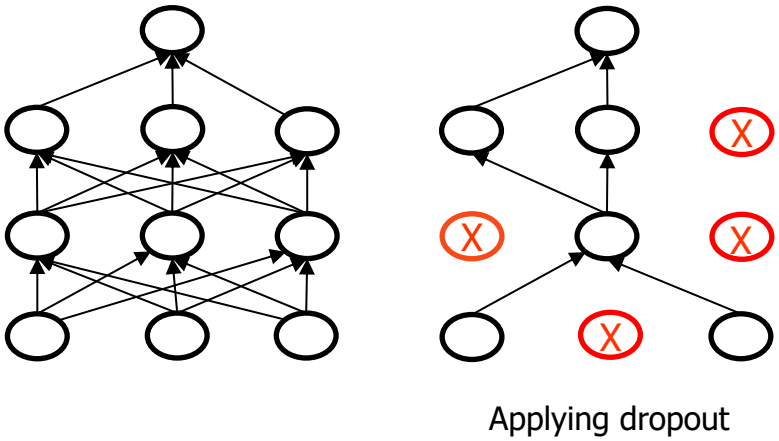


Figure 7 Illustration of dropout

2.2 Networks for Time Series data

The feedforward Neural Network shown in Figure 2 is acyclic. One of the main limitations of feedforward Neural Network is its inability to learn from time series or sequential datasets. This can be theoretically overcome by using the Recurrent Neural Network (RNN), which is a cyclic neural network. RNNs are the deepest Neural Networks of all and are powerful compared to feedforward neural networks (Schmidhuber, 2015). In practice, RNN can learn short-term data dependencies, while learning long-term data dependencies is challenging for RNN because of the vanishing gradient or exploding gradients problem during the backpropagation (Colah, 2015). This limitation of RNN is overcome by

- Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber 1997)

- Gated Recurrent Unit (GRU) (Cho et al.,2014).

2.2.1 Recurrent Neural Network

RNN is a special kind of neural network with a memory of the past. RNN has an internal loop, which allows the network to flow information from the past. So, RNN has two inputs: the data at the current time step and important information passed from the previous timestep. It also has one more advantage over the neural network, i.e., it can have different input and output sizes by masking and padding. The main limitation of RNN is in learning long term data dependencies (Vanishing Gradient) (Colah, 2015). Figure 8 shows the rolled and unrolled version of RNN.

$$h_t = \sigma_h(W_x.\ x_t + W_h.y_{t-1} + b_h) \quad (7)$$

$$y_t = \sigma(W_y h_t + b_y) \quad (8)$$

Where h_t is the hidden state

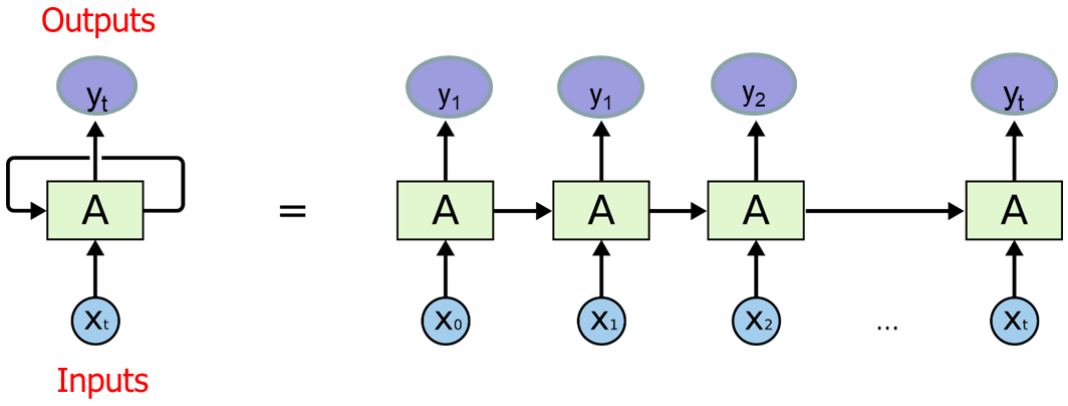


Figure 8 Recurrent Neural Network (Colah, 2015)

2.2.2 Long Short-Term Memory Networks

LSTM is a special kind of RNN that can learn long-term data dependencies. Apart from the output from the previous time step, there is another connection between the units called the cell state. This connection acts as an easy path for information to flow between units. Only minor linear interactions happen to the cell state. So it is very easy for the information to flow along unchanged. The information is added to the cell state with the help of gates. Gates are a neural network layer with a sigmoidal activation function, and [0,1] is the range of output. So, the value of one means all the information is passed to the cell states for the current timestep, while a value of zero means nothing is passed to the cell state (Colah, 2015).

Figure 9 shows the architecture of LSTM. LSTM has many real-world applications. It is well suited for classification, processing, and prediction problems. It is used in language translation, auto-completion of text, speech recognition, handwriting recognition, anomaly detection and recently has been used for time series data forecasting.

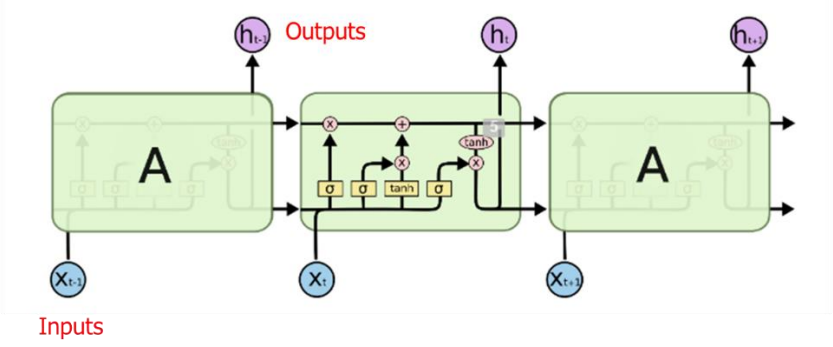


Figure 9 LSTM Architecture (Colah, 2015)

LSTM architecture consists of three gates (Colah, 2015):

- Forget Gate: Figure 10 shows the Forget Gate (f_t), it determines how much of the redundant information to erase from the output of the previous time step (h_{t-1}) based on the current time step input (x_t). In the Forget Gate, information is processed by the sigmoidal activation function, shown in Equation 9

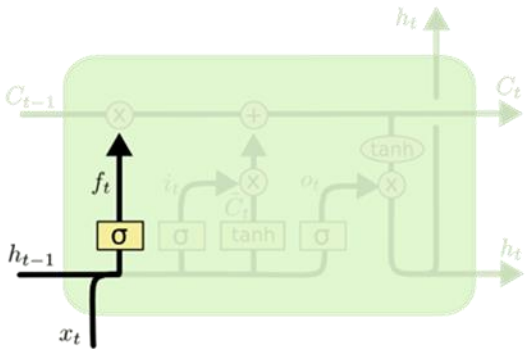


Figure 10 Forget Gate (Colah, 2015)

$$f_t = \sigma\left(W_{fx}.\ x_t + W_{fh}.h_{t-1} + b_f\right) \quad (9)$$

- Input Gate: The Input Gate decides what new information needs to be stored in the cell state (memory) from the current input. This is a two-step process: first, the sigmoidal layer updates the input layer shown in Figure 11 and second, the tanh layer updates the cell state shown in Figure 12. Equation 10 and 11 show the operation of the input gate

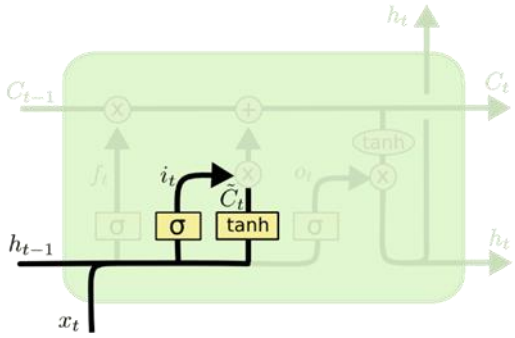


Figure 11 Input Gate (Colah, 2015)

$$i_t = \sigma(W_{ix}.\ x_t + W_{ih}.h_{t-1} + b_i) \quad (10)$$

$$\hat{C}_t = tanh(W_{cx}.\ x_t + W_{ch}.h_{t-1} + b_c) \quad (11)$$

Now we update the cell state with the information from the Forget Gate and Input Gate

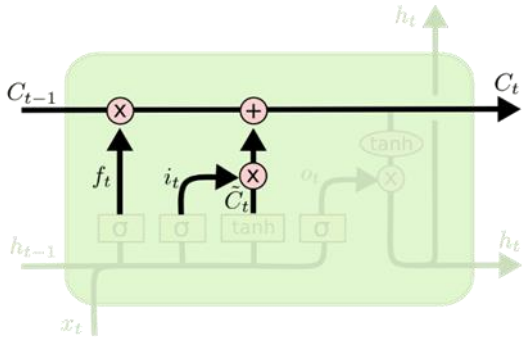


Figure 12 Updating the cell state (Colah, 2015)

$$C_t = f_t * C_{t-1} + i_t * \hat{C}_t \quad (12)$$

- Output Gate: The output gate decides what to output based on the updated cell state, current input and previous output. The output is based on the cell state but is filtered by the output gate. The filtered value is then

multiplied with tanh (C_t) to make sure the output is in between -1 and 1.The Output Gate is shown in Figure 13.

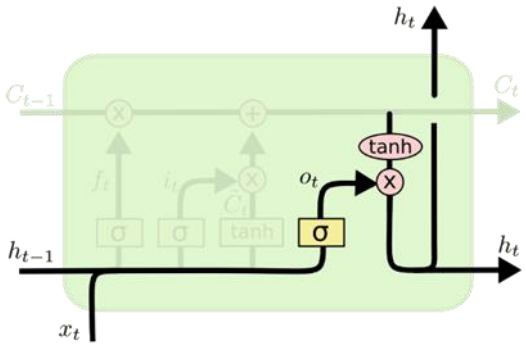


Figure 13 Output Gate (Colah, 2015)

$$o_t = \sigma(W_{ox}.\ x_t + W_{oh}.h_{t-1} + b_o) \quad (13)$$

$$h_t = o_t * \tanh(C_t) \quad (14)$$

The LSTM output is the duplicate of the short-term state (h_t): $y_t=h_t$

The Equations 9-14 show the fundamental equations of the LSTM model. The trainable parameters of the LSTM model are

- Input Weights: W_{fx} , W_{ix} , W_{cx}, $W_{ox} \in \mathbb{R}^{N\times M}$
- Recurrent Weights: W_{fh} , W_{ih} , W_{ch} , $W_{oh} \in \mathbb{R}^{N\times N}$
- Bias: b_f , b_i , b_c , $b_o \in \mathbb{R}^N$

Where N is cell size and M is the number of inputs.

Sequence networks such as RNN, LSTM, and GRU are trained by unrolling the network through time and using backpropagation; this is called backpropagation-

through-time (Géron 2019). A modified version of backpropagation-through-time called truncated backpropagation in time is also used for training.

2.2.3 Gated Recurrent Unit

The Gated Recurrent Unit is a simplified variant of the LSTM (Cho et al.,2014). Figure 14 shows the architecture of GRU network. The GRU has a more parsimonious architecture than the LSTM network; the GRU has fewer parameters to train than the LSTM for a given problem. The LSTM has two connections between cells (previous output and cell state memory). In contrast, the GRU has only one connection between units (previous output). There are only two gates for the GRU: The Update Gate, and the Forget Gate (Colah 2015; Chollet 2018; Géron 2019).

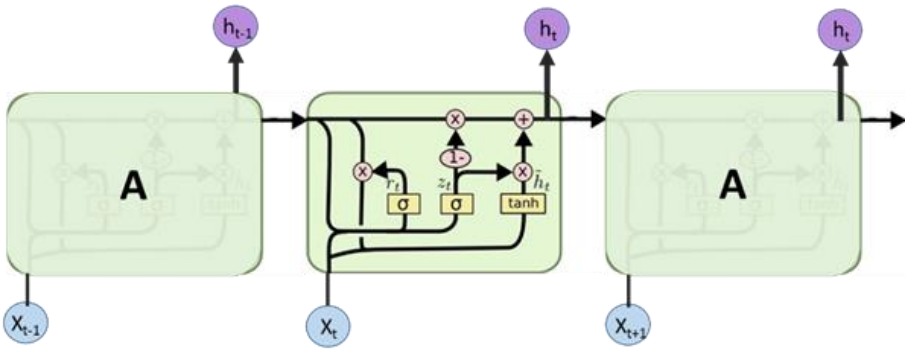


Figure 14 GRU Architecture (Colah, 2015)

$$z_t = \sigma(W_{zx}.\ x_t + W_{zh}.h_{t-1} + b_z) \quad (15)$$

$$r_t = \sigma(W_{rx}.\ x_t + W_{rh}h_{t-1} + b_r) \quad (16)$$

$$\hat{\mathrm{h}}_t = tanh(W_{hx}.\ x_t + W_{hr}.r_t * h_{t-1} + b_h) \quad (17)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \hat{\mathrm{h}}_t \quad (18)$$

The Equations 15-18 show the fundamental equations of the GRU model. The trainable parameters of the GRU model are

- Input Weights: W_{zx} , W_{rx} , $W_{hx} \in \mathbb{R}^{N \times M}$
- Recurrent Weights: W_{zh} , W_{rh} , $W_{hr} \in \mathbb{R}^{N \times N}$
- Bias: b_z , b_r , $b_h \in \mathbb{R}^{N}$

Where N is cell size and M is the number of inputs.

2.2.4 Training Strategy for Time Series Data

The cross-validation strategies are different for problems with time-series data. The traditional K-fold cross-validation techniques do not work properly because of the temporal dependencies in the data. A sliding window and expanding window technique can be used for time-series data. Figure 15 shows the sliding window and expanding window. In the sliding window approach, the training dataset size (window) is fixed and is tested against a fixed window size. The training data set is continuously growing in an expanding window approach and tested against a fixed data set. Generally, we start with an expanding window for training and continue until the window size has grown sufficiently large and then switch to a sliding window (Bell 2019; Yang 2019)

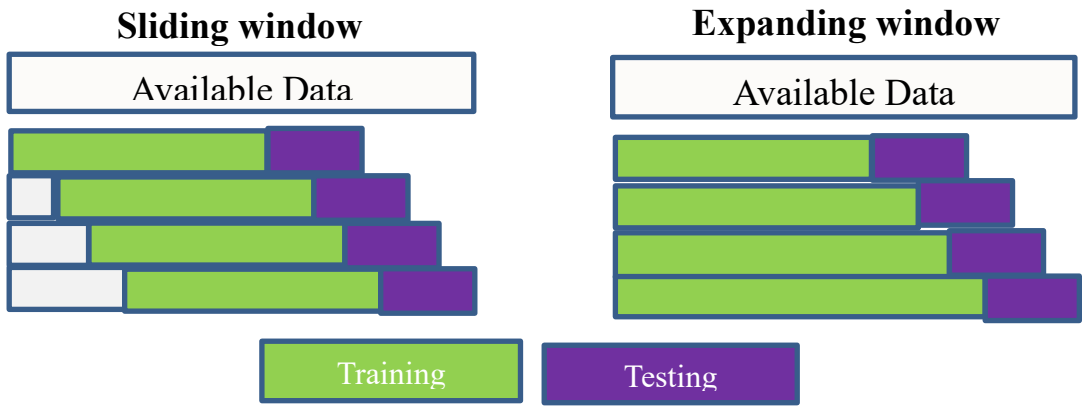


Figure 15 Sliding window and Expanding window

2.2.4 Hyperparameter Optimization

A machine learning model consists of model parameters and hyperparameters. The model parameters, such as weights and bias, are tuned during the training process to match the output data. The model parameters are initialized randomly at the start of training and then optimized during the training process. There are some parameters that need to be defined before the training starts and cannot be tuned during the training process; these are called hyperparameters. Hyperparameters are the high-level parameters that decide on the optimization process and the architecture or complexity of the model. There are many examples of hyperparameters:

- Learning rate
- Number of Epochs
- Hidden layer
- Hidden Units
- Activation Function

The tuning process of hyperparameters is as follows (Chollet 2018)

1. Select hyperparameters based on intuition or randomly
2. Build the machine learning model based on the hyperparameters
3. Train the model on the training dataset and record the performance on the validation dataset
4. Repeat the process for different values of hyperparameters

The hyperparameters are selected from the best performing model on the validation set. There is a systematic approach for tuning the hyperparameters. Some of the techniques used for hyperparameter tuning are

- Grid search
- Random search
- Bayesian Optimization
- Evolutionary Algorithm

2.3 Model Agnostic Interpretation Methods

2.2.1 Why model Interpretability?

Regardless of the problem we are trying to solve, the machine learning interpretability is always preferred. It is important for the model to give insights or intuition between the input and the output data so that it is easy for the general audience to understand and better accept the machine learning model. However, in reality, most of the machine learning models that we use nowadays are black-box models and are not easily interpretable. Hence, a model agnostic interpretation model is necessary to make sense of the black-box model.

For many problems where we do not have much understanding, with the use of opaque machine learning models we were able to get good results. However, the algorithm's main contribution is when we can explain why things are happening along with the predictions. The model interpretation helps us to find the hidden scientific information within the data, which helps us to gain knowledge and bring trust to the prediction.

