Master's thesis

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

Mikael Yuan Estuariwinarno

A Machine Learning Approach for Net Pay Estimation from Borehole Image Logs

Master's thesis in Petroleum Geosciences
Supervisor: Carl Fredrik Berg
June 2020

NTNU
Norwegian University of Science and Technology

Mikael Yuan Estuariwinarno

A Machine Learning Approach for Net Pay Estimation from Borehole Image Logs

Master's thesis in Petroleum Geosciences
Supervisor: Carl Fredrik Berg
June 2020

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

NTNU
Norwegian University of
Science and Technology

To my wife Theresia Bhekti and my daughter Genoveva Ratih, who perpetually support me through the highs and lows. Soon we will be reunited.

Summary

In the realm of hydrocarbon exploration, net pay estimation is essential for the economic calculation and valuation of the field. By definition, net pay is the reservoir interval with economically producible hydrocarbons content. The traditional net pay estimation method involves an interpretation of borehole image logs with a cutoff derived from core data. A cutoff-dependent net pay estimation might inhibit further study of a particular field as the cutoff itself might not be universally reliable for wells or reservoir intervals without core data. In this study, an alternative net pay estimation method based on the implementation of machine learning on borehole image logs interpretation was considered and optimized. The cutoff-independent estimation is achieved through the use of a convolutional neural network (CNN). The convolutional neural network is trained to generate a model which maps the correlation between borehole image logs and net pay fraction from core ultraviolet (UV) photographs.

A convolutional neural network design and data processing procedures from a previous study by the same author were optimized with several optimization approaches, to improve the net pay estimation from borehole image logs. Image logs processing attempted to apply the necessary corrections for an improvement in the image quality. Core UV photographs processing aimed to convert core UV photographs into binarized images from which net pay fraction values were derived. The optimized data processing procedures resulted in several image log datasets and net pay fraction labels based on data from four exploration wells in the Johan Castberg field. A set of net pay estimations with the processed data and optimized CNN were conducted to evaluate the optimized net pay estimation method.

The estimation results showed that the net pay estimation method was able to generate accurate and reliable estimations on unseen image logs. Evaluation of the estimation results proved that the optimization of the net pay estimation method was able to generate a model with more accurate estimations compared to the earlier study. The results also showed that the net pay estimation method has the potential to be implemented in a situation where core information was not available or not reliable. Several limitations of the net pay estimation method were highlighted, i.e. limitation in image logs resolution and model compatibility. Based on the limitations and results evaluation, several recommendations for the continuation of the study were formulated, e.g. utilization of image logs with better quality, the introduction of additional training samples from different wells or fields, and net pay estimations on data from a different field.

Preface

The study of petrophysics has existed for many years and contributed significantly to the decision making of hydrocarbon exploration. With a recent advance of modern data science techniques, the implementation of machine learning to various studies in the realm of petrophysics become increasingly interesting and necessary. This master thesis serves as an endeavour to fulfil the appetite for implementation of data science in petrophysics.

This study would not be finished without the perpetual support from my wife, Theresia Bhekti Putranti; and my daughter, Genoveva Ratih Estuariwinarno. Thank you for the endless love and encouragement from faraway in the past two years.

I would like to express my sincere gratitude to my supervisor, Carl Fredrik Berg, for his guidance throughout the specialization project and master thesis in the past year. I would also like to express my gratitude to the people who have helped me in this study: Kurdistan Chawsin, Erik Skogen and Wang Xidong. Also to the academic community in the Department of Geoscience and Petroleum; my friends in NTNU and in the Indonesian Student Association of Trondheim (PPIT); thank you very much.

I hope this study will be beneficial for anyone reading it. Scientia potentia est.

Mikael Yuan Estuariwinarno

Table of Contents

List of Tables

List of Figures

Abbreviations

AC	=	Alternating current
ANN	=	Artificial neural network
CNN	=	Convolutional neural network
FVF	=	Formation volume factor
FMI	=	formation Micro-Imager
GR	=	Gamma ray
HCPV	=	Hydrocarbon pore volume
LWD	=	Logging while drilling
MAE	=	Mean absolute error
MSE	=	Mean squared error
NPD	=	Norwegian Petroleum Directorate
RF	=	Recovery factor
RMSE	=	Root mean squared error
UV	=	Utraviolet

Chapter 1

Introduction

1.1 Background

In the course of hydrocarbon exploration, net pay estimation is an integral part of the hydrocarbon volumetric evaluation. Net pay is defined as a reservoir interval with economically producible hydrocarbons content (Mæland, 2014). Traditionally, net pay estimation is performed by the interpretation of Gamma Ray or borehole image logs. However, it requires a cutoff generation using core data to differentiate sand and shale in a laminated reservoir. A cutoff-dependent net pay estimation might inhibit further study of a particular field as the cutoff itself might not be universally reliable for wells or reservoir interval without core data.

The recent advance of modern data science studies and powerful computation hardware enable the implementation of data science techniques as a solution for various practices in the energy industry. In a previous study by the same author, an alternative net pay estimation method was developed based on the implementation of machine learning on borehole image logs interpretation (Figure 1.1). The cutoff-independent estimation is achieved through the use of a convolutional neural network (CNN). The network is trained to generate a net pay estimation based on the correlation between borehole image logs and net pay fraction from core ultraviolet (UV) photographs. In the preceding study, data from four exploration wells in the Johan Castberg field were utilized, and several preliminary net pay estimations were performed using the aforementioned data. The preliminary estimation results yielded several insights on the optimization approaches for the net pay estimation method, which were implemented in this study.

This study served as a continuation of the alternative net pay estimation method development and was focused on optimizing the method. By optimizing the data processing procedures and evaluating the machine learning implementation, an optimized model was generated to improve the net pay estimation from borehole image logs. The reliability of the model was also evaluated through a statistical analysis to ensure that the estima-

tion is independent of horizontal orientation within the image logs. The capability of this optimized method was then demonstrated in estimating net pay of reservoir intervals without core photographs. Based on the results evaluation and observed limitations of the net pay estimation method, several recommendations for the continuation of this study were formulated.

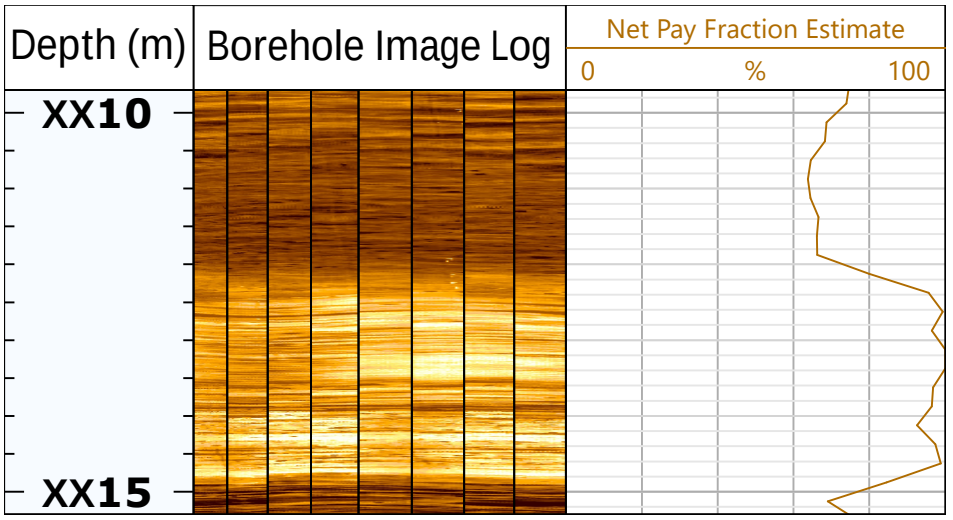


Figure 1.1: An example of net pay estimation of based on a borehole image log interpretation.

1.2 Objective

The objective of this study is to optimize a machine learning-based net pay estimation method from borehole image logs interpretation. The optimization approaches were applied in the borehole image logs processing, core UV photographs processing and machine learning implementation. Several net pay estimations were conducted to evaluate the performance of the optimized method, and several recommendations for the continuation of this study were formulated based on the evaluation of the results.

A workflow was formulated to achieve the objective. The workflow consisted of the following tasks:

1. Optimization of borehole image logs processing by implementing additional processing procedures to improve the quality of the logs.
2. Optimization of core UV photographs processing to generate more accurate labels for the machine learning technique.
3. Evaluation of different elements in the machine learning implementation, including convolutional neural network design and training samples curation.

4. Conducting experiments using the optimized neural network design and data from the optimized processing.
5. Evaluating the reliability of the optimized model by performing statistical analysis on the results.
6. Demonstrating the optimized method capability in reservoir intervals without core photographs.
7. Formulating recommendations for the continuation of the study based on the results evaluation and observed limitations of the net pay estimation method.

1.3 Outline

The master thesis report is comprised of five chapters which discussed different topics. The outline of the report is described as follows:

1. Chapter 1 - Introduction: The first chapter serves as the introduction to the study and is comprised of the motivation and objective of the study.
2. Chapter 2 - Background: The second chapter is comprised of theoretical background and fundamental principle of several elements in this study, e.g. the concept of net pay in hydrocarbon volumetric evaluation; the principle of borehole image log and core data; the fundamental principle of machine learning; and the outcomes of the preceding study.
3. Chapter 3 - Methodology: This chapter talks about how the study is conducted including general workflow of the study; optimized image log and core UV photographs processing; and the optimized machine learning technique implementation.
4. Chapter 4 - Results and Discussions: This chapter is composed of the results of the study and the discussions regarding the results, e.g. optimized processing results; net pay estimation results for each well; accuracy analysis and reliability evaluation based on the results; a demonstration of net pay estimation on reservoir intervals without core photographs; and recommendations for the continuation of the study.
5. Chapter 5 - Conclusion: The final chapter summarizes the main findings and recommendations for continuation of the study.

Since this study is a continuation of a preceding study by the same author, several topics discussed in the Background and Methodology chapter are similar to the topics discussed in the preceding study. Those topics are essential for the integrity of this study; therefore, they were included in this report.

Chapter 2

Background

2.1 Hydrocarbon Volumetric Evaluation

In the scope of a petrophysical study, one of the objectives is to determine the hydrocarbon reserves through a series of volumetric calculation. After a thorough petrophysical interpretation and analysis, a specific interval of the formation will be designated as the reservoir zone and carefully investigated. Some volumetric elements, e.g. gross rock, net sand, net reservoir, and net pay, are introduced during the volumetric evaluation of the reservoir zone. The sequential derivation of those elements is necessary to obtain the hydrocarbon pore volume (HCPV) and eventually the hydrocarbon reserves.

Gross rock is essentially the total thickness of the evaluated interval, i.e. the total thickness of rock in the reservoir zone. Net sand is notionally the thickness of rock with low shaliness or low clay mineral content in the gross rock, which potentially contains hydrocarbon. Net reservoir is a subinterval of net sand and can be delineated by applying a porosity/permeability cutoff. Last but not least, net pay is defined as a subinterval of the net reservoir, which is obtained by applying a saturation cutoff and contains economically producible hydrocarbons for viable exploitation (Worthington, 2010). Figure 2.1 shows how the volumetric elements are correlated to each other in the sense that one is the subinterval of the other.