The model accuracy vs. model interpretability trade-off explains that a complex model used to solve a complex problem is hard to interpret (Kuhn & Johnson, 2013). The model's accuracy or performance is usually increased by adding more features or increasing the complexity of the model; as the model's complexity increases, the model's interpretability decreases. For example, a logistic regression or decision tree model is easy to interpret but does not provide an excellent performance or accuracy to a complex problem. On the other hand, complex models like ensemble models (Random forest, XGboost), a deep neural network works well with real problems but does not provide a straightforward interpretation from the model. Hence, an interpretation method is needed to be applied to a complex trained model (Molnar 2019).

Molnar (2019) has explained model interpretation methods in detail. Some of the standard model interpretation methods used are

- Permutation Feature Importance (Brieman 2001; Fisher et al. 2019)
- Shapley Additive exPlanations (SHAP) (Lundberg & Lee 2017)
- Local Interpretable Model-agnostic Explanation (LIME) (Ribeiro et al. 2016)
- Global surrogate
- Partial Dependence Plot (PDP)
- Individual Conditional Expectation (ICE)

2.2.1 Permutation Feature Importance

Brieman (2001) introduced the concept of permutation feature importance method for the random forest. Later, based on the same idea Fisher et al. (2019)

introduced a model interpretation method of permutation feature importance. This method works on a simple idea. It measures the increase in model prediction error when permuting the feature (Molnar 2019). The original prediction error is compared to the permuted model prediction error for each feature. If the feature is important, the permutation of that feature will increase the model prediction error. Similarly, if the feature is not important, then the permutation of the feature will not change the model prediction error from the trained model. Figure 16 shows an illustration of calculating reservoir connectivity by permutation feature importance.

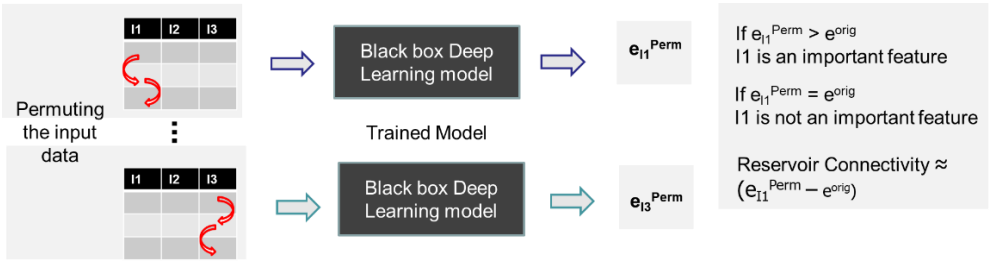


Figure 16 Reservoir Connectivity by Permutation Feature Importance

CHAPTER III

APPLICATION TO A SYNTHETIC STREAK RESERVOIR

3.1 Model Description

The deep learning algorithms are applied to a synthetic reservoir case for proof of concept. The synthetic case used is taken from Sayarpour et al. (2007) and Albertoni et al. (2003), with modification in the relative permeability table. The streak reservoir is a two-dimensional reservoir model with four producers and five injectors. The permeability field is shown in Figure 17; there are two high permeability connections between injector and producer, I1-P1 1000 mD and I3-P4 500 mD. Everywhere else, the permeability is constant 5 mD. The producers are at constant BHP constraint, and the injectors are at a rate constraint. Table 2 shows the model description for the reservoir. The reservoir model is simulated using a commercial reservoir simulator (Eclipse) to get the production response. It is incompressible and the voidage replacement ratio is 1.008.

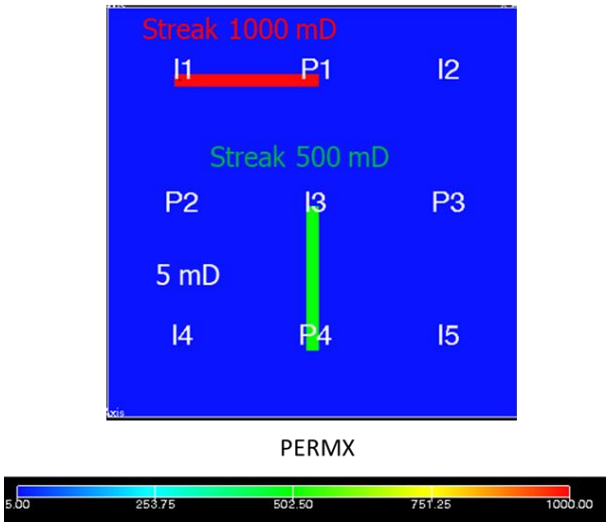


Figure 17 Streak reservoir permeability

Model Description		
Grid Block Number	31 x 31 x 1	
Grid Block Size	80 x 80 x 65 ft	
Reservoir Permeability	P1-I1	1000 mD
	P4-I3	500 mD
		5 mD
Reservoir Porosity	0.18	
Producer BHP Constraint	250 psia	
Injector-Producer Distance	800 ft	

Table 2 Model description for streak reservoir

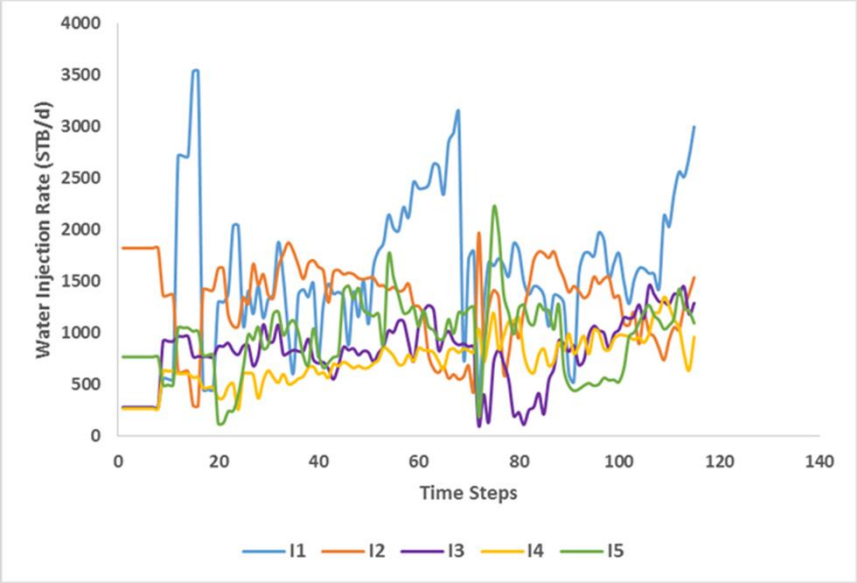


Figure 18 Plot of water injection rate

Figure 18 shows the water injection rates for the synthetic case. The water injection data is taken from Sayarpour et al. (2007). A varying water injection scenario is

used to capture the injection signal at producer; the injection profile is for approximately eight years.

3.2 Exploratory Data Analysis

Figure 19 shows the box and whisker plot of the injection rates for the five injectors. The injector I1 is most varying, and injector I3 is the least varying. There is a difference of more than 500 bbl/day between the average of the highest and the lowest injector.

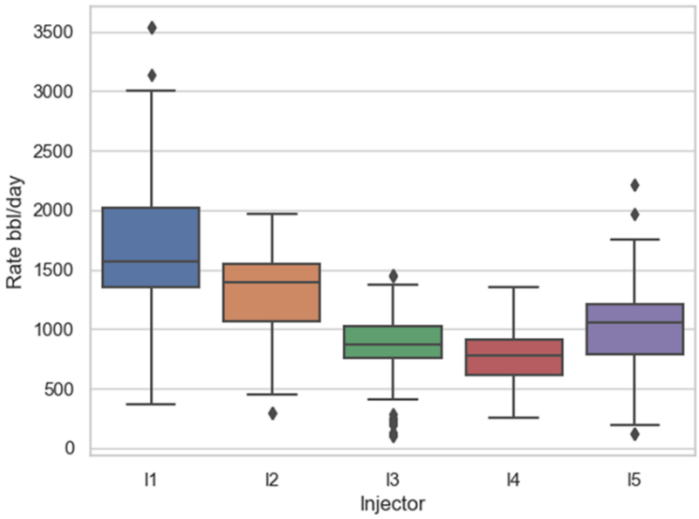


Figure 19 Box and whisker plot of injection rates

Pearson correlation coefficient is used to understand how well the injection data and production response are correlated to each other. Equation 19 shows the Pearson correlation coefficient and it measures the linear correlation between two variables. A coefficient of 1 means a perfect positive correlation, -1 means perfect negative correlation, and 0 means no linear correlation.

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y}))}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \quad (19)$$

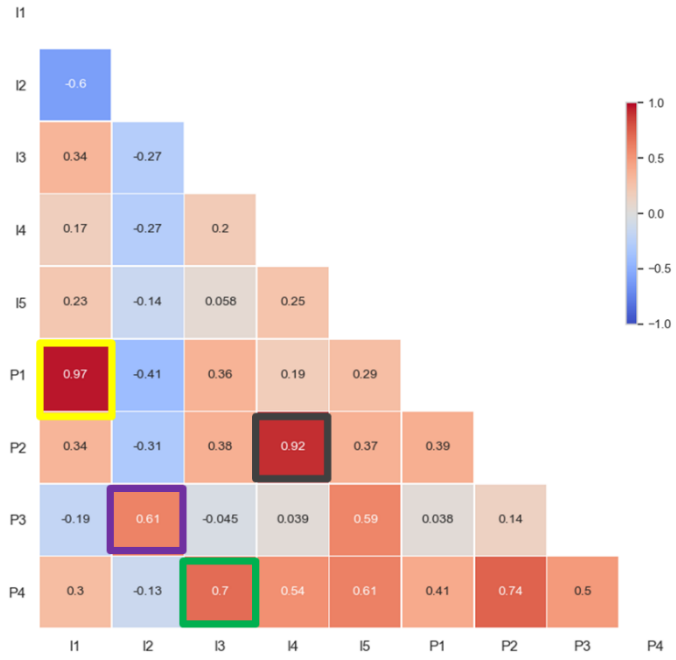


Figure 20 Correlation matrix of input and output data

Rank	Cross Correlation	Value	
1	P1-I1	0.97	High Perm
2	P2-I4	0.92	
3	P4-I3	0.70	High Perm
4	P3-I2	0.61	

Table 3 Connectivity ranking from cross correlation

Figure 20 shows the correlation matrix of injection rates and liquid production rates. The cross-correlation coefficient was able to detect the two high permeability streak P1-I1 and P4-I3 in top 4 connectivity, but not in the correct order. Table 3 shows

the connectivity list. The cross-correlation with time lagged injection rate is shown in Figure 21. This plot shows that even with time lag simple cross correlation was unable to identify the connectivities correctly. We will compare the connectivities from cross-correlation to the connectivities from the physics-based streamline simulation.

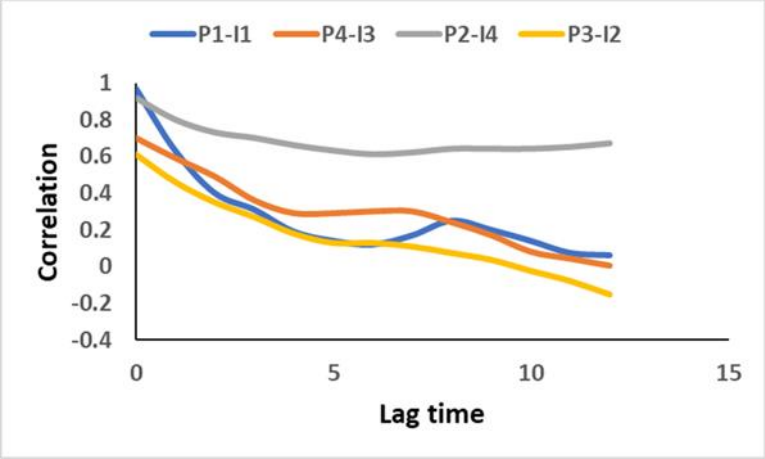


Figure 21 Cross-correlation with time lag

3.3 Training

Problem Statement Prediction of future liquid production rate for all producers given the future water injection rate, and the history of liquid production rate, and water injection rate. The model is then used to understand injector-producer interactions.

An LSTM and GRU model are trained. The input variable is the injection rate, which is trained to match the liquid production rate. Internally, the LSTM and GRU model passes the liquid production rates (outputs) from the previous time step to the current time step through the connection. Figure 22 shows training process.

For the training process, the dataset is split into training, validation, and testing dataset. A single layer GRU and LSTM model is implemented in python using Keras and TensorFlow framework (Chollet 2018). Root Mean Square Error (RMSE) of the observed and predicted value is selected as the performance metrics/ objective function. Weights are updated through the backpropagation of error to calculate the gradients. The Adaptive Moment Estimation (ADAM) optimizer is used to minimize the objective function. The hyperparameters for the model are optimized by using the Grid Search method. Several models are trained for different hyperparameters: cell size and window size. The model with minimum RMSE is selected as the optimal model.

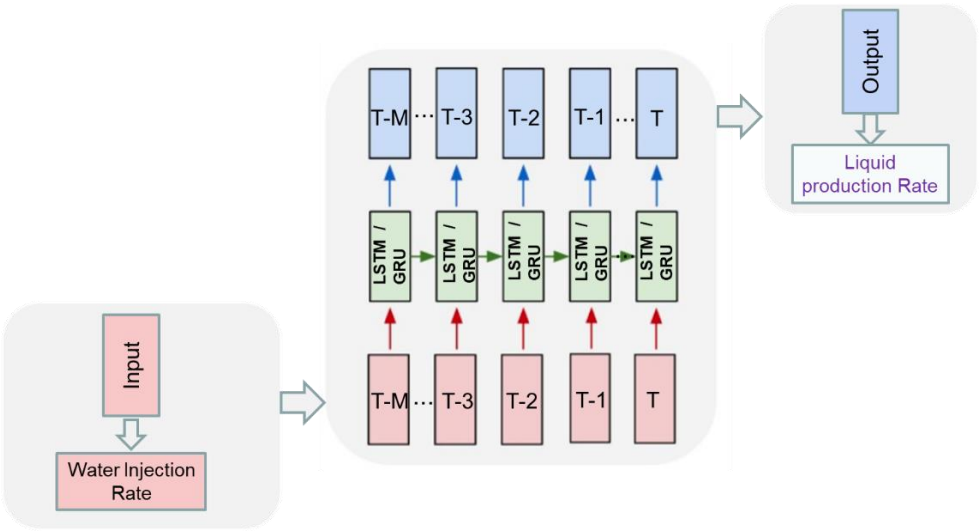
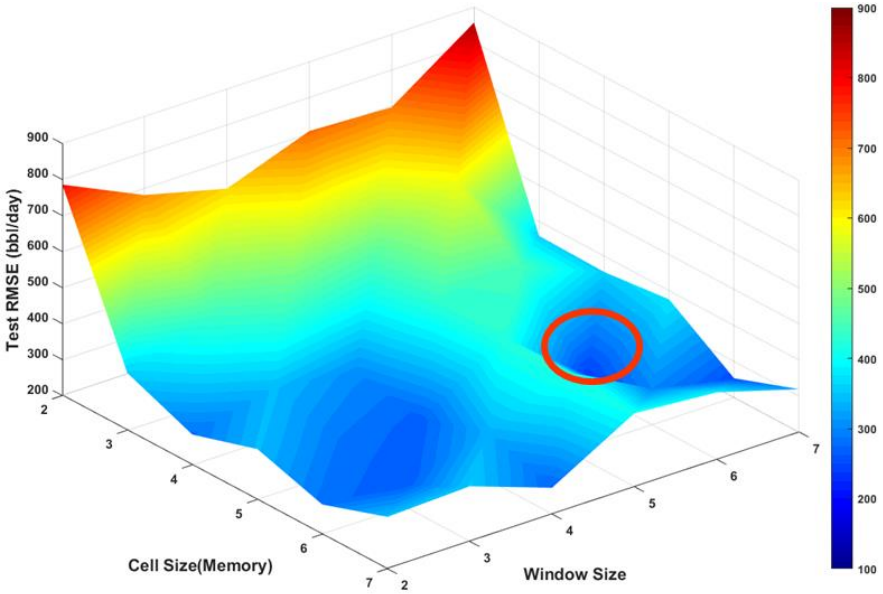


Figure 22 Training Process

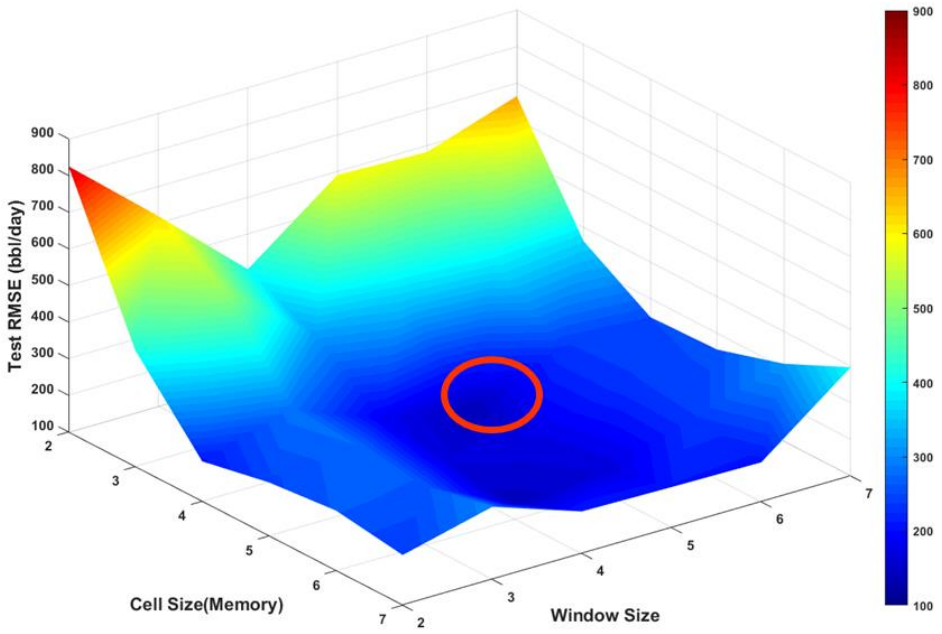
3.4 Results and Discussion

3.4.1 Hyperparameter Optimization

The hyperparameters for the LSTM and GRU network can be classified into two types. The first type are the hyperparameters that affect the architecture of the model. These are cell size (memory) and window size. The second type are the hyperparameters that affect the learning algorithm. These are learning rate, number of epochs, and activation function. We optimize the number of cell size and window size by the Grid Search method and the number of epochs by the Early Stopping method. We do not optimize the learning rate because the training process does not take much time (less than a minute), with the default learning rate of 0.001. The activation function is also set to be the tanh activation function (default).



(a)



(b)

Figure 23 Grid Search Hyperparameter Optimization (a) LSTM (b) GRU

Figure 23 shows the grid search optimization of LSTM and GRU network. The optimal model for LSTM is at cell size 5 and window size 6 with RMSE of 242 bbl/day. The optimal model for GRU is at cell size 4 and window size 5 with RMSE of 136 bbl/day. Since the GRU model outperforms the LSTM model for this case, we select the GRU model for further analysis.

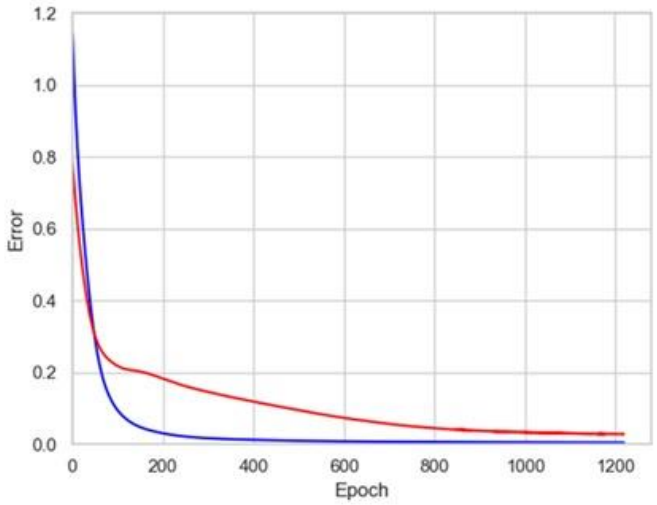


Figure 24 Diagnostic plot of GRU model: Cost function vs Epoch, Training (Blue) and Validation (Red)

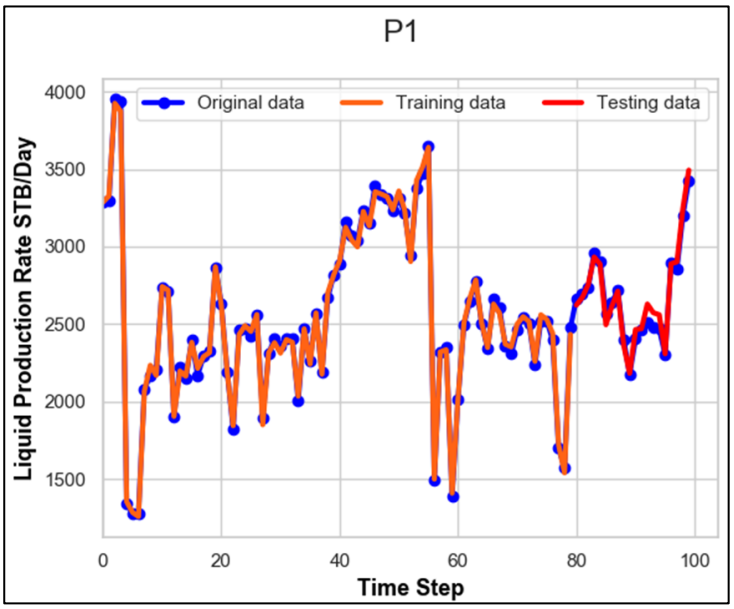
Layer (type)	Output Shape	Param #
gru (GRU)	(1, 5, 4)	120
dense (Dense)	(1, 5, 4)	20

Total params: 140
Trainable params: 140
Non-trainable params: 0

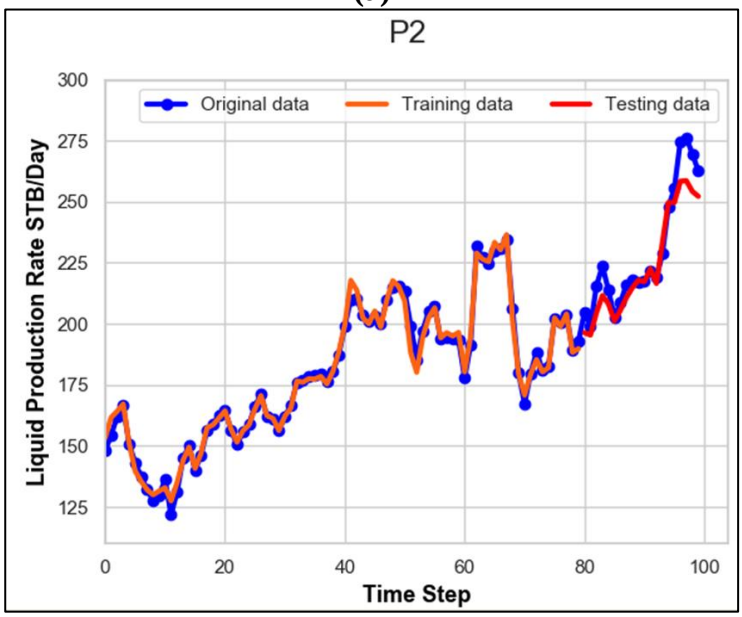
Figure 25 GRU model description

Figure 24 shows the diagnostic plot of the GRU model. We can see that both the training error (blue line) and validation error (red line) decreases with the number of iterations. This shows that the model is a well-trained GRU model. Figure 25 shows the model description of the GRU model. It has a total of 140 trainable parameters.

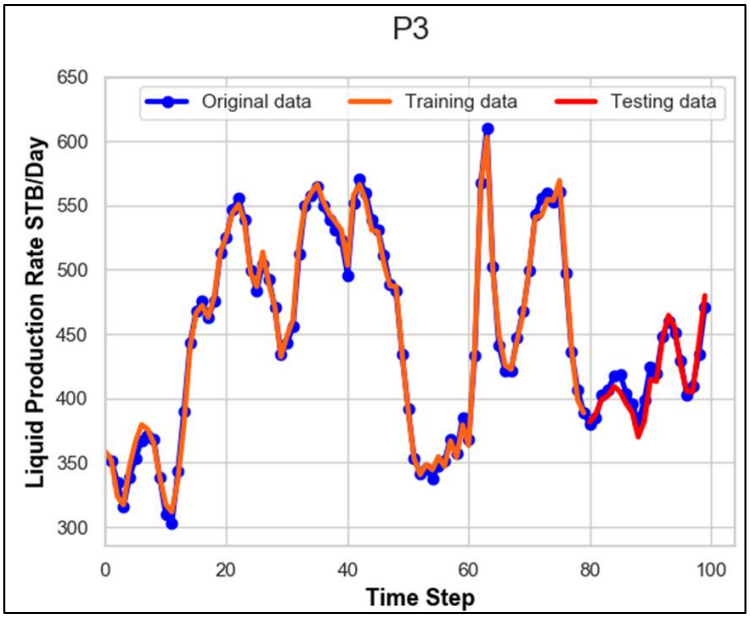
3.4.2 Liquid Production Rate Forecasting



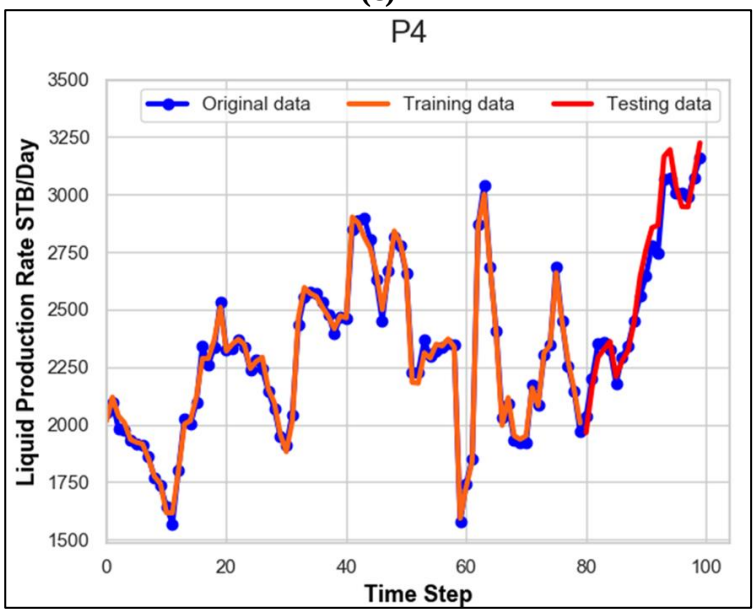
(a)



(b)



(c)



(d)

Figure 26 Liquid production rate prediction (a) Well P1 (b) Well P2 (c) Well P3 (d) Well P4

Figure 26 shows the liquid production rate forecasting. The GRU models have good fitting in the training data and were able to forecast the rates. For well P2 there is a small misfit at the end of the prediction.

3.4.3 Reservoir Connectivity by Permutation Feature Importance

The reservoir connectivity is inferred from the trained GRU model by using the permutation feature importance method. The contribution of injector I to producer P is measured by the increase in error (RMSE) with respect to original error (RMSE) when permuting the input data of the injector I to the model. Table 4 shows the increase in RMSE of GRU model when permuting injectors. Table 5 shows the top four connectivities from the permutation importance method. Both the high permeability connections P1-I1 and P4-I3 are captured by this method.

	P1	P2	P3	P4
I1	931	7	10	37
I2	201	6	70	151
I3	131	5	11	270
I4	47	24	4	122
I5	38	6	51	220

Table 4 Increase in RMSE of GRU model when permuting injectors

Rank	Machine Learning	Value	
1	P1-I1	931	High Perm
2	P4-I3	270	High Perm
3	P4-I5	220	
4	P1-I2	201	

Table 5 Connectivity ranking by permutation feature importance

3.4.4 Validation of Reservoir Connectivity by Streamline Simulation

Streamline simulation helps to understand the underground flow visualization of flux. Destiny, an inhouse streamline simulation software of the MCERI group, is used in this case for streamline tracing, flux allocation factor, and calculation of the minimum time of flight (TOF) of the top 10% streamline. Figure 27 shows the time of flight from injector and producer at a timestep.

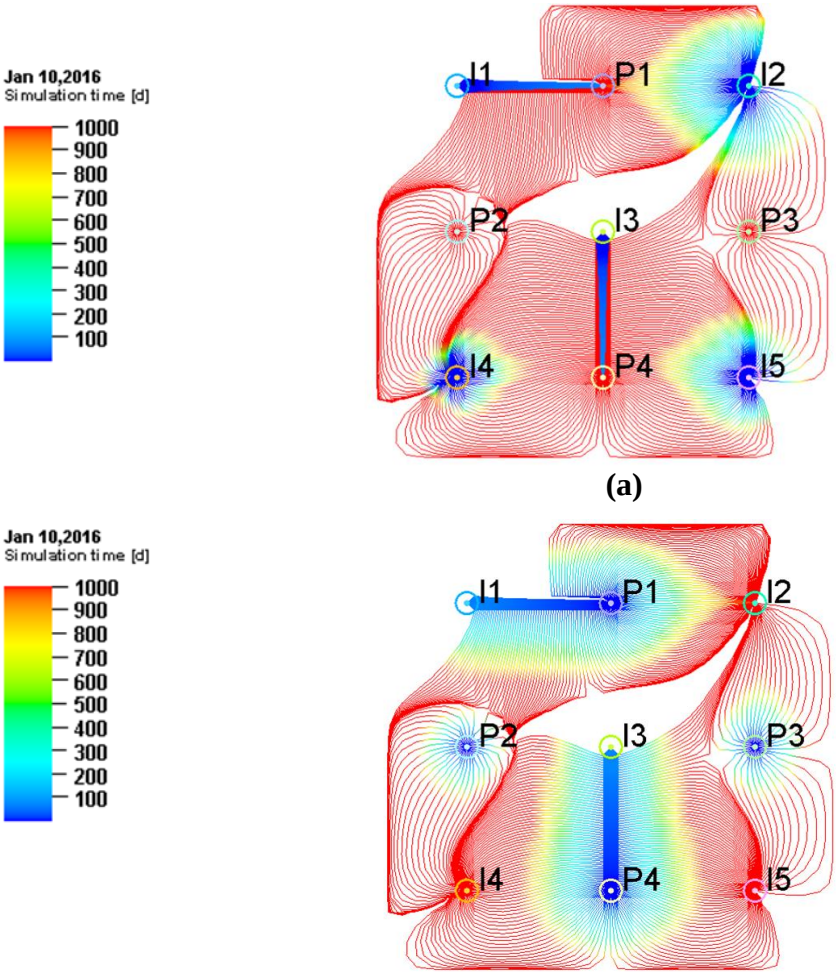
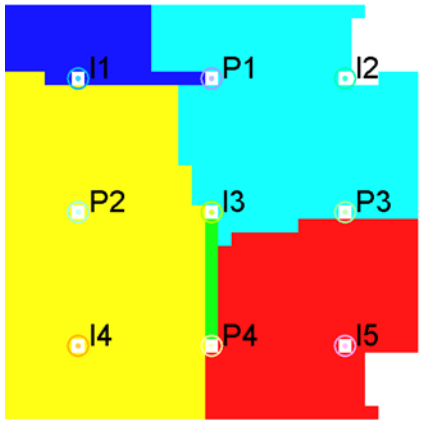
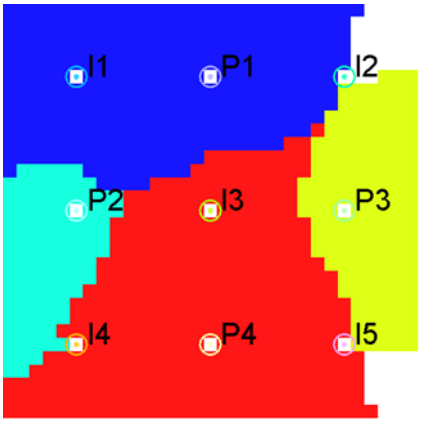


Figure 27 (a) TOF from injector (b) TOF from producer

Figure 28 shows the injector partition and producer partition. From the producer partition, we can see that Producer P4 is connected to 4 injectors (I2, I3, I4, and I5).



(a)



(b)

Figure 28 (a) Injector Partition (b) Producer Partition

	P1	P2	P3	P4
I1	1647	0	0	0
I2	895	0	254	113
I3	0	0	0	970
I4	82	182	0	520
I5	0	0	137	855

Table 6 Time average streamline flux allocation factor

	P1	P2	P3	P4	
I1	1.00	0.00	0.00	0.00	1647
I2	0.71	0.00	0.20	0.09	1262
I3	0.00	0.00	0.00	1.00	970
I4	0.10	0.23	0.00	0.66	784
I5	0.00	0.00	0.14	0.86	992

Table 7 Time average normalized streamline flux allocation factor

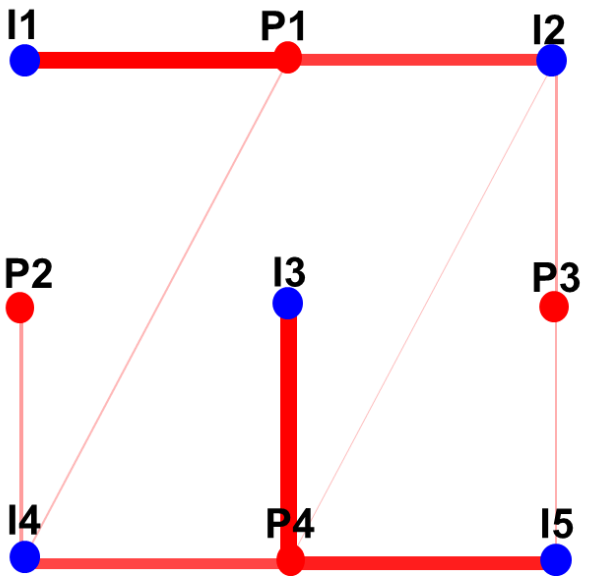
	P1	P2	P3	P4
I1	68	0	0	0
I2	1036	0	2489	4787
I3	0	0	0	87
I4	18675	2962	0	2020
I5	0	0	4571	1364

Table 8 Minimum TOF for fastest 10 percentage streamlines

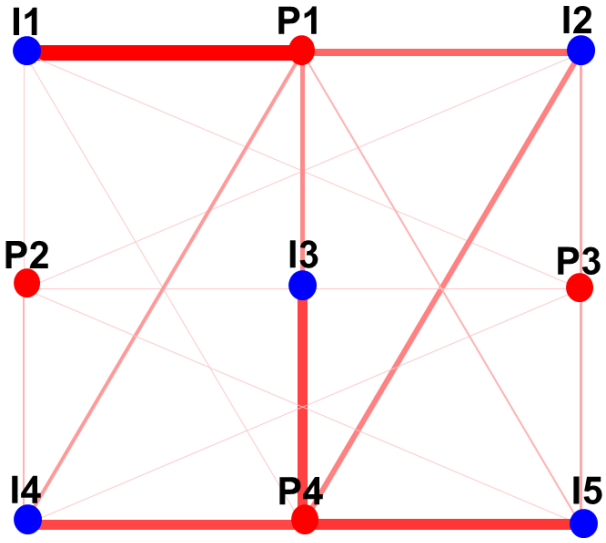
Rank	Streamline Breakthrough	Streamline Flux Allocation	Streamline Flux Allocation (Normalized)	Machine Learning	Cross Correlation
1	P1-I1	P1-I1	P1-I1	P1-I1	P1-I1
2	P4-I3	P4-I3	P4-I3	P4-I3	P2-I4
3	P1-I2	P1-I2	P4-I5	P4-I5	P4-I3
4	P4-I5	P4-I5	P1-I2	P1-I2	P3-I2

Table 9 Comparison of reservoir connectivity by different methods

Table 6 and Table 7 shows the flux allocation factor and normalized flux allocation factor averaged over all the timestep in streamline simulation. This is the true or physics-based dynamic well connectivity of the streak reservoir. The normalized flux allocation factor removes the effect of absolute injection rates in the connectivity. The high permeability streak P1-I1 and P4-I3 are captured by both flux allocation factors. Table 8 shows the minimum time of flight required by the fastest 10 % of streamline for the breakthrough. Table 9 shows a comparison of the connectivity derived from physics-based (streamline breakthrough, streamline flux allocation and normalized streamline flux allocation factor) and the connectivity from data-driven methods (cross-correlation and machine learning). The cross-correlation coefficient was not able to identify the top four connections even accounting for the time lag between the injector-producer interactions. The top 4 connectivity derived from the machine learning method matches the flux allocation factor, but not in the exact order. The order of P1-I2 (flux allocation factor 892) and P4-I5 (flux allocation factor 855) connectivity are interchanged in the machine learning method and flux allocation factor. The comparison of the normalized flux allocation factor with machine learning connectivity would be more meaningful since it removes the effect of magnitude of the injection rates. The connectivity derived from the machine learning model exactly matches the normalized streamline flux allocation factor. Hence, the connectivity of the GRU model is validated. Figure 29 shows the reservoir connectivity map from streamline and GRU model.