The net pay subintervals are combined to form the total net pay (T_{net}) (Mæland, 2014). As shown in Equation (2.1), total net pay is an important parameter in calculating HCPV, beside porosity (ϕ), average reservoir area (A), and water saturation (S_w) (Wheaton, 2016).

$$HCPV = AT_{net}\phi(1 - S_w) \tag{2.1}$$

Hydrocarbon reserve (R) can eventually be estimated by introducing the Formation Volume Factor (FVF) and Recovery Factor (RF), as seen in Equation (2.2). Formation Volume Factor is defined as the ratio of the volume occupied by a hydrocarbon phase at

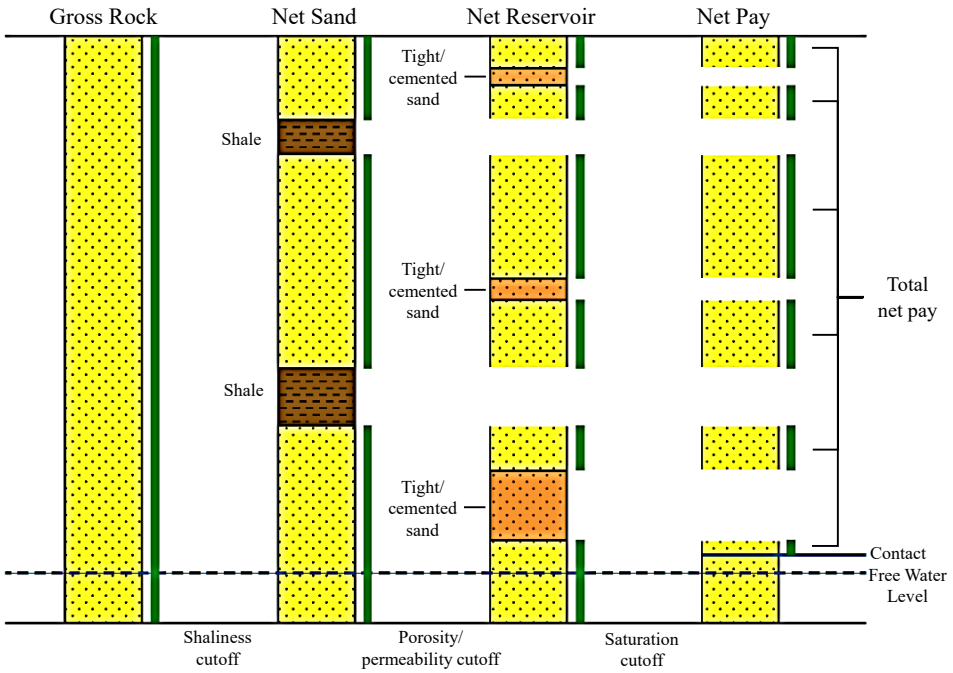


Figure 2.1: Correlation of the volumetric elements in a hydrocarbon volumetric evaluation.

reservoir conditions divided by the volume occupied by the hydrocarbon phase at surface conditions (Fanchi, 2010). Recovery Factor is a term used for a fraction of the hydrocarbons initially in place that can be recovered and depends on the effectiveness of the production mechanism (Metwalli et al., 2017).

$$R = \frac{AT_{net}\phi(1 - S_w)RF}{FVF} \tag{2.2}$$

Beside presenting net pay as a thickness in a unit length, it can also be presented as a ratio in terms of percentage. The term net pay fraction (F_{net}) refers to the ratio of total net pay (T_{net}) over the total gross interval (T_{gross}).

$$F_{net} = \frac{T_{net}}{T_{gross}} \times 100\% \tag{2.3}$$

This term is also commonly known as the Net-To-Gross ratio (NTG). Net pay fraction can simply be multiplied by the gross rock thickness to obtain total net pay.

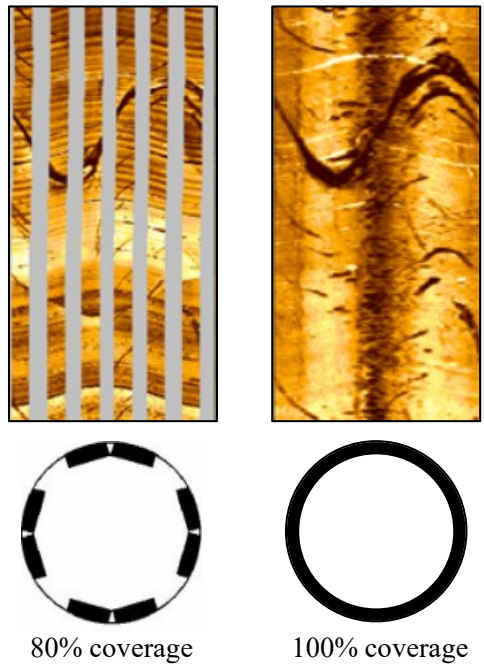
The term ”net pay” is sometimes used interchangeably with ”net sand” or ”pay sand” in various studies in the petroleum industry. In this study, the term net pay is strictly used to denote hydrocarbon bearing rock to avoid confusion with other volumetric terms. On the other hand, the term ”non-net pay” is used for the thickness of rock without hydrocarbon content, either due to shaliness or low porosity/permeability (e.g. calcite cemented sands).

2.2 Borehole Image Logging

2.2.1 Principle of Resistivity Based Borehole Image Logging

Borehole image logging is an attempt to visualize geological features of the formation along the borehole wall for petrophysical evaluation and reservoir characterization. Generally, there are two different image logging techniques with different measurement principles: resistivity and acoustic (ultrasonic) measurement. Both are complimentary when it comes to data acquisition and interpretation.

Resistivity image logging is a pad-based measurement which is able to deliver a high resolution image with up to 80% borehole coverage. It has better tolerance on poor borehole wall conditions and is not affected by the centralization of the tool. Resistivity image logging does not cover the entire borehole due to the limited coverage of each pad (Figure 2.2). The spaces between the pads prevent pads from clashing if the borehole diameter gets smaller due to swelling or thick mudcakes. Acoustic image logging is capable of presenting a full coverage of the borehole because it relies on a rotating acoustic transducer instead of pads. However, it is prone to poor borehole wall condition, poor tool centralization and heavy drilling fluid. The resolution of acoustic image logging is also relatively lower than resistivity image logging (Shahinpour, 2013).



(a) Resistivity image log **(b)** Acoustic image log

Figure 2.2: Comparison of a resistivity image log and acoustic image log. These figures are a reworked version of Figure 4 in Leonard (2016).

Technically, resistivity-based image logging is one of many logging techniques that employs resistivity measurement as its basis. It relies on wireline cable as the conveyance system, although data acquisition is also possible through Logging While Drilling (LWD) systems. Image logging tools generally measure formation microresistivity along the borehole wall through a set of electrode arrays on its pad. Microresistivity is the resistivity measured with a shallow depth of investigation into the formation. It measures the resistivity of the flushed zone, where drilling fluid invasion is prevalent (Snedden, 1984).

The electrodes on image logging tools are arranged in several sets of arrays that are mounted on a pad (Figure 2.3). These pads are located at the edge of a caliper arm. The number of electrodes and pads depends on the type of the tool, but generally, it ranges from 40 to 192 electrodes in 4 to 16 pads. As the pads are touching the borehole wall, an alternating current (AC) flows into the formation between the button-shaped electrodes (lower electrodes) and the upper tool steel housing (Schlumberger, 2013). The upper tool steel housing serves as the upper electrode, and it also contains an inclinometer. The inclinometer provides information on the tool orientation (azimuth, deviation, and relative bearing) in a magnetic earth reference.

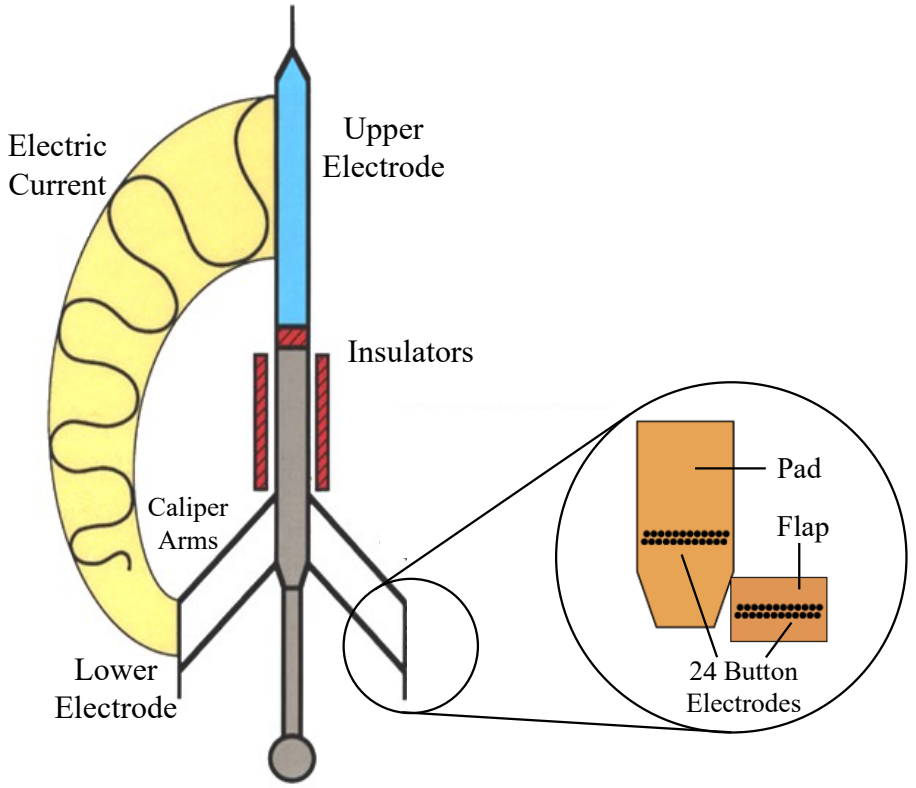


Figure 2.3: Schematic of Fullbore Micro-Imager (FMI) tool. This schematic is redrawn from a figure in page 3 from Schlumberger (2013).

The resistivity image of the borehole wall is created from the current measured by the lower electrodes. The measured resistivity changes according to lithological and petrophysical variations in the rock. The information provided by the inclinometer will be used in determining the orientation of geological features such as beds and faults. The resistivity and inclinometry measurements are presented in an image log, and can be interpreted in terms of lithology, stratigraphic and structural features including dip and fractures (Schlumberger, 2013).

Traditionally, image logging tools are only effective in a conductive environment such as water based mud or highly conductive formations (Luthi, 2001). A conductive environment will generally enhance the flows of the alternating current, and thus improve the quality of the image. However, several modern image logging tools are capable of imaging in a non-conductive borehole fluid provided a sufficiently high formation resistivity and a low ratio of oil to water.

In terms of resolution, most image logging tools yield a 0.2 inch vertical resolution. That implies it is sensitive enough to measure changes in the average formation resistivity within a 0.2 inch interval (Schlumberger, 2013). Table 2.1 displays the measurement specification of the Fullbore Micro-Imager (FMI) tool from Schlumberger, whose output is used in this study.

Table 2.1: Fullbore Micro-Imager (FMI) tool specification (Schlumberger, 2013).

Specification	Details
Range of measurement	Sampling rate: 0.1 inch Borehole coverage: 80% in an 8 inch borehole
Resolution	Vertical resolution: 0.2 inch
Depth of investigation	High-frequency component (image details): 0.39 inch
Mud type	Water based mud (maximum mud resistivity = 50 ohm.m) Oil based mud under specific conditions
Pads	8 Pads (4 pairs of pad and flap)
Button electrodes	192 electrodes (12 electrodes in each pad and flap)

2.2.2 Principles of Image Log Processing

Before interpretation, a series of quality control (QC) and processing is required to reduce logging environmental effects, and also acquisition artefacts. Logging environmental effects include tool sticking, cable speed effect, while acquisition artefact includes noise and dead button electrodes (Firdaus, 2010). This processing is necessary to transform the raw data into an image log with enhanced quality.