(a)



(b)

Figure 29 Reservoir Connectivity Map (a) Flux allocation (normalized) (b) GRU connectivity (normalized)

3.4.5 Discussion

Here we will be discussing some of the formulations of the machine learning problem that did not work for the streak reservoir case. Formulation 1, to build a machine learning model to predict the future oil production rate and future water production rate, given the future water injection rate, and the history of oil production rate, water production rate, and water injection rate. The machine learning model was able to match history, but the predictive power was not good. Formulation 2, to build a machine learning model to predict future liquid production rate given the future injection rate, and the history of liquid production rate, and water injection rate. This formulation also had similar performance; the model was able to match the history, but the predictive power was not good.

There could be several reasons for the poor performance of the model for formulation 1 and 2. Figure 30 is correlation matrix of oil production rate, water production rate and water injection rate. It shows that oil production rates for this problem are correlated with each other and water production rates are also correlated to each other. Hence, when we input them to the model, it creates a problem of multicollinearity. Multicollinearity is a problem that arises when there are highly correlated features in the training data. Formulation 2 gave good results when we removed the liquid production rate from the inputs dataset, and provides only the water injection rate as input to model, and internally the outputs from the previous timestep. Another problem could be that we are trying to predict the production rates of all producers from a single model.

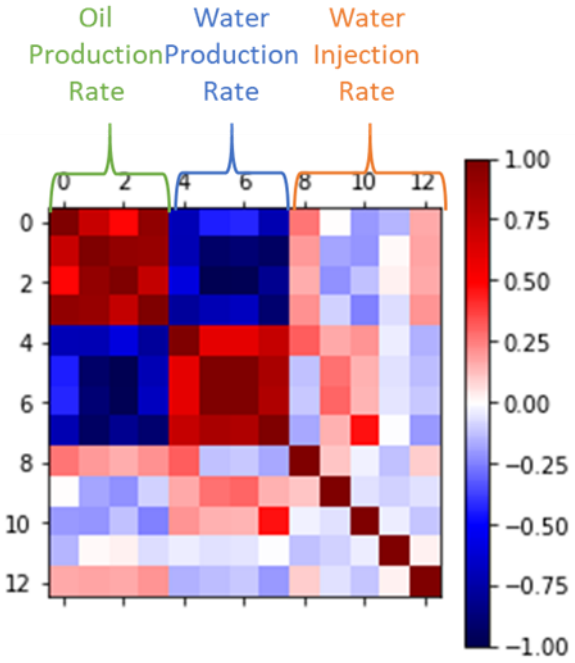


Figure 30 Correlation Matrix of oil production rate, water production rate and water injection rate

CHAPTER IV

APPLICATION TO BRUGGE RESERVOIR

4.1Model Description

For the field application of the deep learning algorithm, we use the Brugge case, a SPE 3D benchmarking case. This reservoir model was developed by the TNO (Peters et al. 2010). The structure of the field consists of a significant boundary fault. The rock properties and thickness of this model are similar to the Brent fields in the North Sea. The reservoir model consists of 139 x 48 x 9 grid blocks and is under waterflooding with 10 injectors and 20 producers. Figure 31 shows the permeability distribution of the field.

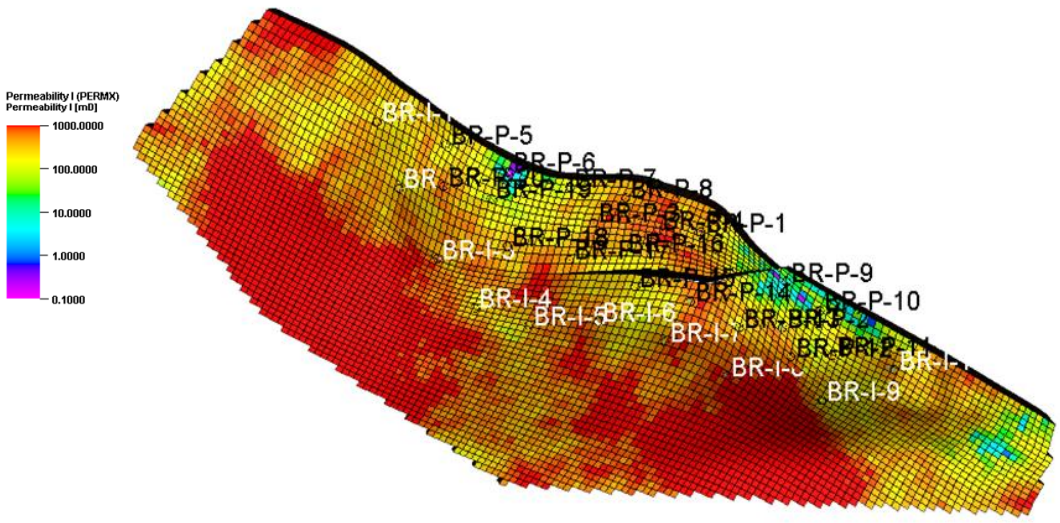


Figure 31 Permeability distribution of Brugge case

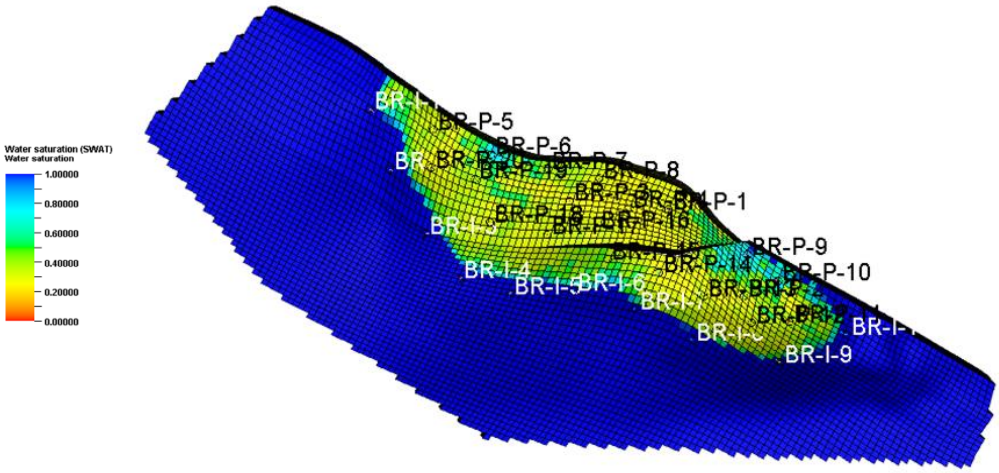


Figure 32 Initial water saturation for Brugge case

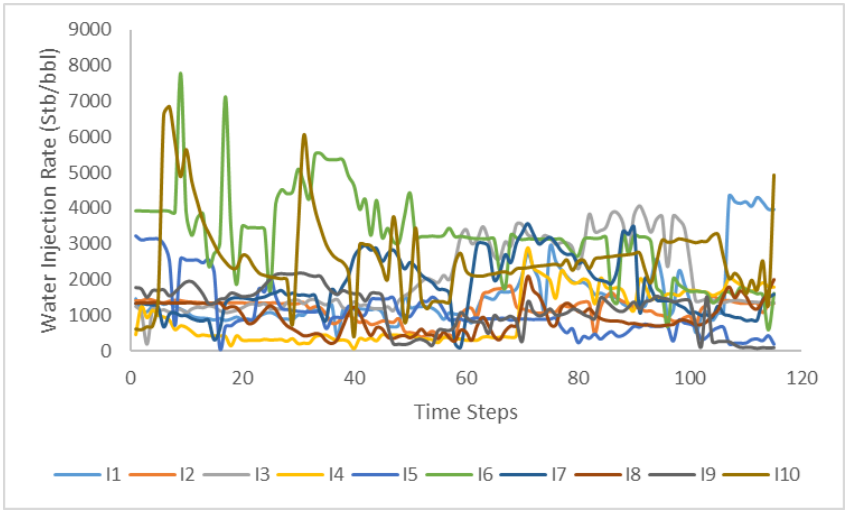


Figure 33 Injection Profile for Brugge Case

Figure 32 shows the initial water saturation for the Brugge case. A varying water injection rate is used for this case to identify producer-injector interactions. Figure 33 shows the injection profile, and Figure 34 shows the box and whisker plot of injection

rates. The box plot shows the maximum, first quartile, median, third quartile, and minimum of the injection rate for the injectors. From this, we can see that there is a significant variation in the injection rate. In Figure 33, a one-time step corresponds to 30 days, and the injection profile is for 8 years. The commercial reservoir simulator, Eclipse, is used for getting production response, and the results are shown in Appendix A. All the producers are at constant bottom hole pressure constrained (1700 psia), and the injectors are at rate constrained. The voidage replacement ratio is 0.75.

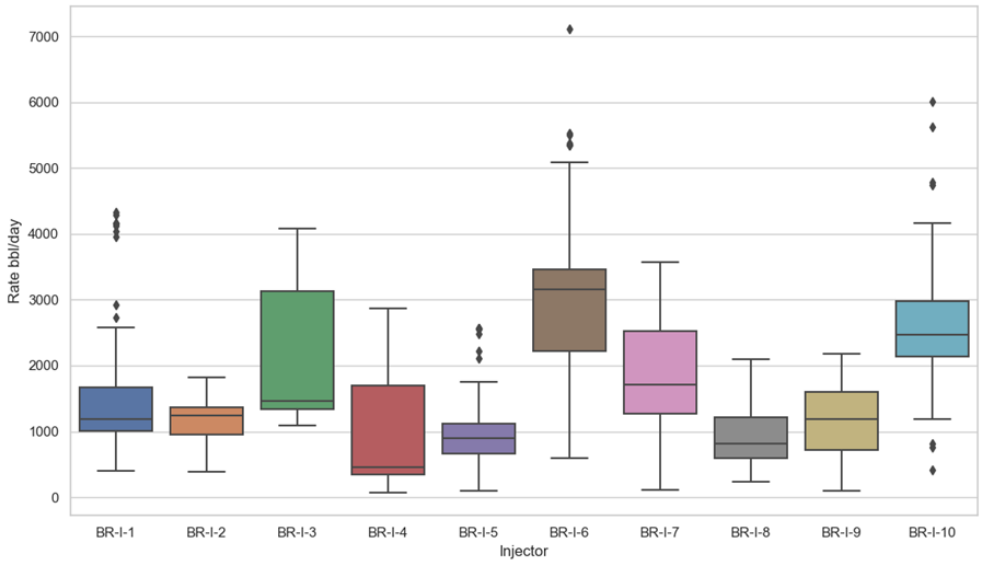


Figure 34 Box and whisker plot of injection rates for Brugge Case

4.2 Exploratory Data Analysis

The liquid production rate for all the wells follows the same trend. Hence, we select the water production response as the prediction/output of the model. Figure 35 shows the Pearson correlation coefficient for water injection rate and water production response of the well without lag time. The injectors I4, I5 and I6 are correlated to most

of the producers. The injector I4 has a strong positive correlation between P5, P11, P12, P13, P14, P15, P18, and P20. The injector I5 has a negative correlation between P1, P5, P8, P9, P11, P12, P13, P14, P15, P18, and P20. The injector I6 has a negative correlation between P5, P11, P12, P13, P14, P15, P17, P18, P19, and P20. The water injector and water production correlation do not provide any insight into connectivity. Since there are only 4 injectors, that has a strong correlation with producers and the correlation values for these injectors to different producers are almost similar. There is also a high positive correlation between producers to producers that are near the injectors (P2, P5, P10, P11, P12, P13, P14, P15, P17, P18, P19, and P20).

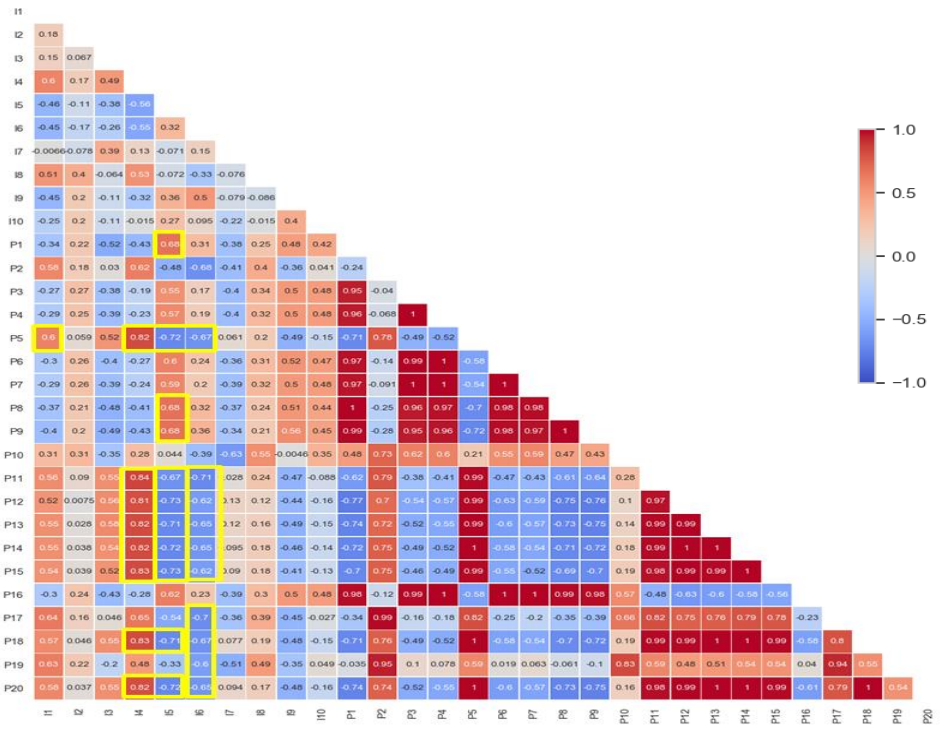


Figure 35 Correlation matrix of input and output data for Brugge Case

4.3 Training

Problem Statement: Prediction of future water production rate for all producers given the future water injection rate, and the history of water production rate, and water injection rate. The model is then used to understand injector-producer interactions.

A search radius is applied to the deep learning model to integrate the spatial information of injectors and producers. To model the production response for a producer, we include only the injectors inside the search radius. All other injectors are excluded in the modeling of the production response for that producer. Similarly, the search radius is applied to all the producers. It includes only the injectors which fall under the search radius. Figure 36 shows an illustration of search radius. A sensitivity analysis is also performed to understand the effect of the search radius to reservoir connectivities.

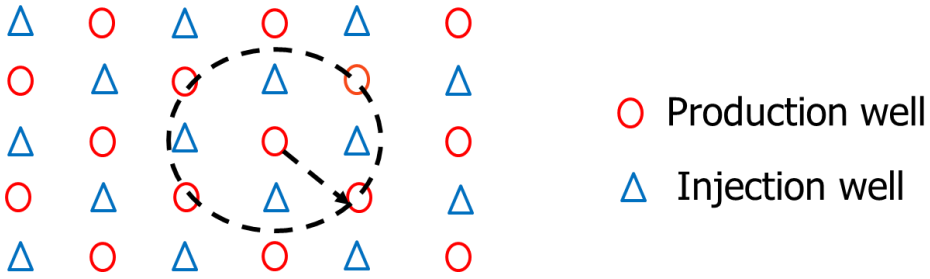


Figure 36 Illustration of search radius to a five-spot pattern

The training methodology used here is the same as in the training methodology of the synthetic case (3.3 Training) except that the search radius is introduced to limit the input features for modeling of each producer's response, and the water production rate is

used as output instead of liquid production rate. An LSTM and GRU model are trained, the input variable is the water injection rate, which is trained to match the water production rate.