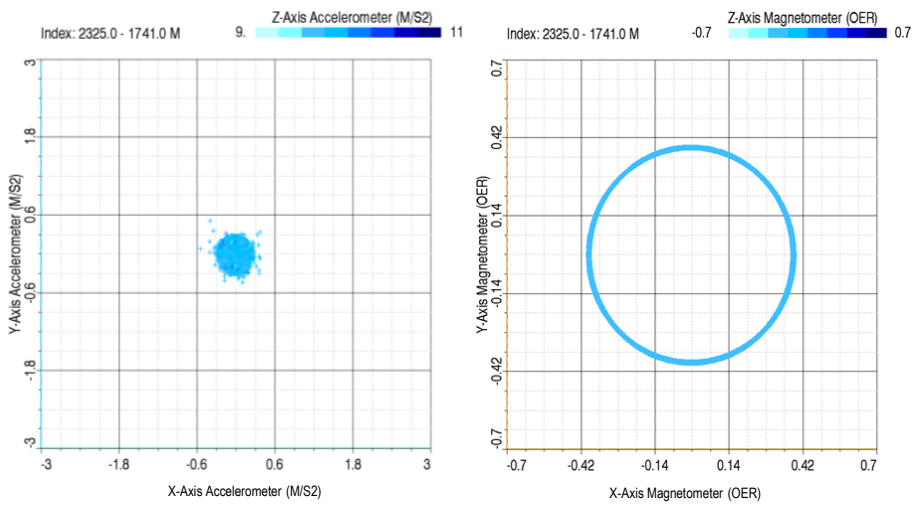
Enhancing the quality of image logs are essential for in the interpretation of the logs as the presence of logging environmental effects or image artefacts might hinder an accurate interpretation. Table 2.2 outlines the necessary processing steps of the image log.

Table 2.2: Image log processing steps.

Processing	Details
Quality Control	Caliper and inclinometry QC
Basic processing	Speed correction Image equalization
Image enhancement	Pads/buttons alignment Dead button repair Filtering
Image normalization	Static normalization Dynamic normalization

Quality Control (QC)

The QC procedure includes investigation of caliper reading and inclinometer measurement. Generally, a caliper check is performed during logging by opening the caliper arms in the cased hole section. The value of caliper reading in the cased hole section must be checked and corrected so that the value is close to the casing inner diameter (ID). Inclinometer measurement is investigated by plotting the accelerometer and magnetometer readings. Ideally, the accelerometer readings should lie in the middle of the plot, and the magnetometer readings should form a circular shape in a vertical well (Figure 2.4). An ideal plot guarantees that the tool is not rotating beyond the tolerance. If erratic readings are observed, a correction must be performed in the petrophysical software (Kalukal, 2014).



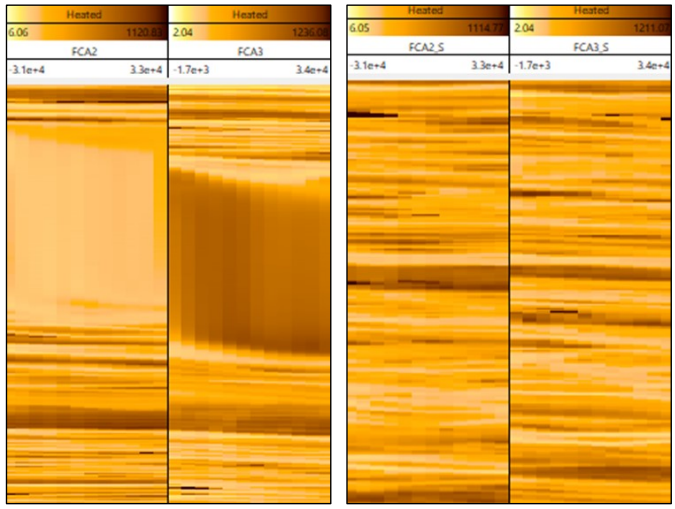
(a) Plot of accelerometer readings

(b) Plot of magnetometer readings

Figure 2.4: Ideal plot of accelerometer and magnetometer reading in a vertical well (Schlumberger, 2019a).

Basic Processing

The next step is basic processing, including speed correction and image equalization. If the image logging tool gets stuck momentarily and then speeds up, a smearing effect might be generated in the image log. A speed correction must be applied to correct the smearing effect (Figure 2.5). Speed correction attempts to correct irregular tool movement which generates artefacts in the image, by utilizing the cable speed measurement. If one or more button electrodes display inconsistent responses that are not caused by lithological variation, image equalization can be implemented to regularize the button electrodes responses. Image equalization is performed by replacing individual gain and offset of each electrode by the average of all electrodes.

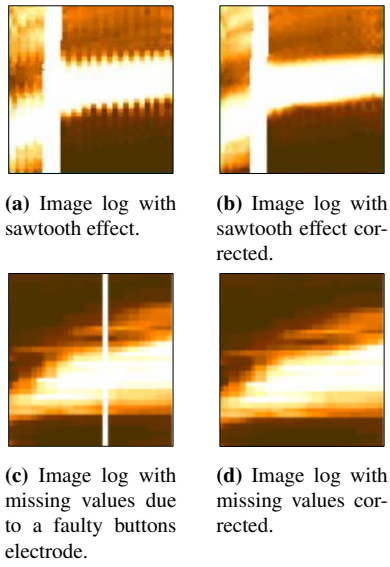


(a) Before speed correction **(b)** After speed correction

Figure 2.5: Speed correction on image log corrects smearing effect due to tool sticking issue from well 7220/7-3S image log which is used in this study.

Image Enhancement

Once basic processing is performed, image enhancement will be conducted. As the button electrodes have different vertical placement on the pad or flap (see Figure 2.3), the raw measurement might display a sawtooth effect in areas with strong contrast (Figure 2.6). Sawtooth effects can be corrected by applying buttons alignment, which involves a vertical shift of the readings from the individual button electrode. In addition to the sawtooth effect, a faulty button electrode might fail to measure, which results in an empty measurement. Missing values due to faulty buttons can be replaced by values computed from neighboring button electrodes.



(a) Image log with sawtooth effect.

(b) Image log with sawtooth effect corrected.

(c) Image log with missing values due to a faulty buttons electrode.

(d) Image log with missing values corrected.

Figure 2.6: Examples of image enhancement on image logs (Schlumberger, 2019a).

Image Normalization

The final processing step will be image normalization, which aims to increase image contrast for a better interpretation. Image normalization attempts to map each measurement value to a position on a normalized color scale, e.g. a scale that ranges from 0 to 255 (Figure 2.7). There are two types of image log normalization: static and dynamic normalization (Figure 2.8). Static image normalization will normalize the values over the entire logged interval. A static image is purposed for large scale feature observation. In dynamic normalization, the values are normalized over a 0.5 m window of the logged interval. This will enhance a tiny detail such as sand shale lamination and is more suitable for small scale feature observation (Donselaar and Schmidt, 2005). Static and dynamic normalization possess a visual contrast for human eyes, but for a machine learning algorithm, it can differentiate layers based only on the pixel values. The color distribution might still be fairly similar between static and dynamic normalization, which implies that the machine learning technique might not be significantly affected by the type of normalization.

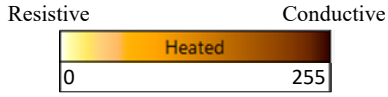
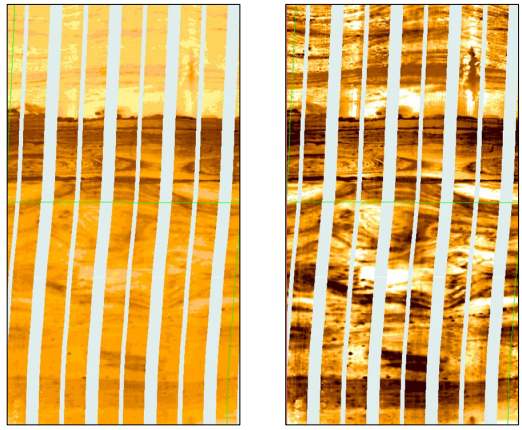


Figure 2.7: An example of an image log color scale with values between 0 and 255. The scale assigns a heated color map on the values which range from white, to orange and brown, to black.



(a) Image log with static normalization.

(b) Image log with dynamic normalization.

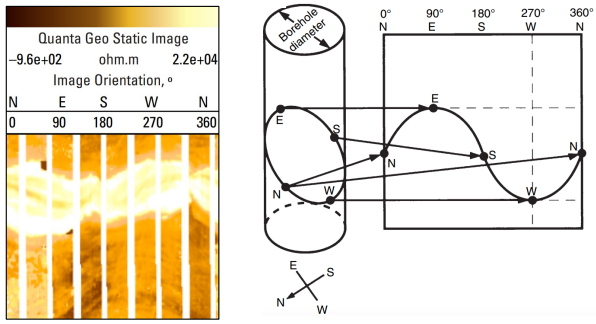
Figure 2.8: Examples of static and dynamic normalization on image logs (Schlumberger, 2019a).

2.2.3 Image Log Display

As the product of borehole image logging, image logs present a high resolution visualization of the formations, which enables interpretation in terms of lithology, structural, and sedimentary analysis. It is also utilized as a supplement for core data, for example, in core depth and orientation matching. In the interval where cores are not sampled, image log acts as a substitute for core data, which enables rock textural analysis (Schlumberger, 2013). In some cases, the acquired cores might not be reliable due to the presence of soft or unconsolidated formation, which may lead to the acquisition of poorly consolidated cores which are easy to break apart (Bajsarowicz, 1992). In those cases, image logs may serve as a supplement to aid the interpretation of unconsolidated formation.

Since image logging tools measure along the perimeter of the borehole, image logs must be displayed as an unwrapped cylinder representing the borehole (Endres et al., 2008). It is typically displayed in a color scale alongside the image orientation scale (Figure 2.9). The color represents resistivity value, and the orientation scale is usually displayed in terms of degree and cardinal direction. Traditionally, darker color represents a conductive layer (shale, open fracture), while the lighter color represents a resistive layer (oil-bearing sands, healed fractures). The vertical white stripes in the log represent the area of the borehole wall not covered by the tool (c.f. Section 2.2.1).

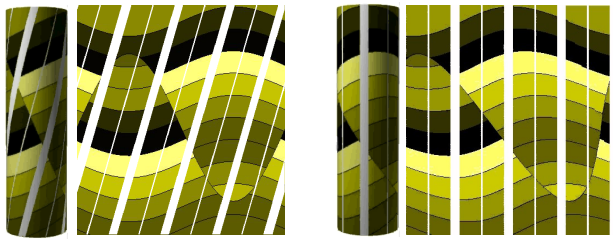
Although it is common to display a borehole image log with image orientation information, it is also possible to display it without orientation. For some specific purposes, such as rock textural analysis and core depth matching, orientation information is not required. In that case, image logs can be displayed without the orientation to enable a straightforward interpretation (Figure 2.10).



(a) A regular display of borehole image log.

(b) An image log displayed as an unwrapped cylinder representing the borehole.

Figure 2.9: Figures showing (a) a regular display of borehole image log and (b) how an image log is displayed as an unwrapped cylinder representing the borehole. Figure 2.9a is taken from Schlumberger (2014), and Figure 2.9b is taken from Donselaar and Schmidt (2005).



(a) Image log with orientation.

(b) Image log without orientation.

Figure 2.10: Comparison of image log dsplay with and without orientation (Schlumberger, 2019a).

2.2.4 Net Pay Estimation Based on Borehole Image Log

Borehole image logs can be utilized for net pay estimation based on the interpretation of sand and shale in the reservoir zone. Net pay estimation based on the image log requires a resistivity curve, which is extracted from the image log. The extracted resistivity curve is the average of resistivity values from each button electrodes for each depth. The interpretation of sand and shale is performed by applying a cutoff on the resistivity curve (Figure 2.11). The cutoff is derived from a core comparison.

As shown in Figure 2.11, interval with a resistivity value above the cutoff will be marked as net pay (h_{net}). The net pay intervals will be added up as the total net pay (T_{net}). Total net pay will be used in HCPV calculation. Furthermore, net pay fraction (F_{net}) can be estimated by averaging the total net pay over the total gross interval (T_{gross}) as in Equation (2.4).

$$F_{net} = \frac{T_{net}}{T_{gross}} \times 100\% \tag{2.4}$$

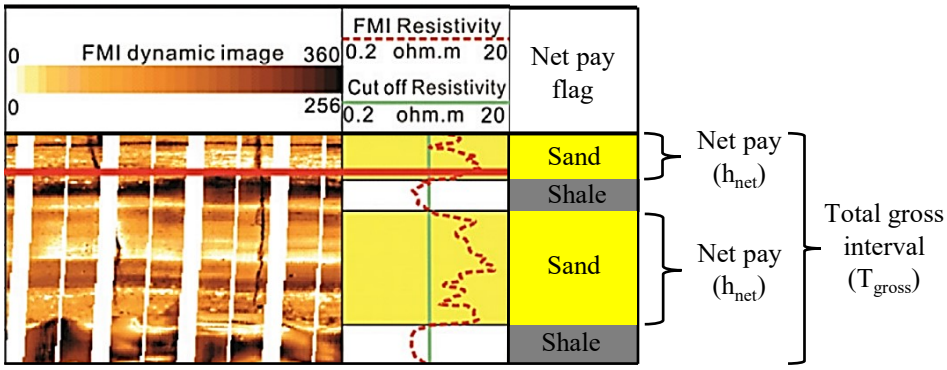


Figure 2.11: An example of net pay estimation based on a borehole image log. A cutoff is applied on the "FMI Resistivity" curve, which is the extracted resistivity curve. This figure is a reworked version of Figure 6 in Wang et al. (2016).

This method has several drawbacks (Gong et al., 2019):

- Cutoff accuracy is sensitive to the logging environment. In a mud with high salinity, the resistivity of sand and shale is not contrasting enough. Besides that, if oil based mud is used as the borehole fluid, we might have non-conductive mudcake, which hinders the resistivity measurement.

- Derivation of cutoff requires a meticulous core comparison which might not be universal for wells without core data. This prevents the user from applying a universal cutoff to uncored wells in a field-wide study.

In this study, a net pay estimation method is developed based on an image log without reliance on an extracted resistivity curve and a resistivity cutoff.

2.3 Core Photograph

2.3.1 Core Acquisition

Core acquisition is an integral part of data acquisition during exploration drilling. In the upstream sector, a core is defined as a cylindrical sample of rock from a certain interval of the well, which is used in core analysis (Figure 2.12). Although wireline or LWD logs can provide insight into the reservoir quality, a study of core samples is essential in determining the commercial producibility of the reservoir (Archer and Wall, 1986). With a higher resolution compared to a regular log, core analysis enables a more detailed characterization of important reservoir properties, e.g. lithology, porosity, permeability, and fluid saturation.

Figure 2.12: Core sample from McArthur Basin, Australia (Revie, 2017).

Cores are acquired by drilling through the formation. Conventional coring system is the most common technique to retrieve core samples. The core is cut with a rotating hollow bit and passes into a barrel around which the drill string rotates (Figure 2.13). After the barrel is full of the core, the core will be snapped free from the formation (Kennedy, 2015). Individual cores are limited in length to prevent possible collapsing under its weight (McPhee et al., 2015). The diameter of core samples is typically half of the bit size, e.g. for an 8.5 inches bit size, the diameter will be around 4 inches. In the conventional coring system, the cores are taken disregarding the precise orientation.

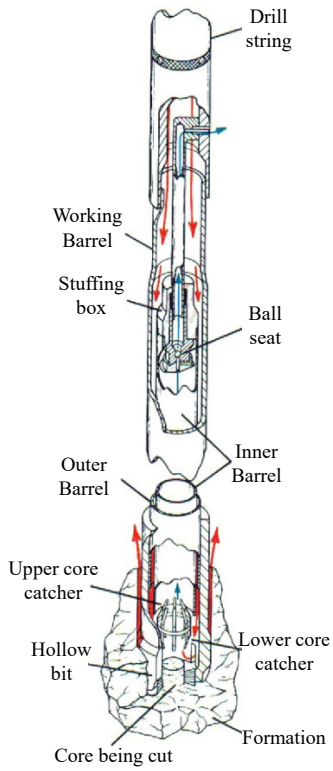


Figure 2.13: Schematic of a conventional coring system. This figure is a modified version of Figure 3.2 from Kennedy (2015).

Once the cores are taken up to the surface, there are several procedures that need to be done before transporting the core to the core laboratory. The first step is removing the core filled liners from the barrels and marking them with depth. The next step will be cutting the liner and core into a certain length, commonly 1 m. Since the physical condition of the core and fluid saturation can be altered during pulling out of the hole, it is vital to preserve and stabilize the core to prevent further alteration due to outdoor exposure and transportation. A preservation technique which is commonly conducted before transportation is resin or gypsum injection into the annulus between the core and the liner (McPhee et al., 2015).

2.3.2 Core Analysis

The goal of core analysis is to reduce uncertainty in reservoir evaluation and characterization by providing reservoir property through a set of laboratory measurements (Al-Saddique et al., 2000). The diversity of information available from a core implies that core analysis is fundamental for geologist and reservoir engineers alike.

A routine core analysis aims to determine fundamental reservoir properties of unpreserved cores, e.g. porosity, permeability, fluid saturation, and grain density. Prior to analysis, a whole core section will be preserved with wax or paraffin (seal peel) for a special core analysis (SCAL), with a length of 25-30 cm. Several cylindrical core plugs (typically with a diameter of 2.5 cm and length of 2.5-7.5 cm) will also be sampled. The core will also be cut into different parts for different purposes (Figure 2.14). After core cutting, the core slabs are photographed with white and ultraviolet (UV) light.

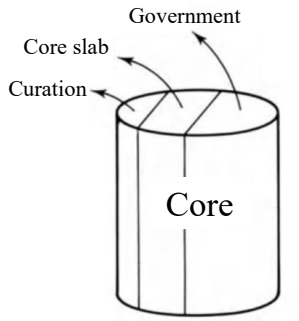


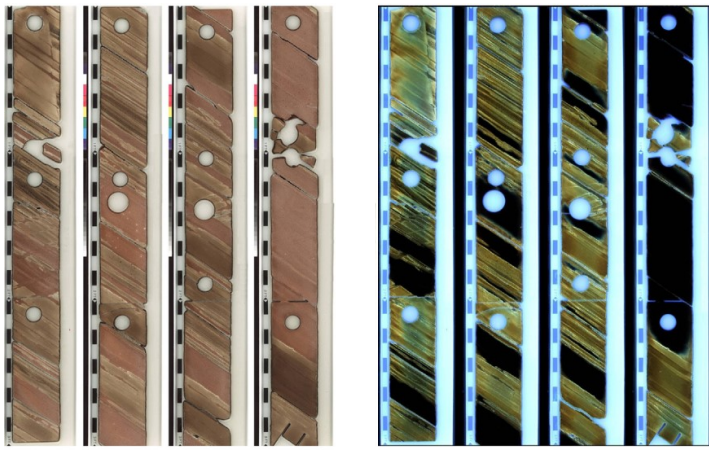
Figure 2.14: Division of a core sample. This figure is a modified version of Figure 5.1 from Archer and Wall (1986).

2.3.3 Net Pay Estimation Based on Core Photograph

One of the steps in core preparation prior to core analysis is the photography of core slabs. Core photography produces high resolution digital images that enable remote observation of the core and also digital interpretation using petrophysical software. Digital images also serve as a visual record that enables a more straightforward data storage and management. The cores are photographed using white light and ultraviolet light (UV) (Figure

2.15). Ideally, the UV photo will highlight oil bearing rock due to oil fluorescence. This provides information on oil-water contact (OWC) or gas-oil contact (GOC) when log data is inconclusive.

A standard color scale is applied to the resulting digital image. White light photo is commonly used for lithology, sedimentology, and stratigraphic examination. Under UV light, oil will fluoresce with orange-brown shading for heavy oils; bright yellow for light oil; and very light white to blue-white for condensate (McPhee et al., 2015). Shale, gas and non-oil bearing rock does not fluoresce under UV light and therefore will have dark or purple shading.



(a) White light core photographs. **(b)** UV light core photographs.

Figure 2.15: White light and UV light core photographs (McPhee et al., 2015).

Core photographs, especially ultraviolet core photographs, can be utilized as an alternative method in net pay quantification. Core UV photos ideally show good contrast between oil stained sands and shale or non-oil bearing rock. This enables a visual net pay quantification by generating binary flags of net pay and shale, based on the color shading of core UV photos (Anyaehie, 2015). As shown in Figure 2.16, interval with visible oil staining in the UV photos will be marked as net pay (h_{net}). The net pay intervals will be added up as the total net pay (T_{net}).

The concept of distinguishing net pay based color intensity inspired the method of net pay fraction labels generation in this study. Net pay fraction values were generated from core UV photographs by converting them into binarized core UV images, which consists of black and white color. White color represents net pay, and black color represents shale or non-oil bearing rock. Further details on net pay fraction labels generation are explained in Section 3.5.3.

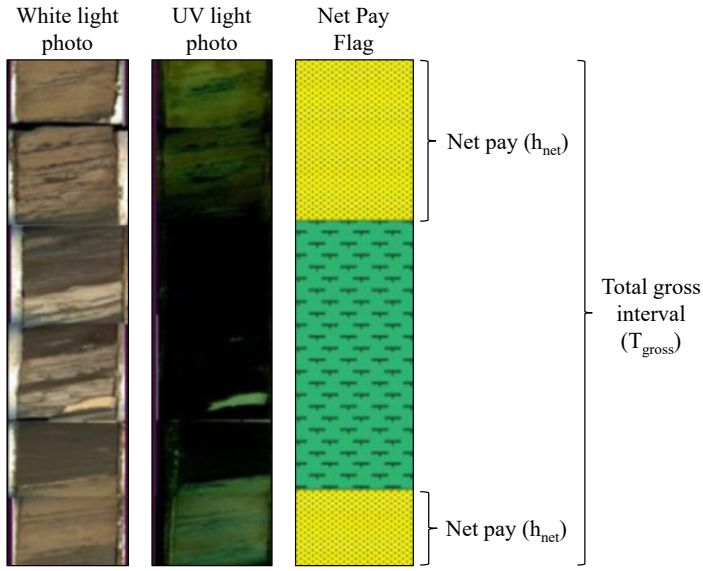


Figure 2.16: Net pay estimation based on core photographs. This figure is a reworked version of Figure 4.1 in Anyaehie (2015).

2.4 Machine Learning for Continuous Value Estimation

2.4.1 Introduction to Machine Learning

In this modern era where digitalization has penetrated through various sectors, machine learning is one of the data science techniques applied to extract knowledge and insights from a dataset. By definition, machine learning is the process of generating models to learn from available data in order to make predictions on future data, based on a certain algorithm (Samanpour et al., 2018). Machine learning aims to produce a precise and automated prediction from densely packed information in a large dataset and transforms it into a useful format.

In general, machine learning can be categorized as a subset of Artificial Intelligence (AI). Typically, machine learning is divided into three types based on the learning characteristic: unsupervised, supervised, and reinforcement learning (Figure 2.17).