4.4 Results and Discussion

4.4.1 Hyperparameter Optimization

Cell Size	GRU	LSTM
6	676.605	1121.739
8	621.27	490.632
10	459.658	410.954
12	410.105	393.635
14	780.485	377.803
16	387.444	343.529
24	338.019	254.322
32	286.992	179.88
36	264.887	192.357
40	246.113	159.46
44	203.896	103.287
48	187.695	165.914
52	199.424	116.278

Figure 37 Grid search hyperparameter optimization

Figure 37 shows the optimization of hyperparameters of LSTM and GRU. We optimized the memory or cell size of the network by the Grid Search method, and the number of epochs for the network training is optimized using early stopping criteria. For this case, while building the LSTM model in Keras(REF) we put ‘None’ in place of window size. While training, the LSTM model assumes the window size as the length of the training dataset, while prediction it uses masking and padding to get the results. The best performing model for GRU is at a cell size of 48, and the error for training is 187

bbl/day. The best performing model for LSTM is at a cell size of 44, and the error for training is 103 bbl/day. Both the GRU and LSTM models were able to history match the water production data. However, the forecasting of the model was not good. We picked the LSTM model with cell size 44 for further analysis.

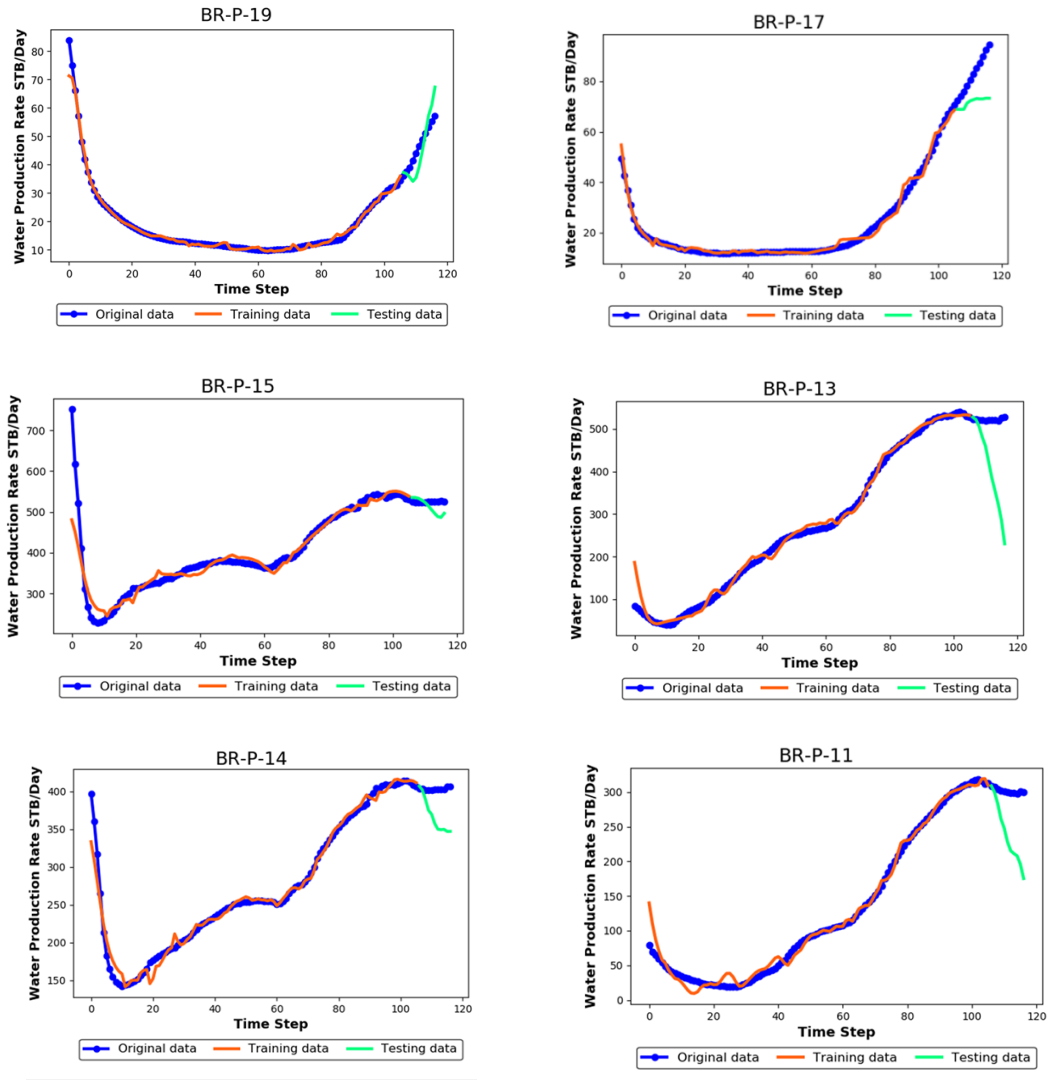
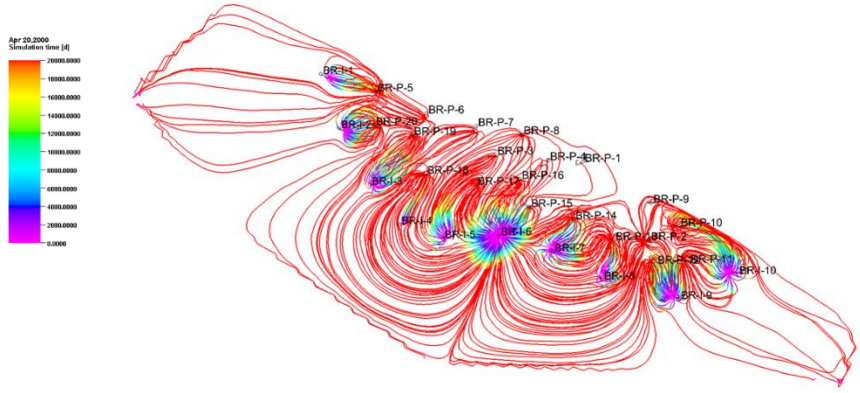


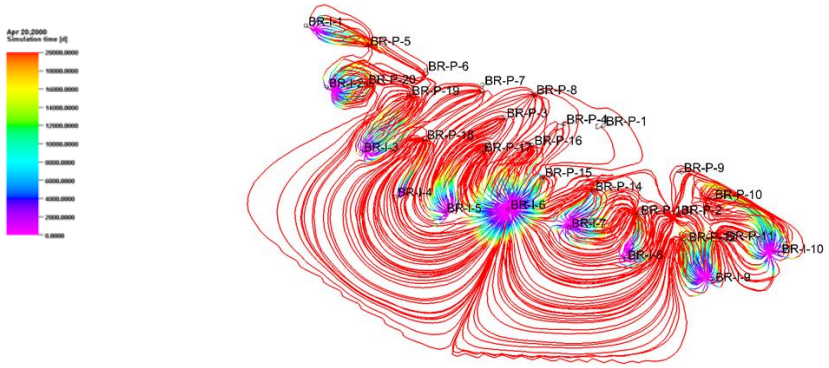
Figure 38 Training and Prediction of LSTM model for search radius of 6,000 ft

Figure 38 shows the training and forecasting of the LSTM model for a search radius of 6,000 ft for some of the producers. The results of the remaining producers are shown in Appendix B.

We try to improve the forecasting performance of the model by investigating the influence of aquifer in the model. A streamline simulation was run in Destiny (MCERI inhouse software) to visualize the underground flow.

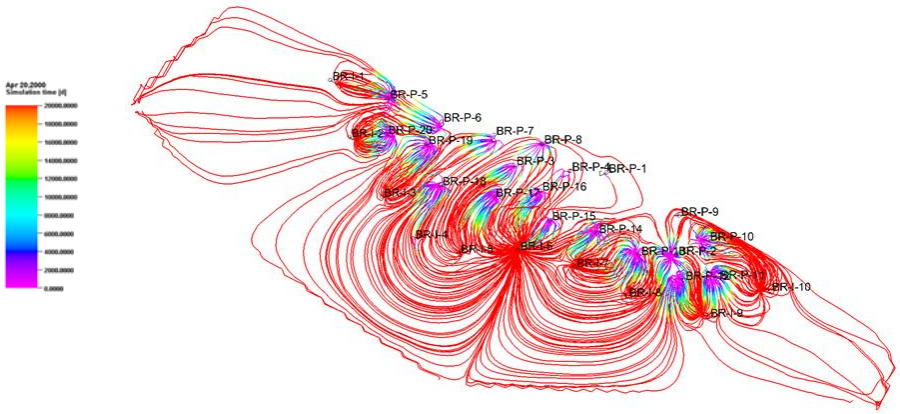


(a)

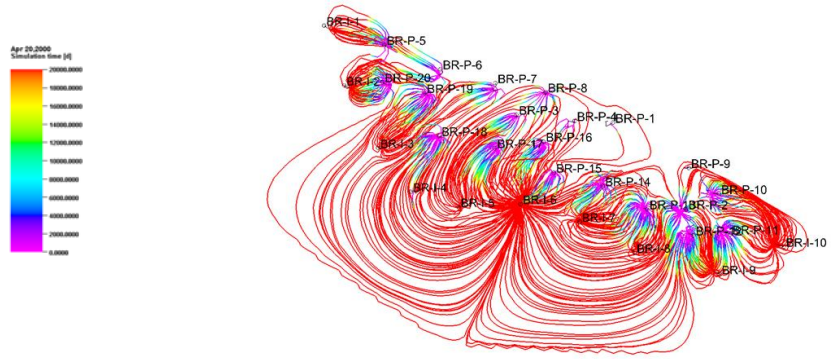


(b)

Figure 39 Time of Flight from Injector (a) with stagnation points (b) without stagnation points



(a)



(b)

Figure 40 Time of Flight from Producer (a) with stagnation point (b) without stagnation point

Figure 39 and Figure 40 shows the Time of Flight from injector and from producer, with stagnation point and without stagnation point. We can notice that only few streamlines are removed in the TOF map when we remove the stagnation points. This show that the aquifer support is week for this model.

4.4.2 Reservoir Connectivity by Feature Importance

The permutation feature importance method is applied to the trained LSTM models to understand the well-connectivity of the reservoir. For each of the trained LSTM models, we permute or shuffle the injection data (input feature) and monitor the increase in RMSE error with respect to the original base case RMSE. When an injector is permuted, if it causes an increase in the model prediction error, it is an important injector to model. If the permuted injector does not increase the model prediction error it is not important. Figure 41-43 shows the connectivity for different search radii.

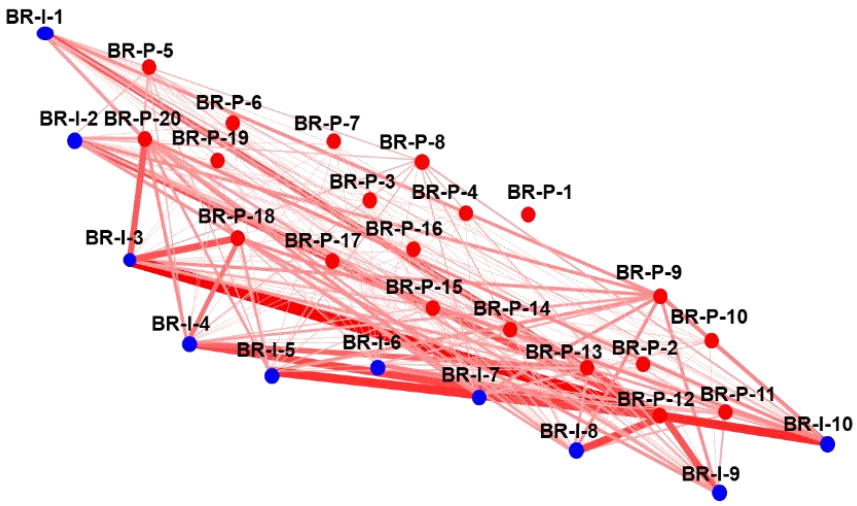


Figure 41 LSTM reservoir connectivity map (Without Search Radius)

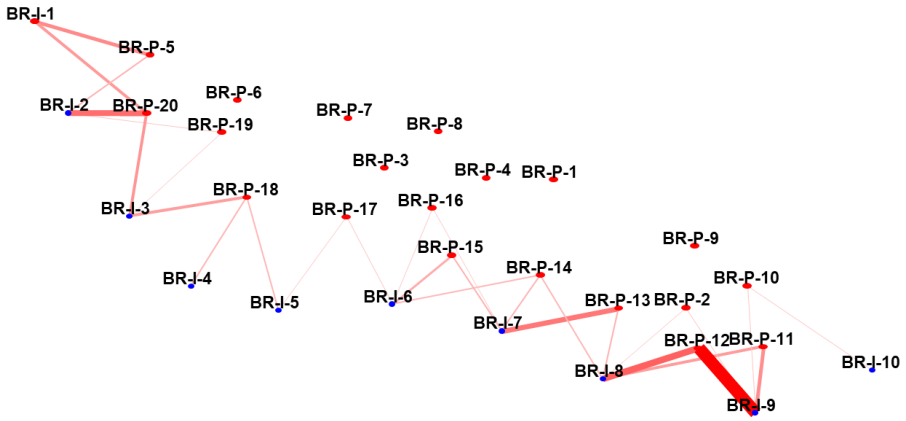


Figure 42 LSTM reservoir connectivity map (6000 ft search radius)

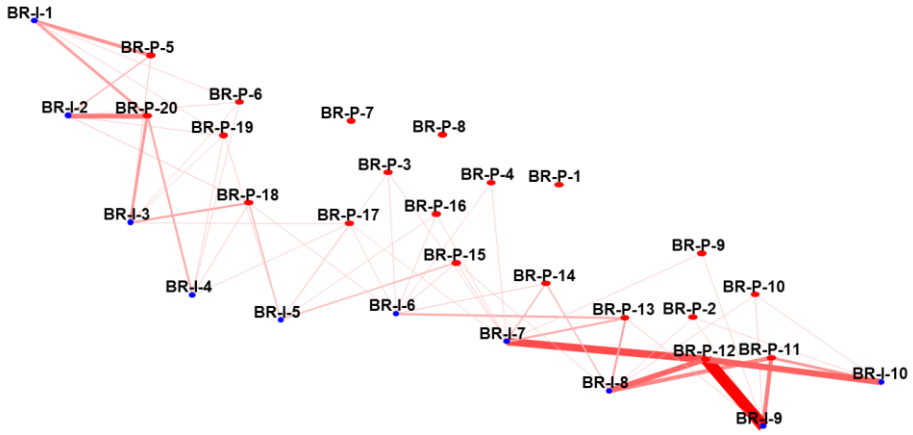


Figure 43 Reservoir connectivity of LSTM model (search radius 10000 ft)

4.4.3 Sensitivity of connectivity to search radius

Figure 44 Normalized LSTM reservoir connectivity for: no search radius, 6000 ft, and 10000 ft

Figure 44 shows the sensitivity of reservoir connectivity for different search radii. For the case without a search radius, the producer is connected to all the injectors in the field and with almost identical connectivities. This shows us the importance of using location constraints in the model. We neglect the case without a search radius for further analysis. For the case with a search radius of 6,000 ft, 13 producers were modeled, while the remaining 7 producers were not modelled because there was no injector in the 6,000 ft search radius. The producer P1 is closed during the early stage of reservoir simulation for violating the BHP constraint. For the case with a search radius of 10,000 ft, 16 producers were modeled, while remaining 4 producers were not modelled because there were no injectors in the 10,000 ft search radius. The comparison of 8 producers is shown in Figure 39; the remaining 5 producers are shown in Appendix B. The search radius 6,000 ft is selected arbitrarily for further analysis..

4.4.4 Reservoir Connectivity by Streamline Simulation

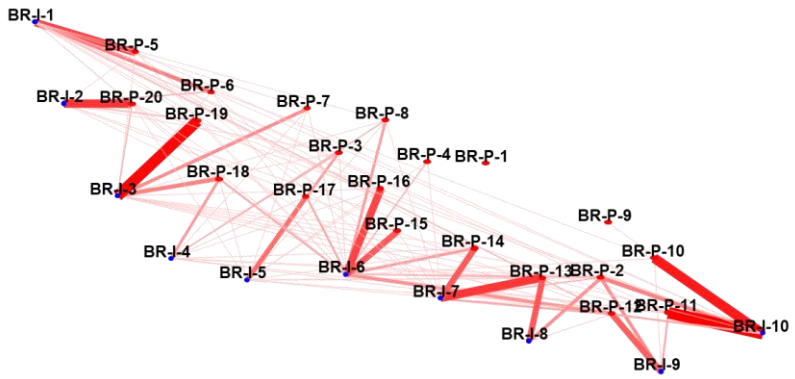


Figure 45 Streamline flow allocation factor for Brugge Case

Figure 45 shows the time-averaged streamline flux allocation factor for the reservoir.