Unsupervised Learning

Unsupervised learning attempts at finding hidden patterns or intrinsic structures in a dataset (input) without correct answers or labels (output). The term ”unsupervised” refers to how the model learns from the dataset without supervision or guideline from a ”teacher” which is the correct label. Although correct labels are not provided, the algorithm attempts to identify patterns within the data so that data with something in common are categorized together (Marsland, 2014). Unsupervised learning includes any algorithm where the model

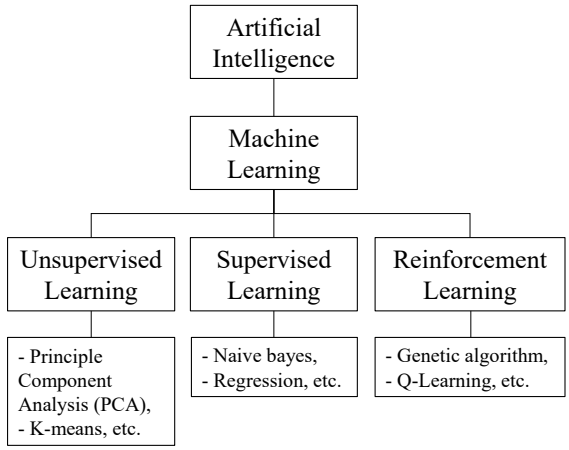


Figure 2.17: Classification of machine learning based on the learning characteristic.

is only learning based on input data without the corresponding correct label, e.g. Principle Component Analysis (PCA).

Supervised Learning

Supervised learning differs from unsupervised learning as it generates a model based on a dataset (input) with correct answers/labels (output). The model serves as a mapping function that will learn the correlation between input data and the labels. Based on a model, the algorithm attempts to predict correct output to any unseen input data in the future (Marsland, 2014). The prediction can either be an estimation of continuous value (regression) or a prediction based on some certain classes (classification). Examples of supervised learning include naive bayes and regression.

Reinforcement Learning

Reinforcement learning provides a distinct approach compared to supervised and unsupervised learning. Contrasting with the aforementioned algorithms, evaluative feedback is given to the model based on the prediction, without providing a correct answer/label (Li, 2018). The algorithm attempts to explore and try out different possibilities until it generates a correct prediction. Genetic algorithm is an example of the algorithm within reinforcement learning.

Generally, machine learning techniques are executed with the aid of programming software. In this study, implementation of machine learning technique for net pay estimation is considered as supervised learning because the model was generated based on a dataset (image logs) with correct labels (net pay fraction from core UV photographs). The machine learning technique in this study was implemented through the use of artificial neural network principles, which is explained in the following section.

2.4.2 Artificial Neural Network

An artificial neural network (ANN) is a multi-layered computing system that consists of a multitude of interconnected nodes (Figure 2.18). An artificial neural network is composed of one input layer, one output layer with one or multiple hidden layers between the input and output layer. The input layer is composed of independent variables from the data, while the output layer is the prediction made by the network. The hidden layer is comprised of multiple nodes or neurons which perform a series of computation on the input data. An artificial neural network is comprised of several elements and computation mechanism, which are explained in this subsection.

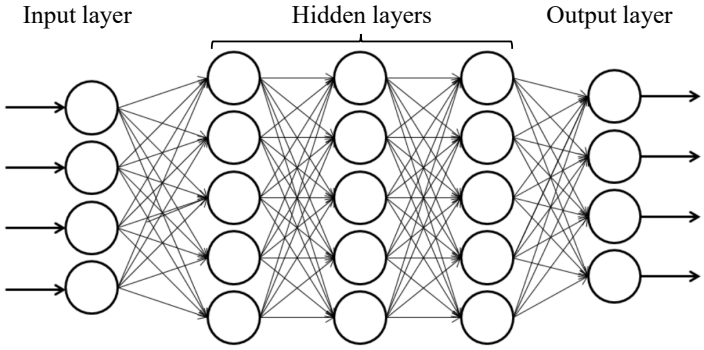


Figure 2.18: General architecture of multilayered artificial neural network. This figure is a modified version of Figure 1 in Miralles et al. (2016).

Neuron

An artificial neural network is comprised of multiple neurons that are connected by links where interactions occur between the neurons (Goodfellow et al., 2016). Each neuron is connected to a set of inputs through some weighted link and generates an output on the other end (Figure 2.19). The inputs are independent variables from the dataset and represented by x_1,x_2,x_3...x_n. The independent variables come from a single observation, e.g. for an observation of a reservoir rock, the variables can be porosity, permeability, and grain density. A dataset comprises of a collection of observations with its corresponding variables. Each input is multiplied by the weighted link or commonly referred to as a synapse. The weights are represented by w_1,w_2,w_3...w_n and correspond to the strength of the synapse.

Weights are pivotal in ANN as they determine the essential variables, and thus which input signal will be passed through the neurons. Generally, if a particular weight is less than 1, it will weaken the input signal; and if it is higher than 1, it will amplify the signal. Initially, random values are assigned on the weights. Throughout the training process, the weights are altered and updated to improve the performance of the model. The sum of input and

weight multiplications (S) is computed by the neuron, using Equation (2.5).

$$\begin{aligned} S &= x_1 w_1 + x_2 w_2 + ... + x_n w_n \\ &= \sum_{i=1}^{n} x_i w_i \end{aligned} \tag{2.5}$$

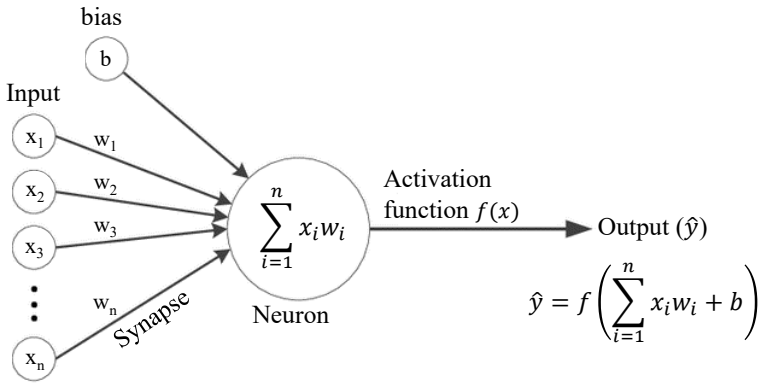


Figure 2.19: Schematic of a neuron in artificial neural network. Each individual neuron operates as a single computation unit of the network.

Activation Function

Beside summation, the neuron will also apply an activation function, $f(x)$. The activation function is a function that decides the activation of the neuron as a response to the weighted inputs (Marsland, 2014). In this case, activation refers to the passing of the input signal from one neuron to the neuron in the next layer. Contrasting with the linear summation shown in Equation (2.5), the activation function is a non-linear function with a degree of more than one. The complexity of a non-linear function will introduce non-linearity on the output of the neuron. Non-linearity is important because it helps the model in learning non-linearities in complex data.

As seen in Figure 2.19, besides the weighted input, there is also a bias (b) that is passed into the neuron. Bias is an additional input for the neuron, which comprised of a bias neuron and a bias weight. The bias neuron is added to each layer in the neural network and linked to the neurons in the layer. Bias neuron stores the value of 1, and the weight of the bias is usually initialized with zero values. Bias works by shifting the activation function output to the left or right along the x-axis. The goal is to generate a better fit with the data, which is critical for successful learning. The output of the neuron ($\hat{y}$) is calculated by adding the summation result form each neuron with bias and then passing it into an activation function, i.e. Equation (2.6). In general, weight influences the slope of activation function, while bias is used to shift the activation function.

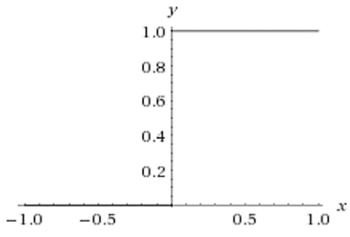
$$\hat{y} = f\left(S + b\right) \tag{2.6}$$

Generally, having an activation function with the best mapping of input to the output is desirable. There are two important criteria for an activation function:

- **Monotonically increasing**: Monotonically increasing implies that the function is entirely increasing. This characteristic helps the neural network to converge easier into a more accurate prediction (Goodfellow et al., 2016). During the training phase, back-propagation dictates the amount of influence of each neuron to the neurons in the next layer through a weight update. Without a monotonically increasing function, increasing the weight might lead to less influence, which means the model is unlikely to converge to a state that yields an accurate prediction.

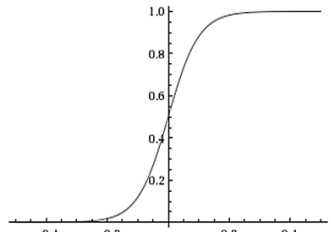
- **Differentiable**: Differentiable implies that the function can be differentiated to a lower order function instead of a constant. Back-propagation requires the gradients of the activation function to be supplied alongside the error in updating weights and biases. Weight optimization also utilizes gradient descent, which means the function has to be differentiable.

The most commonly used activation functions are shown in Figure 2.20. Each function possesses a different characteristic, as outlined in Table 2.3.



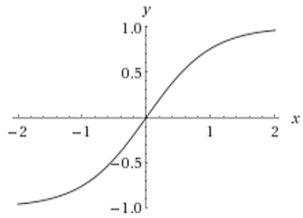
$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$$

(a) Step



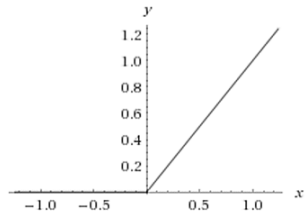
$$\sigma(x) = \frac{1}{1+e^{-x}}$$

(b) Sigmoid



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

(c) Tanh



$$f(x) = \begin{cases} 0 & \text{for } x \leq 0 \\ x & \text{for } x > 0 \end{cases}$$

(d) ReLU

Figure 2.20: Commonly used activation functions for artificial neural networks.

Table 2.3: Activation functions and their characteristics.

Activation Function	Details
Step	- A threshold-based activation function which activates the neuron if the input value is above a certain threshold. - The output is a binary signal (0 or 1).
Sigmoid	- Function with an s-shaped or sigmoid curve. - The output ranges between 0 and 1. - Monotonically increasing. - Output is not zero centered, hence gradient updates might go too far in different direction which leads to a more difficult optimization. - Suitable for a probability prediction.
Hyperbolic tangent (Tanh)	- Function with an s-shaped or sigmoid curve. - The output ranges between -1 and 1. - Monotonically increasing. - Output is zero centered, enabling a more fluid optimization.
Rectified linear units (ReLU)	- Commonly used in modern artificial neural network. - The output ranges between 0 and inifinite. - Monotonically increasing. - Efficient because not all neurons are activated simultaneously. - Output is not zero centered, similar to Sigmoid function.

Forward-propagation

Forward-propagation is one of the mechanisms in the artificial neural network. The first layer of ANN sends outputs to all the neurons in the second layer (the hidden layer). The neurons in the second layer utilize the first layer outputs as input, and they will, in turn, send outputs to the subsequent layer (Figure 2.21). This output propagation goes all the way to the final output layer and is referred to as forward-propagation. The final output layer will be the prediction.

The synapses which connect the neurons can either be weighted with zero or non-zero values. The weight indicates the importance of the input, which goes through the synapse. Zero value means the input will be discarded, while non-zero values indicate that the input will be used by the neuron. The weights will be updated through the learning process during back-propagation later on.

Loss function

A loss function is a measure of error based on the difference between the prediction ($\hat{y}$) and the correct answer or label (y). Every observation in a dataset is given a label that serves as the ground truth in loss calculation. The final output layer in ANN will be the prediction, which is the result of the computation and propagation in the network. The key to evaluating the performance of the network is by computing the loss. The lower the loss, the closer the prediction value to the actual value.

Commonly used loss evaluation metrics for prediction of a continuous value is outlined in Table 2.4.

Table 2.4: Loss functions which are commonly used in ANN.

Loss Function	Formula
Mean squared error (MSE)	$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
Mean absolute error (MSA)	$MAE = \frac{1}{n}\sum_{i=1}^{n}\lvert y_i - \hat{y}_i\rvert$
Root Mean Squared Error (RMSE)	$RMSE = \sqrt{\sum_{i=1}^{n}\frac{(y_i - \hat{y}_i)^2}{n}}$

Back-propagation

Once a loss is obtained, it is utilized in back-propagation. Back-propagation is an essential part of the learning process, where a series of computations is performed backward along the layers in order to update the weights in the network (Figure 2.21). It is conducted by utilizing the loss value in a series of partial derivative computations in each neuron (Yamashita et al., 2018). The computation is performed backward on every neuron from the last hidden layer all the way to the first hidden layer. Back-propagation attempts to generate optimum weights, which will minimize the loss. Once back-propagation is finished, the loss is re-evaluated to see if it is successfully minimized. An optimization algorithm is required to achieve a minimum loss. The basic optimization algorithm in ANN is the gradient descent method.

Gradient Descent

When applied in ANN, gradient descent attempts at obtaining the optimum weights, which will give the minimum loss (Figure 2.22). It is achieved through a series of computations in each neuron during back-propagation. In each neuron, the first computation is a partial derivative of the loss function (J) with respect to weight (w), or commonly referred to as loss function gradient (Goodfellow et al., 2016). The computation of loss function gradient

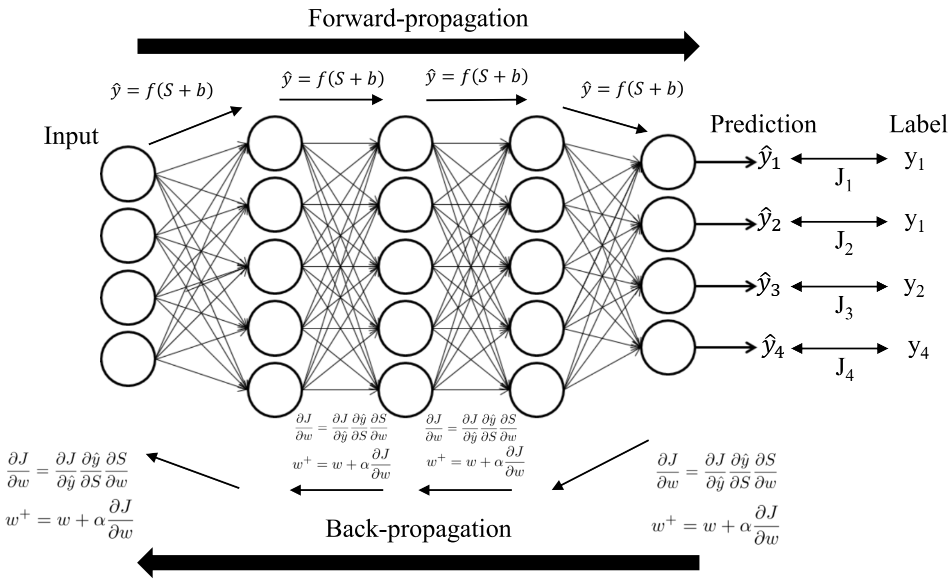


Figure 2.21: Forward and back-propagation mechanism.

involves activation function output from each neuron ($\hat{y}$), and the sum of input and weight multiplications from each neuron (S) as shown in Equation (2.7).

$$\frac{\partial J}{\partial w} = \frac{\partial J}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial S} \frac{\partial S}{\partial w} \tag{2.7}$$

Once the loss function gradient is obtained, the updated weight (w^+) of a certain synapse can be computed. An additional parameter, learning rate (α) is introduced in Equation (2.8). The learning rate regulates how fast the weight is updated, which implies if the learning rate is too big or too small, the model will fail in learning the proper values for the weights.

$$w^+ = w + \alpha \frac{\partial J}{\partial w} \tag{2.8}$$

After the weights are updated, the cycle is repeated, beginning with computation on each neuron, forward-propagation, and finally, loss re-evaluation. In back-propagation, optimization of weights ideally leads to minimization of the loss. The goal is a minimization of loss function through a series of learning iteration until it converges to a certain minimum. Once the loss converges to a certain minimum, the model is ready to predict a future unseen data. A prediction will be performed on a test set that contains data which is not included in the training dataset. The prediction based on the test set will be evaluated to investigate the accuracy and performance of the model.

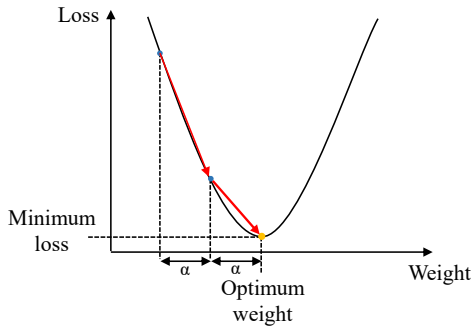


Figure 2.22: Gradient descent in the search of optimum weights.

2.4.3 Convolutional Neural Network

Convolutional neural network (CNN) is an extension of an artificial neural network, in which convolutional and pooling layers are added to extract important features within an image dataset. A convolutional neural network is suitable for both supervised and unsupervised learning tasks. In supervised learning, it can be applied for the estimation of continuous values (regression) or classification based on classes. It is also applied for unsupervised learning tasks, e.g. dimensionality reduction of a large image dataset using the principle of an autoencoder. The architecture of CNN is comprised of convolutional, pooling, and fully connected layers (Figure 2.23).

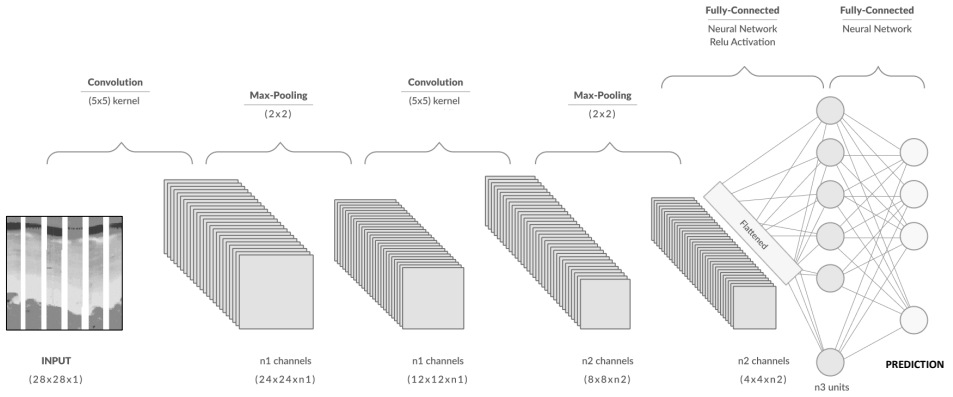


Figure 2.23: An example of a convolutional neural network architecture. This figure is a reworked version of an image in MissingLink (2019a).

Input

A convolutional neural network is commonly used on a dataset which consists of images. An image is an array of pixels with individual pixel values. The pixel values have a certain range, which depends on the color scale, e.g. binary, grayscale, or Red-Green-Blue (RGB).

- **Binary image**: Images with black and white color with pixel values of 0 or 1. The shape of the array is two dimensional.
- **Grayscale image**: Images with pixel values between 0 and 255 in an 8-bit format. The shape of the array is two dimensional.
- **RGB image**: Images with three color channels: red, green, and blue. The array shape is three dimensional, and the third dimension represents the color channel. The pixel values depend on the color depth, e.g. 8-bit RGB image will have pixel values between 0 and 255.

Convolutional Layer

The convolutional layer is one of the building blocks of CNN. It operates convolution operations, a mathematical operation that merges two sets of information. Convolutional layers aim to detect visual features in images such as edges, lines, curvatures, color variation and layering (Yamashita et al., 2018). There are three essential elements in a convolutional layer: image input, filter, and feature map. The convolution is implemented on the input image by using a filter to produce a feature map.

Filters, or commonly referred to as kernel, acts as feature identifiers. A filter is an array of values that will identify an essential feature, such as lines, edges, curves and colors. These values act as the weight in CNN. Throughout the training, the values in each filter are ideally converging towards the optimum values which will minimize the loss.

Convolution is performed by sliding the filter over the input image. As shown in Figure 2.24, it begins by placing the filter at the top left corner of the input image. The area where the filter sits on the image is referred to as the receptive field. An element-wise multiplication is performed in the receptive field. The multiplications are summed up, and the result is stored in the first cell of the feature map. Each convolution value represents a specific receptive field in the image. Once a convolution operation is stored, the filter will be moved one column/row away, either horizontally or vertically. The movement of the filter is referred to as stride. The second convolution value is stored in the second cell of the feature map, which lies next to the first cell. This process is repeated until all areas of the image is covered, and the feature map is entirely populated.

As in the artificial neural network, non-linearity is essential in CNN. A rectified linear unit (ReLU) is generally introduced as the activation function at the end of a convolutional layer. It will increase the non-linearity even further to achieve better prediction. The result of the convolution operation will be passed through a ReLU activation function. The final value in the final feature maps will be the output of the ReLU function. The feature map will be the output of the convolutional layer which will be passed on to the next layer.

Since one filter will eventually learn only one specific feature, multiple filters are required to learn all the important features in the image. In CNN, the first convolutional layer is comprised of multiple filters, as shown in Figure 2.25. Multiple convolutional layers can be added in CNN, to gain a better feature identification. Convolutional layers are capable

of learning hierarchies of features by preserving the spatial relationship. For example, the first convolutional layer learns basic patterns such as lines; while the second convolutional layer learns feature composed of basic patterns learned from the first layer. This continuous process allows CNN to learn increasingly complex features within an image dataset through training.

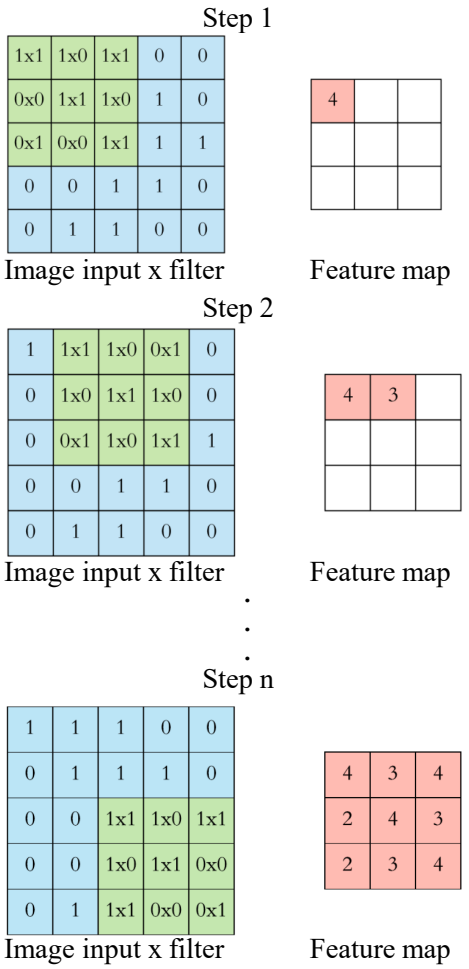


Figure 2.24: An example of convolution operation in a convolutional neural network. The green box is the filter which is applied upon an image input. For each step, the computation result of the convolution operation is stored in the feature map until it is fully populated. This figure is a reworked version of an image in Dertat (2019).

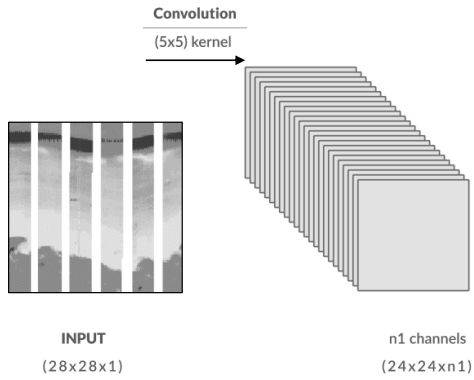


Figure 2.25: An example of a convolutional layer output in convolutional neural network. This figure is a reworked version of an image in MissingLink (2019a).