4.4.5 Validation of reservoir connectivity

Producer	Top Connectivity		
	Streamline	DL: 6,000 ft	DL: 10,000 ft
P-2	I-10	I-9	I-9
P-5	I-1	I-1	I-1
P-10	I-10	I-10	I-10
P-11	I-10	I-9	I-9
P-12	I-9	I-9	I-9
P-13	I-7	I-7	I-8
P-14	I-7	I-7	I-7
P-15	I-6	I-6	I-5
P-16	I-6	I-6	I-6
P-17	I-5	I-6	I-5
P-18	I-3	I-3	I-3
P-19	I-3	I-3	I-3
P-20	I-2	I-2	I-2

Table 10 Reservoir connectivity comparison of Streamline and DL with search radius

Table 10 shows the connectivity comparison between the normalized streamline flux allocation factor and ML/DL model with a search radius of 6,000 ft and 10,000 ft for 13 producers. For most of the producers, the top connectivity from the streamline and ML/DL model is the same. However, for producers P-2 and P-11, the top connectivity from ML/ DL model is different from the streamline. For the producer P-2, the top connectivity from streamline is I-10, which is not included in the search radius of 6,000 ft. For P-15, the top connectivity from the streamline matches with the ML/DL for a

search radius of 6,000 ft but does not match with the search radius of 10,000 ft. For P-17, the top connectivity from streamline matches with the DL for a search radius of 10,000 ft but does not match with a search radius of 6,000 ft. Further research is needed to get all the connectivities to match with the streamline connectivities.

4.4.6 Discussion

Since the voidage replacement ratio for this case is 0.75, we have also tried a formulation which considers for the additional energy provided by the reservoir. In this formulation we have an additional input (Total production – Total Injection). Even though in real life the future total production rate would not be available to calculate the instantaneous additional energy input feature for the forecasting, we tried this case for research purpose to understand the contribution of the input feature, total production – total injection.

The forecasting results did not improve with the additional energy input feature. A case with no search radius was applied to LSTM model. Figure 46 show the forecasting results for some of the wells for LSTM model with additional energy input. The results of the remaining wells are shown in Appendix B.

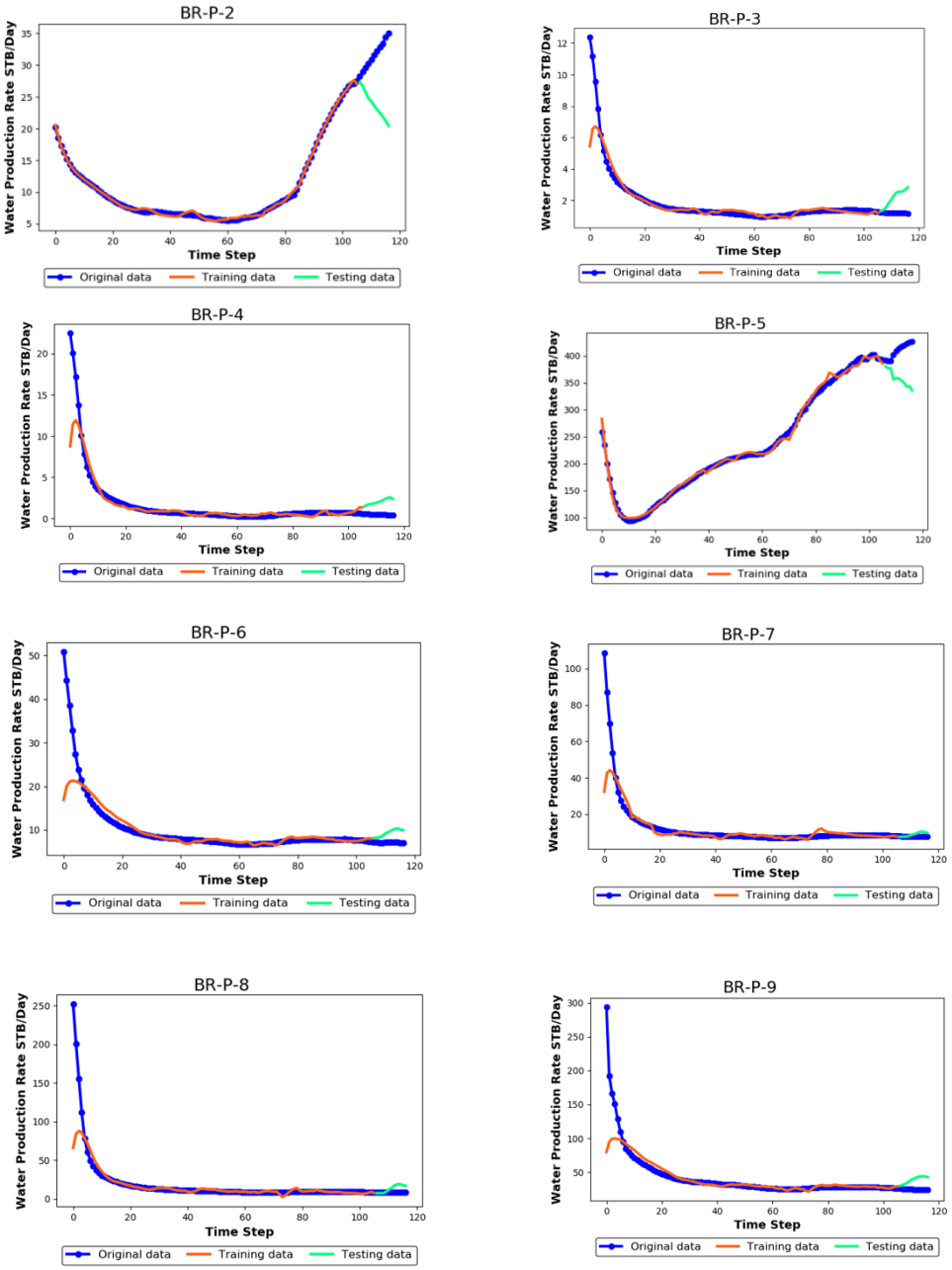


Figure 46 LSTM model prediction with additional energy input feature for no search radius

CHAPTER V

CONCLUSIONS AND RECOMMENDATION

5.1 Summary

In this research, we have applied state-of-the-art deep learning algorithms, such as GRU and LSTM, to formulate a workflow to understand the dynamic well-connectivity of producers and injectors for a reservoir under waterflooding. The GRU and LSTM model are trained to match the production response of the well. A model agnostic interpretation method is then applied to the trained model to get the producer-injector interactions. The results from the deep learning model are then validated with the physics-based model.

We have presented a training methodology for time-series production data, techniques for preventing overfitting and underfitting, and optimizing the hyperparameters. The machine learning methodology was applied to two models: a synthetic streak reservoir and a field-scale Brugge case. The permutation feature importance technique was applied to the trained model to understand reservoir-connectivity. The permutation feature importance method calculates the reservoir-connectivity by permuting or shuffling the water injection rate to the trained model and calculating the increase in the RMSE error compared to the original case. The connectivity derived from the deep learning model was validated by the normalized streamline flux allocation factor.

For synthetic reservoirs, we trained both the LSTM and GRU models. We optimized the hyperparameters, the window size and the cell size, using a Grid Search

method. The optimal GRU model outperformed the optimal LSTM model. Hence, we picked the GRU model for all the analyses. The GRU model was able to match the historical production data during training and forecast the liquid production rate accurately. The connectivity rankings derived from the deep learning model, cross-correlation coefficient, flux allocation factor, and normalized flux allocation factor were compared to each other. All the methods except the correlation coefficient were able to identify both of the high permeability streaks (P1-I1 and P4-I3) as the top two connectivities. The cross-correlation liquid production rates and time lagged injection rates were not able to identify the high permeability streak as the top connectivities. The top two connectivities in the lists from the deep learning model and the flux allocation factor matched with each other. However, the third and fourth connectivities in the lists from the deep learning model and the flux allocation factor were interchanged. This was due to the influence of the injection rate; hence we compare the connectivity from the normalized flux allocation factor and the deep learning model. The ranking of the connectivities from the deep learning model matched with the normalized flux allocation factor.

For the Brugge reservoir, we selected the water production rates as the output instead of the liquid production rate, since the liquid production rates for all the producers follow a similar pattern to each other. The signal from the injector was not visible using liquid production rate as the output. The reason could be because of the low water cut. A Grid Search method was applied to optimize the cell size of the GRU and LSTM network. We selected the LSTM model based on its performance. We have also

applied a spatial constraint or search radius to the training model, limiting the number of injectors used to model the production response for a well. The injectors that fell under the search radius were only considered for the modeling of a producer. In this training method, we trained a deep learning model for each of the producers. For the training method without a search radius, only a single deep learning model was trained for all the producers. Both the trained deep learning model with and without the search radius matched the historical production data but could not accurately predict the future water production rate. The permutation feature importance method was applied to get the reservoir connectivity. The connectivity derived from the model without a search radius shows that the weights are almost equally distributed among the injectors. This signifies the importance of spatial constraints for understanding the reservoir connectivities. The top reservoir connectivity derived from the deep learning model with a search radii of 6,000 ft and 10,000 ft was compared to the normalized flux allocation factor. The deep learning model was able to identify the top reservoir connectivity ranking for most of the wells. However, it was not able to identify connectivity ranking correctly for some wells.

5.2 Recommendation

Further research is needed to forecast the production response of the Brugge case accurately. It would be beneficial to add physics-based constraints to the deep learning model's loss function, so that we could get a better forecast of production rates. In this research, we have shown the importance of having a spatial constraint or search radius. However, further research is needed to find the optimal search radius to understanding the well-connectivity using the deep learning model. Another place of improvement

would be using a different model interpretation method and comparing its results with the permutation feature importance method. Once we calculate the well connectivity accurately, we could use it to optimize the injection rate for the field and get a better sweep efficiency.

REFERENCES

Artun, E. (2016, May 23). Characterizing Reservoir Connectivity and Forecasting Waterflood Performance Using Data-Driven and Reduced-Physics Models. Society of Petroleum Engineers. doi:10.2118/180488-MS.

Barbounis, Thanasis G., John B. Theocharis, Minas C. Alexiadis, and Petros S. Dokopoulos. "Long-term wind speed and power forecasting using local recurrent neural network models." IEEE Transactions on Energy Conversion 21, no. 1 (2006): 273-284.

Bell, F. (2019, November 7). Forecasting at Uber: An Introduction. Retrieved June 1, 2020, from https://eng.uber.com/forecasting-introduction/.

Ben, Y., Perrotte, M., Ezzatabadipour, M., Ali, I., Sankaran, S., Harlin, C., & Cao, D. (2020, January 28). Real-Time Hydraulic Fracturing Pressure Prediction with Machine Learning. Society of Petroleum Engineers. doi:10.2118/199699-MS.

Breiman, L. (2001). Random forests. Machine learning, 45(1), 5-32.

Brownlee, J. (2019, October 3). Use Early Stopping to Halt the Training of Neural Networks At the Right Time. Retrieved June 1, 2020, from https://machinelearningmastery.com/how-to-stop-training-deep-neural-networks-at-the-right-time-using-early-stopping/.

Cao, Q., Banerjee, R., Gupta, S., Li, J., Zhou, W., & Jeyachandra, B. (2016, June 1). Data Driven Production Forecasting Using Machine Learning. Society of Petroleum Engineers. doi:10.2118/180984-MS.

Cardoso, M. A., & Durlofsky, L. J. (2010, June 1). Use of Reduced-Order Modeling Procedures for Production Optimization. Society of Petroleum Engineers. doi:10.2118/119057-PA

Carpenter, C. (2019, March 1). Machine-Learning Approach Identifies Wolfcamp Reservoirs. Society of Petroleum Engineers. doi:10.2118/0319-0087-JPT.

Chen, Y. Y., Lin, Y. H., Kung, C. C., Chung, M. H., & Yen, I. (2019). Design and Implementation of Cloud Analytics-Assisted Smart Power Meters Considering Advanced Artificial Intelligence as Edge Analytics in Demand-Side Management for Smart Homes. Sensors, 19(9), 2047.

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.

Chollet, F. (2018). Deep learning with Python.

Efendiev, Y., & Hou, T. Y. (2009). Multiscale finite element methods: theory and applications (Vol. 4). Springer Science & Business Media.

Elichev, V., Bilogan, A., Litvinenko, K., Khabibullin, R., Alferov, A., & Vodopyan, A. (2019, October 22). Understanding Well Events with Machine Learning. Society of Petroleum Engineers. doi:10.2118/196861-MS.

Fisher, A., Rudin, C., & Dominici, F. (2019). All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously. Journal of Machine Learning Research, 20(177), 1-81.

Gaganis, V., & Varotsis, N. (2012, January 1). Machine Learning Methods to Speed up Compositional Reservoir Simulation. Society of Petroleum Engineers. doi:10.2118/154505-MS.

Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. O'Reilly Media.

Glorot, X., & Bengio, Y. (2010, March). Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the thirteenth international conference on artificial intelligence and statistics (pp. 249-256).

Goodfellow, I., Bengio, Y., Courville, A. and Bengio, Y., 2016. Deep learning (Vol. 1). Cambridge: MIT press.

Guo, Z., & Reynolds, A. C. (2019, March 29). INSIM-FT-3D: A Three-Dimensional Data-Driven Model for History Matching and Waterflooding Optimization. Society of Petroleum Engineers. doi:10.2118/193841-MS.

Haroon, S., Viswanathan, A., & Shenoy, R. (2018, November 12). From Insight to Foresight: Knowing How to Apply Artificial Intelligence in the Oil & Gas Industry. Society of Petroleum Engineers. doi:10.2118/192629-MS.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

Hsieh, Tsung-Jung, Hsiao-Fen Hsiao, and Wei-Chang Yeh. "Forecasting stock markets using wavelet transforms and recurrent neural networks: An integrated system based on artificial bee colony algorithm." Applied soft computing 11, no. 2 (2011): 2510-2525.

Jansen Van Rensburg, N., Kamin, L., & Davis, S. (Robert). (2019, November 11). Using Machine Learning-Based Predictive Models to Enable Preventative Maintenance and Prevent ESP Downtime. Society of Petroleum Engineers. doi:10.2118/197146-MS.

King, M.J., K.S.Burn, Pengju Wang, et al. 2005. Optimal Layer coarsening of 3D Reservoir Models for Flow Simulation. Paper SPE 96258 presented at SPE Annual Technical Conference and Exhibition, Dallas, Texas, USA, 9-12 October.

Kızrak, A. (2020, January 7). Comparison of Activation Functions for Deep Neural Networks. Retrieved June 1, 2020, from https://towardsdatascience.com/comparison-of-activation-functions-for-deep-neural-networks-706ac4284c8a.

Kuhn, M., & Johnson, K. (2013). Applied predictive modeling (Vol. 26). New York: Springer.

LeCun, Y., Bengio, Y. and Hinton, G., 2015. Deep learning. nature, 521(7553), p.436.

Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions. In Advances in neural information processing systems (pp. 4765-4774).

Maniar, H., Ryali, S., Kulkarni, M. S., & Abubakar, A. (2018, November 30). Machine-learning methods in geoscience. Society of Exploration Geophysicists.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics, 5(4), 115-133

Molnar, Christoph. “Interpretable Machine Learning.” 5.5 Permutation Feature Importance, 17 Dec. 2019, christophm.github.io/interpretable-ml-book/feature-importance.html.

Ng, Andrew. “Why Deep Representations? - Deep Neural Networks.” Coursera, 2017, www.coursera.org/learn/neural-networks-deep-learning/lecture/rz9xJ/why-deep-representations.

Olah, Christopher. "Understanding LSTM Networks." Understanding LSTM Networks -- Colah's Blog. August 27, 2015.http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Olden, J. D., & Jackson, D. A. (2002). Illuminating the “black box”: a randomization approach for understanding variable contributions in artificial neural networks. Ecological modelling, 154(1-2), 135-150.

Omrani, P. S., Dobrovolschi, I., Loh, K., & Belfroid, S. P. C. (2019, December 6). Slugging Monitoring and Classification with Machine Learning. BHR Group.

Parshall, J. (2018, July 1). Machine Learning Can Enhance and Add Jobs. Society of Petroleum Engineers. doi:10.2118/0718-0035-JPT.

Peters, L., Arts, R., Brouwer, G., Geel, C., Cullick, S., Lorentzen, R. J., … Reynolds, A. (2010, June 1). Results of the Brugge Benchmark Study for Flooding Optimization and History Matching. Society of Petroleum Engineers. doi:10.2118/119094-PA.

Pollock, J., Stoecker-Sylvia, Z., Veedu, V., Panchal, N., & Elshahawi, H. (2018, April 30). Machine Learning for Improved Directional Drilling. Offshore Technology Conference. doi:10.4043/28633-MS.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016, August). " Why should i trust you?" Explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining (pp. 1135-1144).

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. nature, 323(6088), 533-536.

Sagheer, A., & Kotb, M. (2019). Time series forecasting of petroleum production using deep LSTM recurrent networks. Neurocomputing, 323, 203-213.

Salian, Isha. “NVIDIA Blog: Supervised Vs. Unsupervised Learning.” The Official NVIDIA Blog, 20 Aug. 2019, blogs.nvidia.com/blog/2018/08/02/supervised-unsupervised-learning/.

Sayarpour, M. (2008). Development and application of capacitance-resistive models to water/CO_2 floods.

Sayarpour, M., Zuluaga, E., Kabir, C. S., & Lake, L. W. (2009). The use of capacitance–resistance models for rapid estimation of waterflood performance and optimization. Journal of Petroleum Science and Engineering, 69(3-4), 227-238.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. Neural networks, 61, 85-117.

Sneed, J. (2017, July 24). Predicting ESP Lifespan With Machine Learning. Unconventional Resources Technology Conference. doi:10.15530/URTEC-2017-2669988.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1), 1929-1958.