Pooling layer

Besides the convolutional layer, another integral part of the CNN is pooling layers. Pooling layers take the feature maps from convolutional layers as the input and attempts to reduce the dimensionality of the feature map. The goal is to train the network in learning features in the image without being confused by spatial variation in the image, e.g. textures, feature position and angle. Pooling layers downsample each feature map independently by reducing the size of the array while keeping the depth intact (Torres, 2016). This enables a shorter training time, and prevents the generation of an excessively complex model which leads to overfitting.

The common pooling method in CNN is max pooling. Similar to convolution, max pooling is implemented by sliding a pooling window over the input image (Figure 2.26). The pooling window will be placed over a particular area in the feature map. However, instead of performing element-wise multiplications, the pooling window will identify the maximum value over a particular area. The maximum value will be stored in a new feature map. The window will then be moved a certain row/column away, depending on the value of stride. This process is repeated until the new feature map is fully populated. The new feature map is commonly referred to as a pooled feature map.

The pooled feature map is a downsampling product of the initial feature map (Figure 2.27). The creation of a pooled feature map implies that unimportant features on the initial feature map are disposed. This enables the network to work more efficiently. Max pooling is applied to each of the feature maps from the previous convolutional layers separately. Hence the amount of pooling windows in the pooling layer will be the same as the number of filters in the convolutional layer. A regular CNN architecture might have more than one pair of convolutional and pooling layer. The output of the last pooling layer will be passed through the fully connected layer.

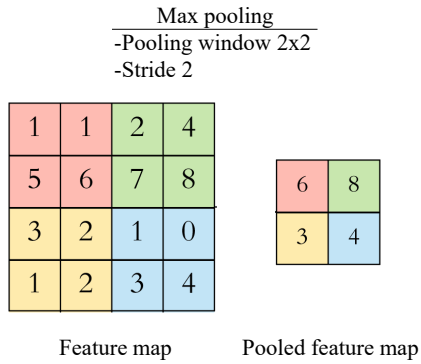


Figure 2.26: An example of max pooling operation in a convolutional neural network. The pooling window will be placed over a certain area in the feature map, and identify the maximum value over a certain area. The maximum value will be stored in a pooled feature map. This figure is a reworked version of an image in Dertat (2019).

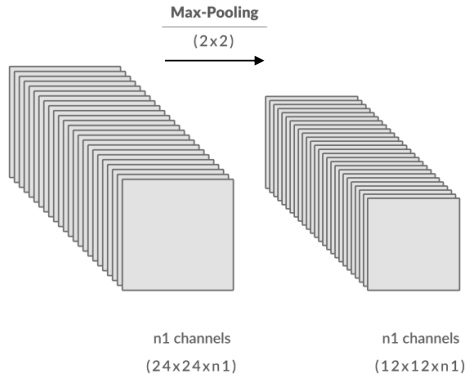


Figure 2.27: An example of a pooling layer output in a convolutional neural network. This figure is a reworked version of an image in MissingLink (2019a).

Fully connected layer

The final part of the CNN architecture is the fully connected layer. A fully connected layer is an artificial neural network that takes pooled feature maps as input and will eventually output a prediction, which can either be a classification or estimation of continuous values. As in artificial neural networks, the prediction will be based on a set of labels which serves as the ground truth.

The output of a pooling layer is three dimensional. Therefore it must be transformed into a one-dimensional array before being fed into a fully connected layer. The transformation is performed by flattening the pooled feature maps into a 1D array (Figure 2.28). The flattened pooled feature maps contain values that represent a probability that a particular

feature belongs to a label (Yamashita et al., 2018). It is possible to have more than one fully connected layer which works in the same manner as in ANN. The array will be processed by the neurons through a series of computations through a forward-propagation mechanism.

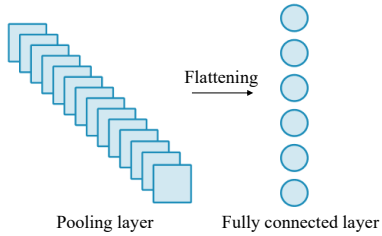


Figure 2.28: An example of flattening operation, in which the final pooled feature map is re-arranged into a 1D array. This figure is a reworked version of an image in MissingLink (2019b).

In the final layer, the activation function will generate the final output, which is the prediction. For classification tasks, each neuron in the final layer corresponds to a particular class. The activation function will generate a probability value for each class. The class with the highest probability will be selected as the prediction (Figure 2.29). The prediction will be evaluated with reference to the label. The loss will be utilized to update the filters, feature maps, and the weights in a fully connected layer through a back-propagation mechanism. The whole process is iterated, and once the loss converges to a certain minimum, the model is ready to predict unseen images in a test set.

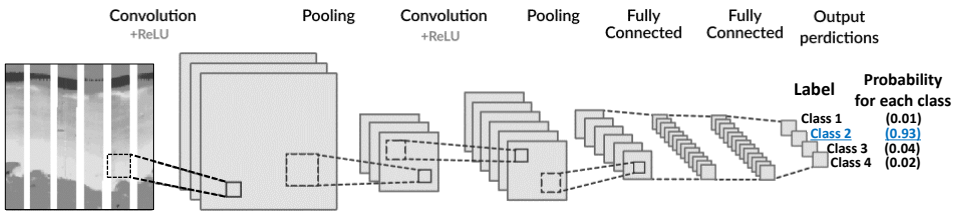


Figure 2.29: An example of classification prediction generated by convolutional neural network. This figure is a reworked version of an image in MissingLink (2019a).

The number of neurons, type of activation function, and loss function in the final layer will vary depending on the task. For classification, the number of neurons in the final layer corresponds to the number of classes; sigmoid or softmax is preferred as the activation function; and cross-entropy is used to measure the loss. For regression of a single continuous value, there will only be one neuron in the final layer, which will generate the prediction; the preferred activation function is a linear function; and mean squared error is the preferred loss function.

Hyperparameters of CNN

A hyperparameter is a parameter that is set before the learning process. In CNN, there are several hyperparameters which are commonly set at the beginning:

- **Number and size of filter**: The number of filters specifies the number of features to be learned in the image. A higher number of filters results in a more powerful model. However, it increases the risk of overfitting due to increased parameter count (Torres, 2016). Generally, the first convolutional layer starts with a small number of filters, and the number is progressively increased for the subsequent layers. Commonly used filters are a square-shaped array with a size of 3x3, 5x5 or 7x7. The decision on the filter size depends on the task performed. Small-sized filters are more suitable to learn local features without much image overview. For large scale features learning, larger size filters can be utilized.
- **Stride**: Stride specifies the amount of convolution filter movement at each step. Generally, convolution introduces overlap between the receptive fields. If less overlap is preferred, a more substantial value of stride can be applied, and this results in a smaller feature map.
- **Padding**: Padding refers to the addition of zero values around the input before applying a filter/pooling window (Figure 2.30). Padding aims to preserve the size of the feature maps. Normally, the size of the feature map is smaller than the input. Therefore for CNN with many convolutional/pooling layers, the feature map might shrink into an unwanted size. With the introduction of padding, the dimension of the feature map is preserved.

2.4.4 Technical Aspects of Artificial Neural Network

An artificial neural network requires a considerable parameter tuning and practical preparation step on the data. Some of the technical aspects are outlined in this section.

Dataset Split

In ANN, the dataset is generally split into three types (Marsland, 2014):

- **Training set**: Training set is designated to train the algorithm in learning the patterns/mapping the correlation. The samples in the training set are utilized by the model for learning throughout the iteration.
- **Validation set**: The validation set is used to evaluate the model performance for each iteration. The model does not learn from the dataset in this set, and the loss from this set is not used in back-propagation. Instead, the validation loss will be observed to fine-tune the hyperparameter.
- **Test set**: Test set is utilized to produce the final prediction. The samples in the test set have not been seen before by the model, and it is used after all the training iterations are completed.

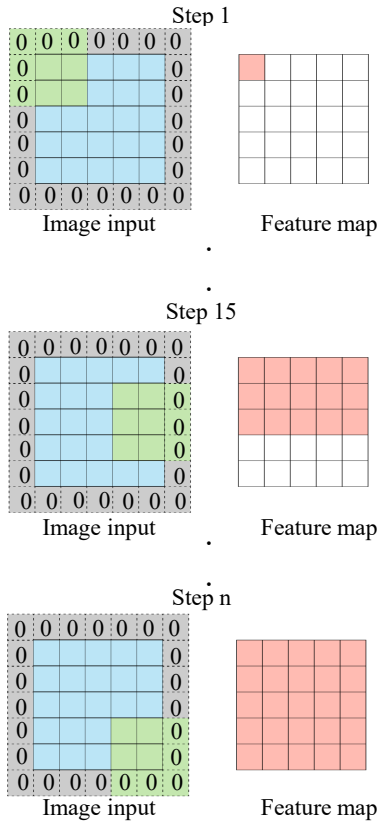


Figure 2.30: An example of padding mechanism which aims to preserve the size of the feature maps by adding zero values around the input. This figure is a reworked version of an image in Dertat (2019).

In supervised learning, each sample will be given a label that serves as the ground truth. The dataset split normally follows a general rule, e.g. 64% of the samples for the training set, 16% of the samples for the validation set, and 20% of the samples for the test set (Figure 2.31).

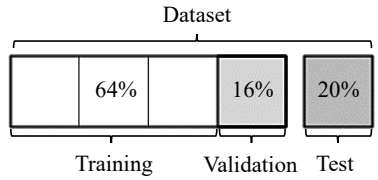


Figure 2.31: An example of dataset split in artificial neural network.

Weight initialization

Weight initialization attempts to give starting values for all the weights for the first iteration. Generally, weights are assigned with random values as part of the weight initialization. Random initialization is preferred to introduce non-linearities in the initial training iteration.

Advanced weight initialization methods, e.g. Xavier and He initialization, use a different approach in initializing weights. Xavier and He initialization manipulate the weights differently, but essentially they are setting the weights close to 1. The aim of using advanced weight initialization is to mitigate computation problems during training, e.g. vanishing and exploding gradients (Marsland, 2014). Vanishing gradients means that the gradient of the loss function during the learning process becomes too small to flow backward during back propagation for updating the weight. This results in a minor weight update and slower minimization of the loss function. Exploding gradients is the exact opposite of a vanishing gradient as the loss gradient becomes too large, and the minimization result will oscillate around the minima. Oscillation around the minima might lead to unsuccessful learning by the model, which is unwanted.

Optimization Algorithm

An artificial neural network employs an optimization algorithm (optimizer) in search of optimum weights, which will minimize the loss. There are various optimizers which are used in ANN, and they are mostly based on the gradient descent method. Beside utilizing concept from gradient descent, many optimizers implement elements from stochastic gradient descent (SGD) and mini-batch gradient descent.

Stochastic gradient descent updates weights by using a single observation instead of the whole dataset as the input for every iteration. The observation is selected randomly, hence the term ”stochastic” is used. In the conventional gradient descent method, the whole data is used as an input for every iteration, and the weights are updated based on the whole dataset. It often results in a long computation time due to the heavy use of computer memory to load and process the whole data. In SGD, a single observation is processed every iteration. The weights of the network are updated based on the observation. It results in a lighter algorithm with significantly faster computational time.