Vu-Quoc, Loc. “Neuron3.Png.” English Wikipedia, 16 Sept. 2018, https://commons.wikimedia.org/w/index.php?curid=72816083. CC BY-SA (https://creativecommons.org/licenses/by-sa/3.0).

Yang, R. (2019, November 16). Omphalos, Uber's Parallel and Language-Extensible Time Series Backtesting Tool. Retrieved June 1, 2020, from https://eng.uber.com/omphalos/.

APPENDIX A

SIMULATION RESULTS OF BRUGGE CASE

The commercial reservoir simulator Eclipse is used for getting the production response. Figure 47-50 show the water production rate, oil production rate, liquid production rate and water cut for Brugge case.

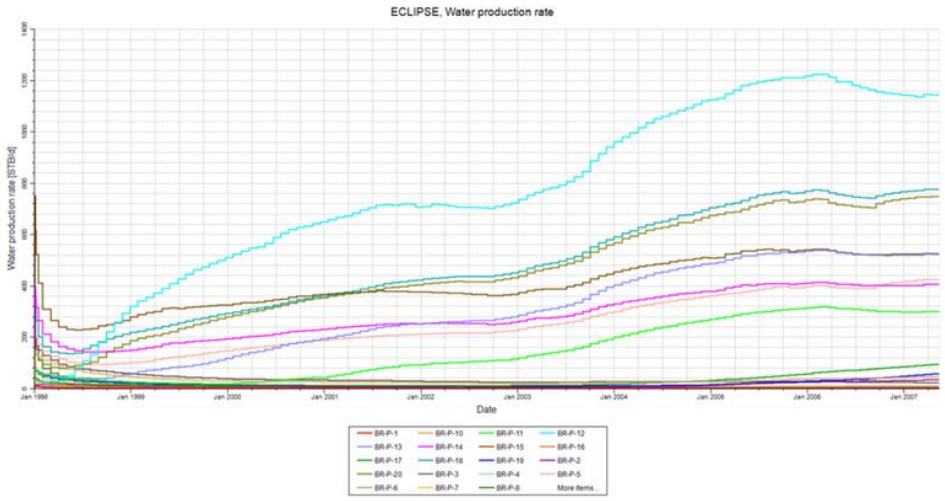


Figure 47 Water Production Rate of Brugge Case

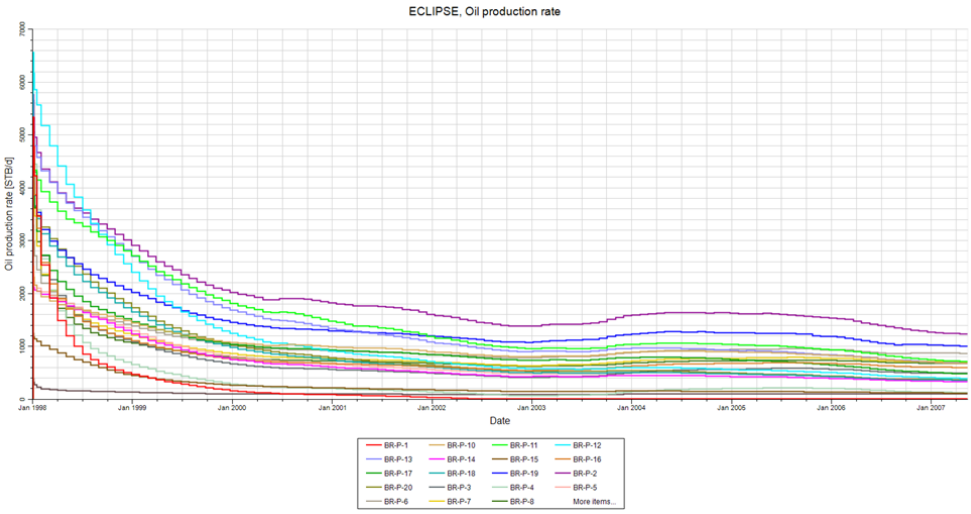


Figure 48 Oil Production Rate of Brugge Case

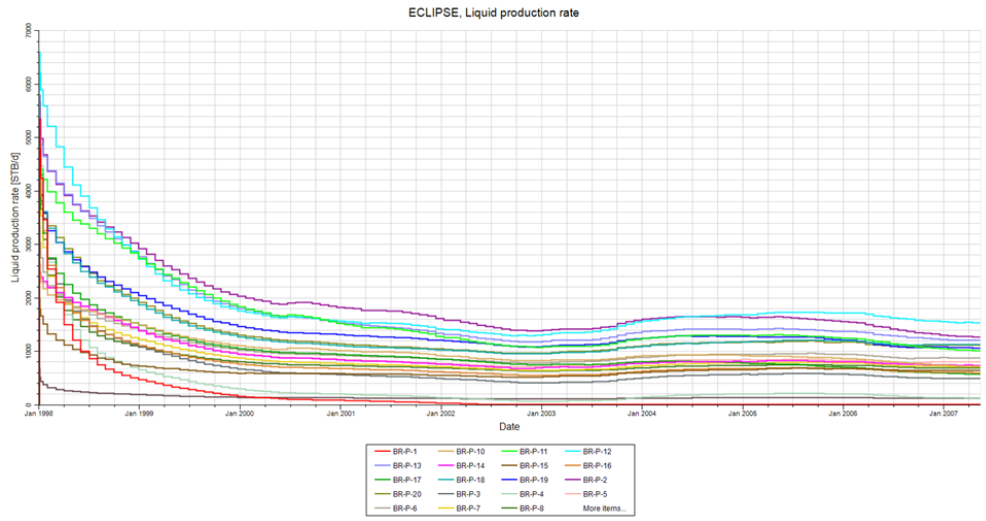


Figure 49 Liquid Production Rate of Brugge Case

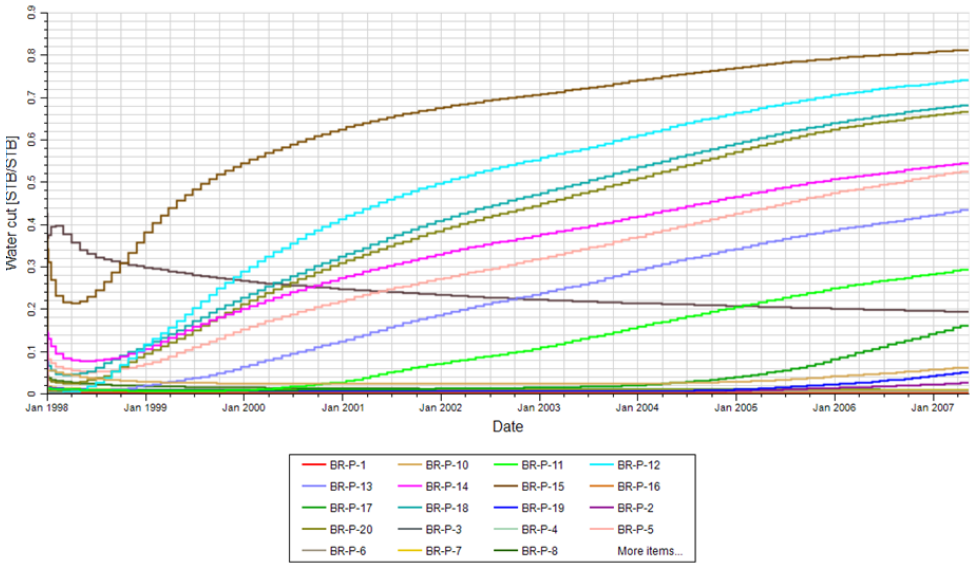


Figure 50 Plot of water cut for Brugge case

APPENDIX B

BRUGGE CASE LSTM MODEL RESULTS

Figure 51 shows the sensitivity of reservoir connectivities for different search radii.

Figure 51 Normalized LSTM reservoir connectivity for: no search radius, 6000 ft, and 10000 ft (Well P16 – Well P20)

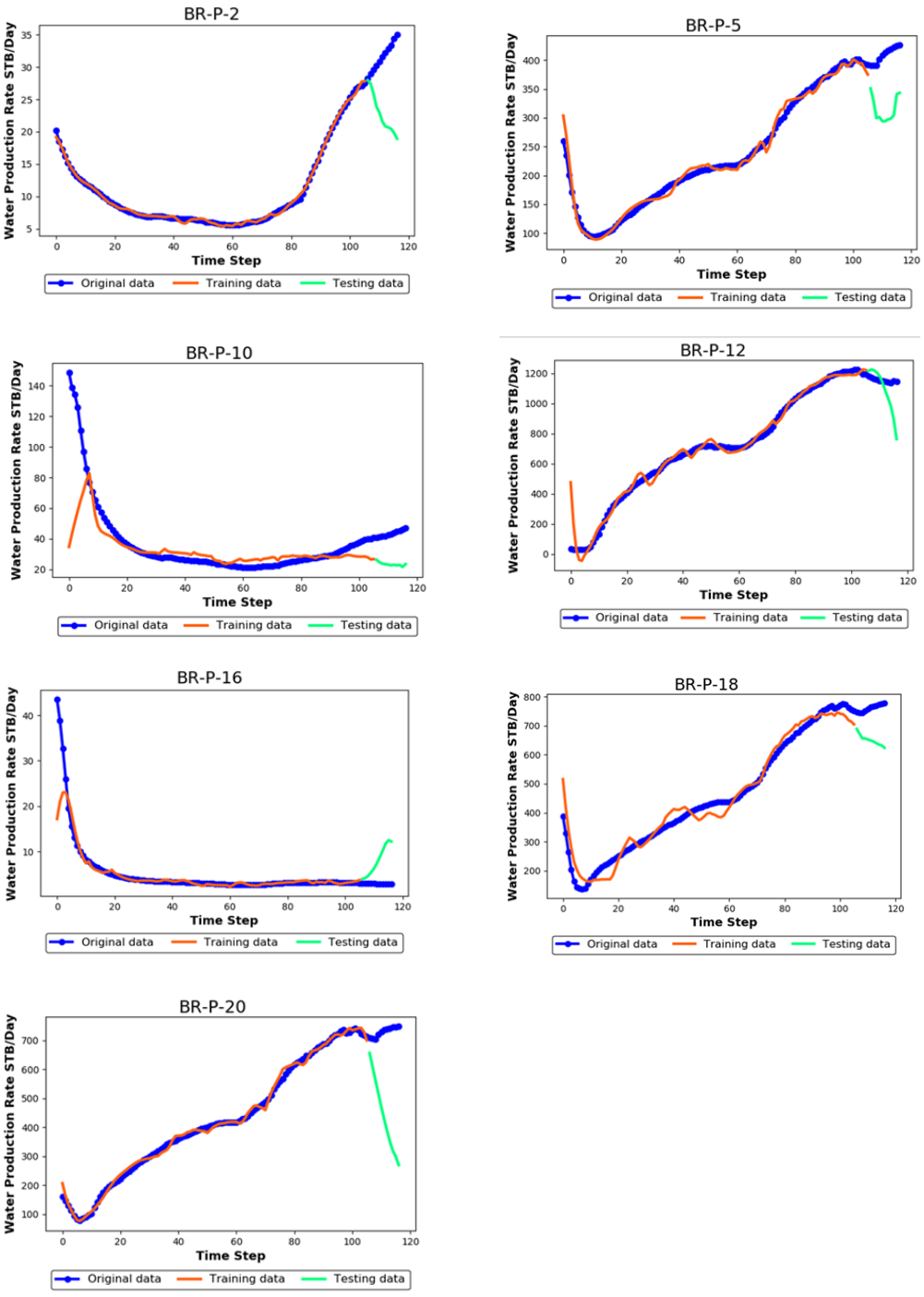


Figure 52 Training and Prediction of LSTM for Search radius of 6,000 ft (Well P2, Well P5, Well P10, Well P12, Well P16, Well P18 and Well P20)

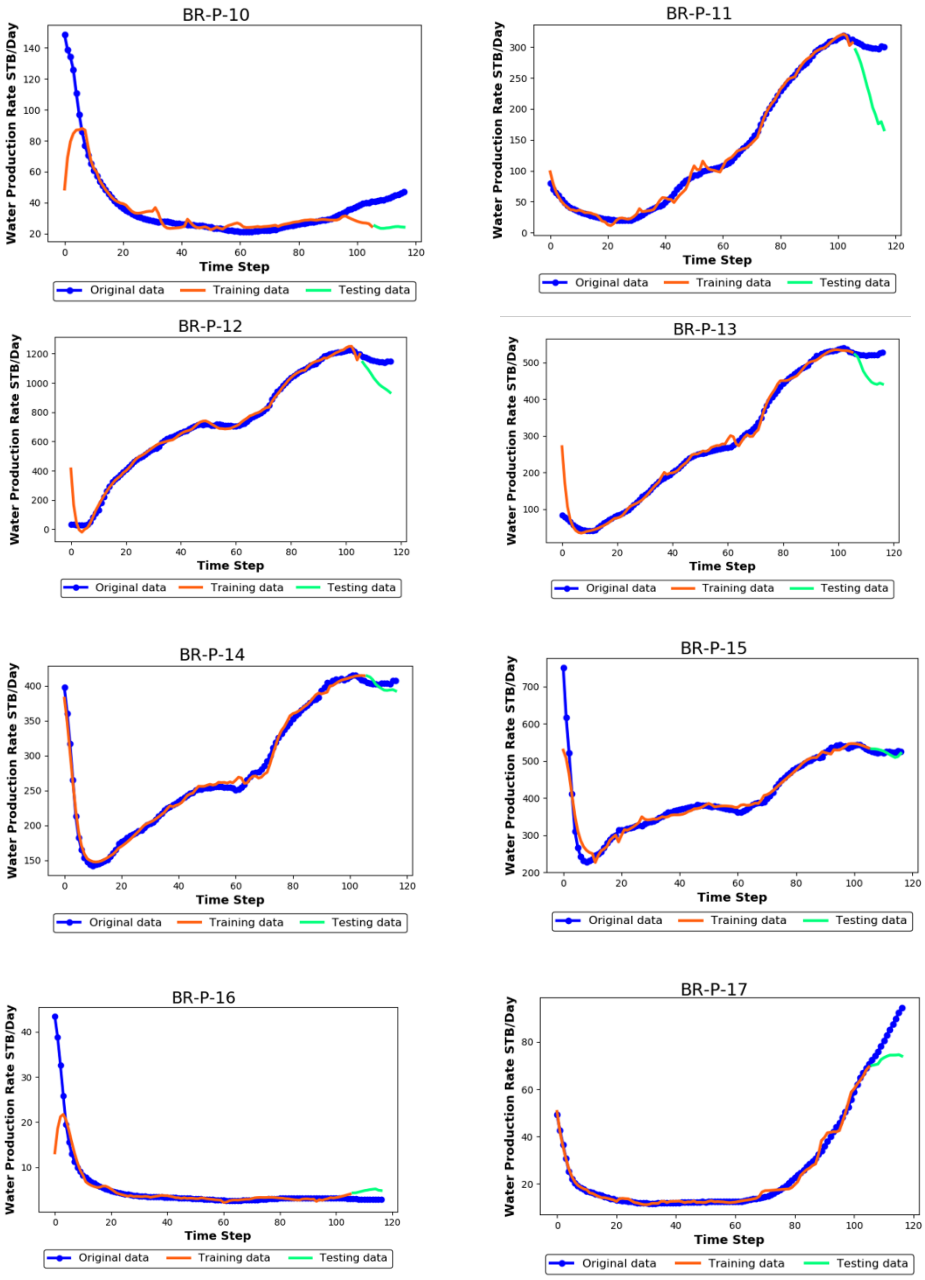


Figure 53 LSTM model prediction with additional energy input feature for no search radius (Well P10 – Well P17)

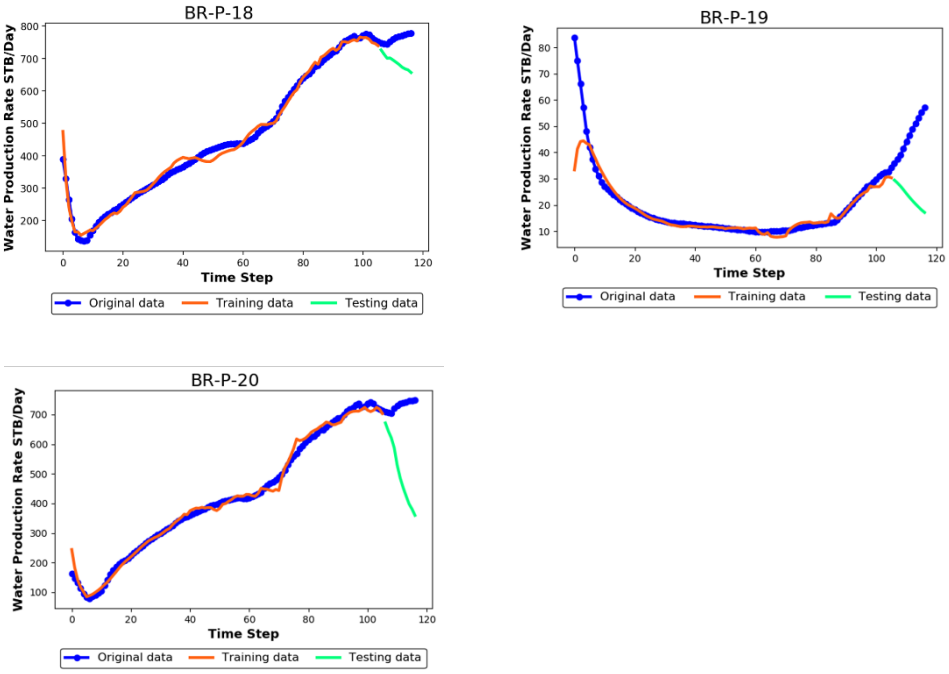


Figure 54 LSTM model prediction with additional energy input feature for no search radius (Well P18 – Well P20)

APPENDIX C

BACKPROPAGATION

Backpropagation is an efficeint algorithm for calcaulting the partial derivatives of the cost function with respect to the model parameters. In this section we will be giving an illustration of the backpropagation of error to calculate the partial derivatives for neural network shown in Figure 55.

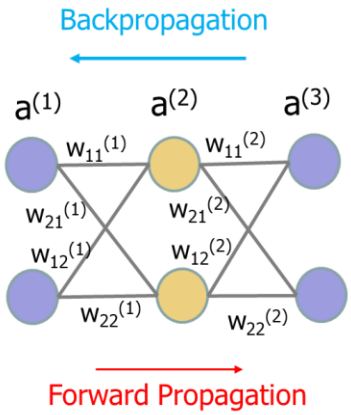


Figure 55 Neural Network forward propagation and backward propagation

Forward Propagation:

$a^{(1)} = x$ (Input) (20)

$z^{(2)} = w^{(1)} a^{(1)} + b^{(1)}$ (21)

$a^{(2)} = g(z^{(2)})$ (22)

$z^{(3)} = w^{(2)} a^{(2)} + b^{(2)}$ (23)

$a^{(3)} = g(z^{(3)})$ (Output) (24)

Backward propagation:

$$\frac{\partial C_o}{\partial w_{11}^{(2)}} = \frac{\partial C_o}{\partial a_1^{(3)}} \frac{\partial a_1^{(3)}}{\partial z_1^{(3)}} \frac{\partial z_1^{(3)}}{\partial w_{11}^{(2)}} \tag{25}$$

Where $C_o(X, w) = \frac{1}{2N} \sum_{i=1}^{N} (\hat{y_i} - y_i)^2$

$$\frac{\partial Co}{\partial a_1^3} = \left(a_1^{(3)} - y_1\right) \tag{26}$$

$$\frac{\partial a_1^{(3)}}{\partial z_1^{(3)}} = \sigma'(z_1^{(3)}) \tag{27}$$

$$\frac{\partial z_1^{(3)}}{\partial w_{11}^{(2)}} = a_1^{(2)} \tag{28}$$

$$\frac{\partial C_o}{\partial w_{11}^{(2)}} = \left(a_1^{(3)} - y_1\right) \sigma'(z_1^{(3)}) a_1^{(2)} \tag{29}$$

Similarly, we can calculate $\frac{\partial C_o}{\partial w_{12}^{(2)}}$, $\frac{\partial C_o}{\partial w_{22}^{(2)}}$ and $\frac{\partial C_o}{\partial w_{21}^{(2)}}$

$$\frac{\partial C_o}{\partial w_{11}^{(1)}} = \frac{\partial C_o}{\partial a_1^{(3)}} \frac{\partial a_1^{(3)}}{\partial z_1^{(3)}} \frac{\partial z_1^{(3)}}{\partial a_1^{(2)}} \frac{\partial a_1^{(2)}}{\partial z_1^{(2)}} \frac{\partial z_1^{(2)}}{\partial w_{11}^{(1)}} + \frac{\partial C_o}{\partial a_2^{(3)}} \frac{\partial a_2^{(3)}}{\partial z_2^{(3)}} \frac{\partial z_2^{(3)}}{\partial a_1^{(2)}} \frac{\partial a_1^{(2)}}{\partial z_1^{(2)}} \frac{\partial z_1^{(2)}}{\partial w_{11}^{(1)}} \tag{30}$$

Similarly, we can calculate $\frac{\partial C_o}{\partial w_{12}^{(1)}}$, $\frac{\partial C_o}{\partial w_{22}^{(1)}}$ and $\frac{\partial C_o}{\partial w_{21}^{(1)}}$

APPENDIX D

RANDOM FOREST IMPUTAION

It is common to have missing data points in the production data dues to many reasons. We need to fill the missing points before applying algorithms. One of the popular techniques to impute the data is by using Random Forest.

In random forest, after a tree is built, all the data (training and out of bag data) are run down the tree and the proximities are computed for each case (Brieman 2001). It is represented in matrix of NxN (N is the number of samples). For a pair of observation, I and J, if they end up in the same terminal node, then we increase the proximity by one. Continue this process for all the trees. At the end Proximity Matrix is normalized by the number of trees

The workflow for random forest imputation is to make an initial guess and gradually refine the guess in iterative way until it is good. (Brieman 2001)

1. Initial guess of median or mean
2. Build a RF
3. Run all the data in all the trees
4. Calculate the proximity matrix
5. Use proximities to calculate weighted average
6. Update the guess
7. Repeat step2-6 until it converges

The random forest imputation is illustrated using a simple example.

	Person	Age	Feature 2	Height >55"
1	Alice	14	A	TRUE
2	Bob	-	B	TRUE
3	Carol	10	C	TRUE
4	Dave	8	D	FALSE

Table 11 Data for RF imputation

Table 55 show the example data for RF imputation. The age of Bob is missing.

	Person	Age	Feature 2	Height >55"
1	Alice	14	A	TRUE
2	Bob	10	B	TRUE
3	Carol	10	C	TRUE
4	Dave	8	D	FALSE

Table 12 Initial guess of missing value

Step 1 is to guess the initial missing value. It can be mean, median or mode.

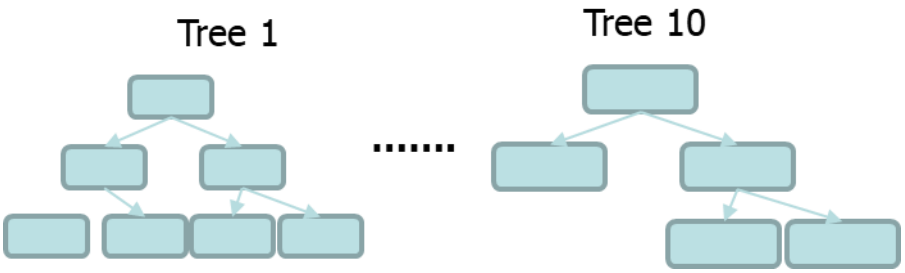


Figure 56 RF model

Step 2 is to build a RF forest model. Here we have 10 trees.

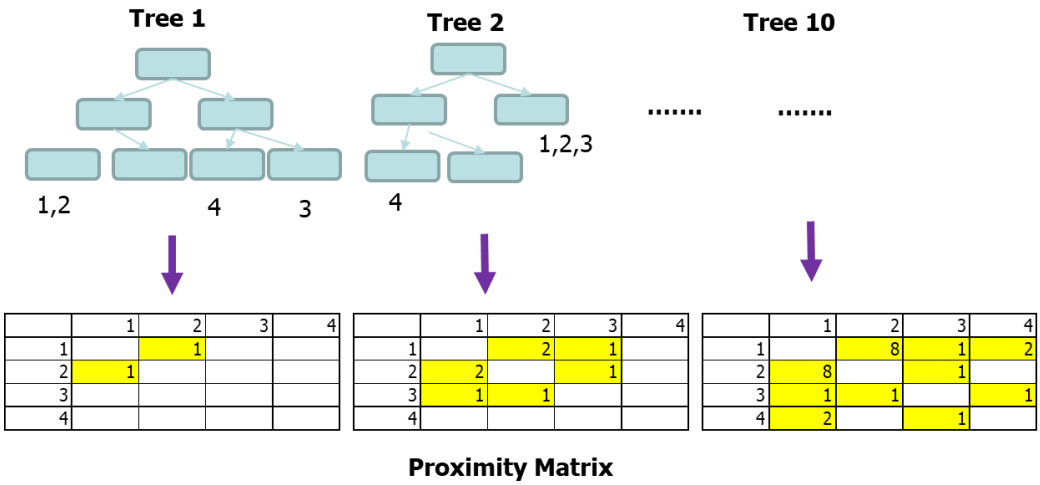


Figure 57 Process of calculating Proximity Matrix

Step 3 and Step 4 is to run all the data for all the tree in the RF model and calculate the proximity matrix. The proximity matrix is based on the number of appearances in the leaf node. For the Tree 1, 1 (Alice) and 2 (Bob) ends up in the same leaf node so are related and we record this by adding one to location of 1 and 2 in the proximity matrix. Similarly, for the Tree 2, 1,2 and 3 (Carol) are related and we add plus one to location corresponding to 1,2 and 3. This process is continued for all the Trees

	1	2	3	4
1		0.8	0.1	0.2
2	0.8		0.1	
3	0.1	0.1		0.1
4	0.2		0.1	

Table 13 Proximity Matrix

The proximity matrix is normalized by the number of trees.

Fifth step, is to guess the age of Bob with the proximity matrix

$$\text{Age of bob} = 14 * \left(\frac{0.8}{0.8+0.1}\right) + 10 * \left(\frac{0.1}{0.8+0.1}\right) = 13.55 \approx 14 \quad (31)$$

Sixth step is to update the initial guess with 14 and continue step 2-6 until it converges

APPENDIX E

RECEIVER OPERATING CHARACTERISTIC (ROC) CURVE

It is a trade-off plot of True Positive Rate (TPR) (Benefits) and False Positive Rate (FPR) (Costs) for all possible cut-off values. It can be used to determine the optimum cut-off value. The closer the graph to the left-hand border, the more accurate the model. Area Under Curve (AUC) represents the accuracy of the model. In Figure 58, Blue Curve is more accurate than the Red Curve. The Diagonal line serves as the reference line with AUC of 0.5.

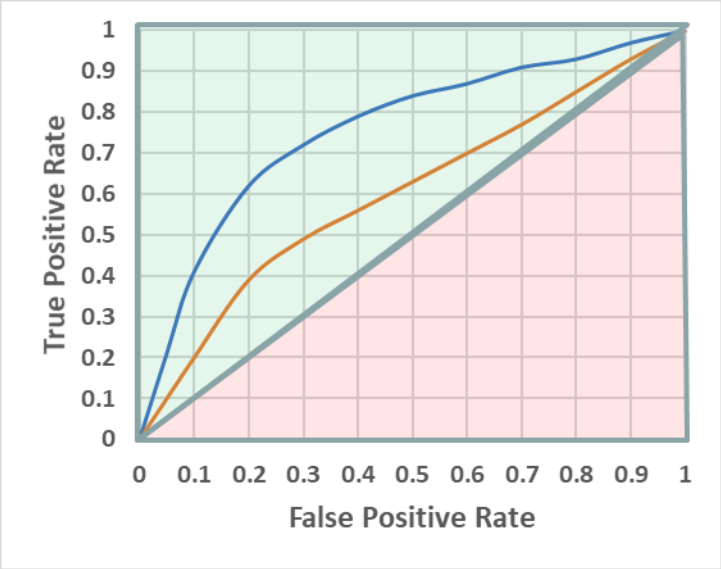


Figure 58 ROC curve

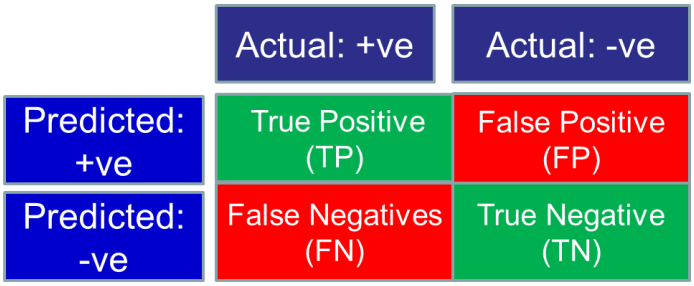


Figure 59 Confusion Matrix

Figure 59 shows the Confusion Matrix. It is a table describing the performance of the classification model. The TPR or Sensitivity measures the actual positive that are correctly identified. Specificity Measure the actual negatives that are correctly identified. ROC is a plot of TPR vs. FPR

$$TPR\ or\ Sensitivity = \frac{TP}{TP+FN} \tag{32}$$

$$Specificity = \frac{TP}{TP+FN} \tag{33}$$

$$FPR\ =\ (1 -\ specificity) \tag{34}$$

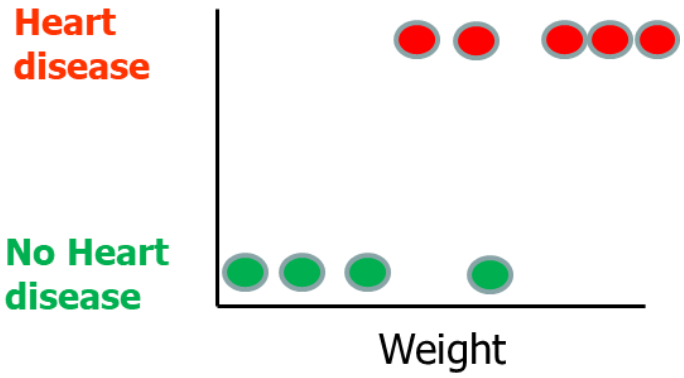


Figure 60 Heart disease dataset for classification

The ROC curve is illustrated with an example dataset of heart disease with weight.

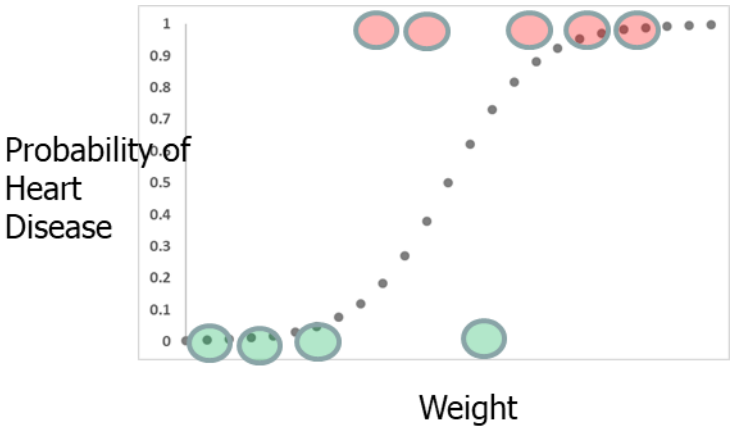


Figure 61 Logistic regression for classification of heart disease

Step 1, to fit a logistic regression for the given heart disease dataset

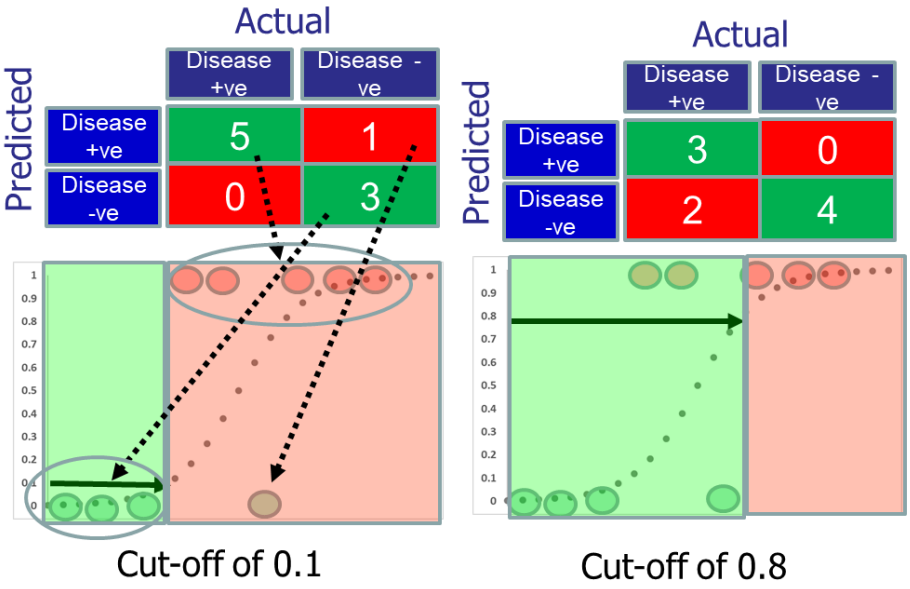


Figure 62 Applying various Cut-off and Calculating Confusion Matrix

Step 2 is applying various probability cutoff from 0 to 1 and calculating the confusion matrix. The TPR and FPR is calculated from the confusion matrix for each of the cutoff value.

Step 3 is plotting the TPR and FPR values to get the ROC curve

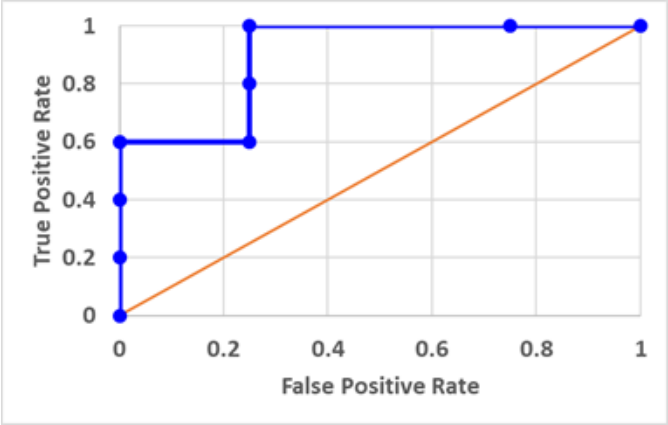


Figure 63 ROC Curve for Heart disease classification