One of the main characteristics of SGD is the fluctuation, which is prevalent during the search of the global minimum. The fluctuation is due to the random selection in SGD. A single observation might not represent the whole data; therefore, it contains an element of noisiness (Torres, 2016). The fluctuation is beneficial when SGD is stuck around a local minimum (Figure 2.32). The smoothness of regular gradient descent makes it prone to being stuck around a local minimum.

Stochastic gradient descent has the element of fluctuation, which might hinder its ability to settle down around a global minimum. An alternative optimization method exists as a trade-off between regular gradient descent and SGD. The method is referred to as the

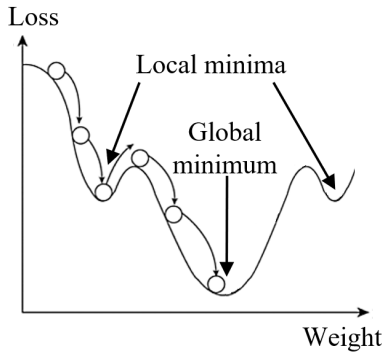
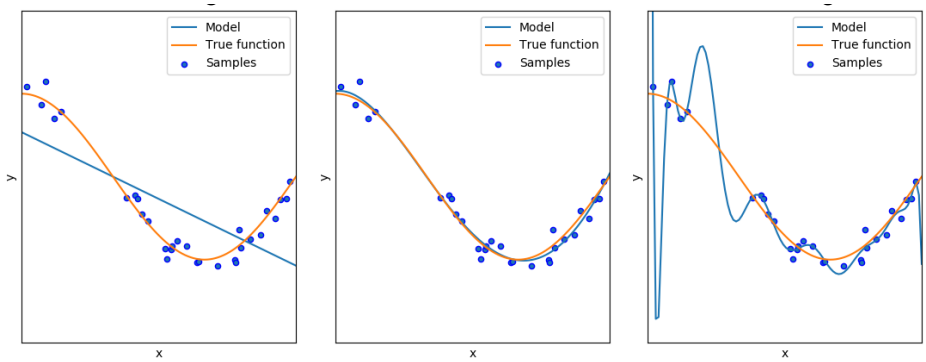


Figure 2.32: Avoiding local minima with stochastic gradient descent.

mini-batch gradient descent method. Instead of using a single observation as in SGD, it utilizes a batch of observations. The weights are updated based on the batch.

Problems on Model Fit

By definition, a model is a system that maps the input to output. Model fit is a measure of how well a machine learning model generalizes to unseen data (Torres, 2016). Model fit can be observed by plotting the training and validation loss for every iteration in the training process. For each iteration, training loss is the final loss based on prediction on training data, while validation loss is the final loss based on prediction on validation samples. During training, the model does not learn from the validation set; therefore, it can be regarded as unseen data for evaluating model performance during training. Ideally, training and validation loss should converge to a certain minimum, which indicates it has a good generalization on both seen and unseen data.



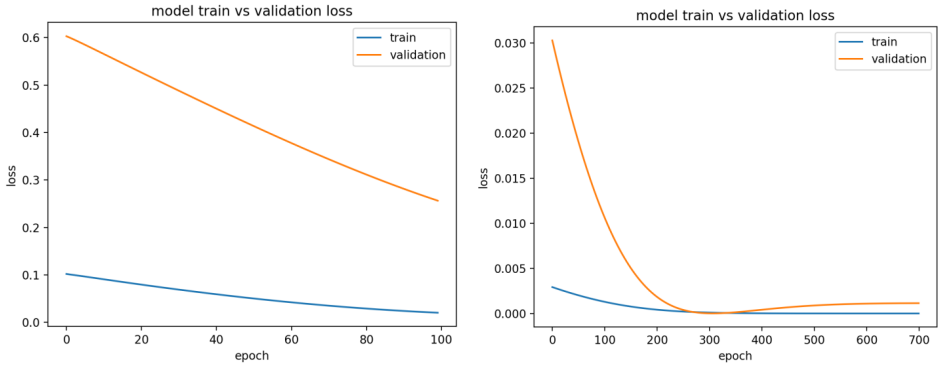
(a) Underfitting **(b)** Well fit **(c)** Overfitting

Figure 2.33: Different types of model fit (Vieira and Paixao, 2018).

A well-fitted model produces a more accurate prediction of unseen data (Figure 2.33). However, the trained model does not always produce a good prediction due to problems in the model fit. There are two types of problems commonly encountered when trying to fit a model:

- **Overfitting**: Overfitting occurs when the model is unable to generate accurate predictions on validation samples, although it performs well on training samples. The loss plot shows that the training loss converges to a minimum, but the validation loss is minimized only to a certain point, before starting to increase (Figure 2.34). Overfitting means the model is memorizing the data it has seen and is unable to generalize to unseen data.

- **Underfitting**: Underfitting occurs when the model is unable to generate accurate predictions on validation samples due to poor performance on training samples. The loss plot generally shows that the training and validation loss have not converged to a certain minimum, and there is still a room of improvement to minimize the losses further (Figure 2.34). Underfitting means the model is not powerful enough; it is over-regularized or has not been trained long enough.



(a) Underfitting **(b)** Overfitting

Figure 2.34: Loss plot on model with underfitting and overfitting problem (Brownlee, 2017).

Rectification of Model Fit Problems

Model fit problems must be rectified in order to generate an accurate prediction. Overfitting can be avoided by implementing these solutions:

- **Adding regularization**: Regularization attempts to discourage the generation of a more complex model. Common regularization techniques include: Dropout, where random neurons are terminated by setting them to zero; and L2 regularization, where some constraints are added to the network by forcing its weights only to take on small values.

- **Early stopping**: In overfitting, the validation loss is minimized to a certain point before starting to rise. With early stopping, the training process is terminated based on the monitoring of validation loss. Once validation loss is not minimized even further, the training is stopped. The last model with the lowest loss is used for prediction on test results.
- **Data augmentation**: Data augmentation is a technique to enlarge training set by generating new samples via random transformation of the existing samples. Common transformations include rescaling, zooming, flipping and contrast change.

On the other hand, underfitting can be avoided with the following solutions:

- **Increasing hidden layers/neurons**: Increasing the number of hidden layers leads to more neurons computation might result in a more robust model. This allows the model to learn more complex features within the data.
- **Reducing regularization**: It is common to overdo regularization during the rectification of overfitting problem. When the loss plot shows a shift from overfitting to underfitting, it is advised to reduce the regularization to remove the underfitting effect.
- **Increasing training iteration**: More training iterations means the model is given more time to self-optimize and learn all the important features.
- **Data augmentation**: Data augmentation might also be implemented to rectify underfitting, as it allows the model to learn more variations with the addition of more samples.

2.5 Outcomes of the Preceding Study

Previously, in the preceding study by the same author, a machine learning-based net pay estimation method had been developed. The objective of the preceding study was comprised of the development of data processing procedures; development of machine learning implementation; and also preliminary net pay estimations on data from the wells of Johan Castberg field. The outcome of the preceding study served as the foundation for this study.

The outcomes of the preceding study was comprised of image logs processing procedures; core UV photographs processing procedures; design of machine learning implementation; and results of the preliminary net pay estimations. Each of the outcomes is outlined in the following section.

2.5.1 Image Logs Processing Procedures

In this study, image logs were used as the input for the neural network to perform net pay estimation. Image logs from Diskos database were raw unprocessed image logs. Therefore a series of processing measures were required prior to further utilization. The image logs processing steps were performed in Techlog software and aimed at reducing logging

environmental effects and acquisition artefacts. Besides processing, an additional data augmentation procedure using Python was also performed. Data augmentation aimed at generating more samples of the data based on the available data. The general workflow of image logs processing is outlined in Figure 2.35. Further details of the image logs processing procedures are explained in Section 3.4.

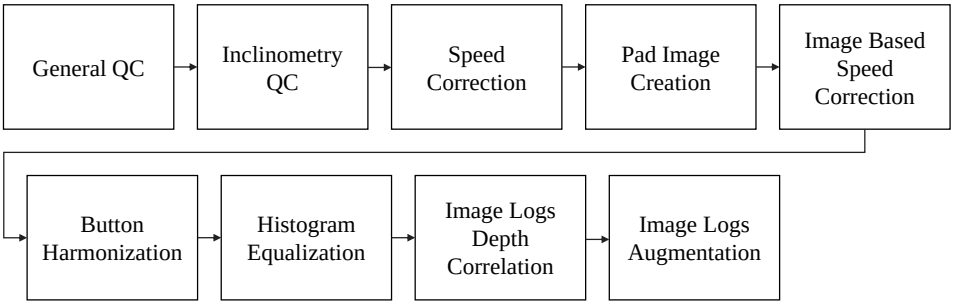


Figure 2.35: General workflow of image logs processing in the preceding study.

2.5.2 Core UV Photographs Processing Procedures

Core UV photographs were used in this study to generate net pay fraction labels for the machine learning technique. To generate net pay fraction labels, several processing steps must be performed. In general, core UV photographs were binarized using software called ImageJ. After core UV photographs binarization, data augmentation was performed similarly with image log augmentation. The final step was the net pay fraction calculation for each binarized image based on the color of the image. The net pay fraction values were used as the label for the machine learning technique. The general workflow of core UV photographs processing is outlined in Figure 2.35. Further details of the core UV photographs processing procedures are explained in Section 3.5.

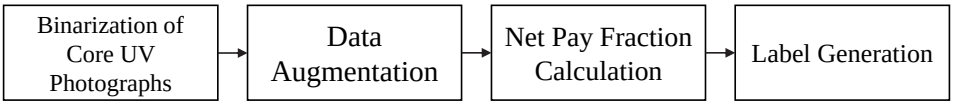


Figure 2.36: General workflow of core UV photographs processing in the preceding study.

2.5.3 Machine Learning Implementation

Machine learning implementation for net pay estimation was achieved by developing a convolutional neural network (CNN) algorithm. The convolutional neural network was developed using the Keras package in Python. A series configuration and hyperparameter tuning were performed to find a network design which gives optimum result in terms of loss minimization, model fit, and estimation accuracy. The architecture of CNN is outlined

in Table 2.5. Further details of the machine learning implementation are explained in Section 3.6.

Table 2.5: Configuration and hyperparameter settings of convolutional neural network in the preceding study.

Configuration	Description
Convolutional layer	Number of layer: 5 Number of filter: 16,32,64,128, and 256 (from layer 1 to 5) Filter size: 3 x 3 Activation function: ReLU Padding: Enabled
Pooling layer	Number of layer: 5 Pooling window size: 2 x 2 Padding: Enabled
Fully connected layer	Number of layer: 2 Number of neuron: 64 and 1 Activation function: ReLU and linear
Loss	Type: Mean squared error (MSE)
Optimization algorithm	Type: Adam Learning rate: 0.0001 Batch size: 64
Other parameter	Data augmentation: Horizontal and vertical flip Early stopping: Stop training after 5 iterations without loss minimization

2.5.4 Preliminary Net Pay Estimations Results

The capability of the net pay estimation method was observed through a set of preliminary net pay estimations. The term preliminary refers to how these experiments were intended to formulate optimization approaches that will be implemented in the continuation of the study. There were two sets of preliminary net pay estimations: net pay estimation based on data from the same well and net pay estimation on data from a different well.

Net Pay Estimation Based on Data From the Same Well

In this net pay estimation, the training, validation, and test samples were taken from well 7220/7-3 S. The dataset from well 7220/7-3 S was selected due to the significant presence of both clean and shaly sand intervals in its reservoir. The model used in this net pay estimation was referred as Model X. Table 2.6 outlines the details of the training process and also the test loss of estimating samples from well 7220/7-3 S. The result of net pay estimation based on data from well 7220/7-3 S in the preceding study is displayed in Figure 2.37.

Table 2.6: Details of the training process of Model X and the test loss achieved in estimating net pay based on data from well 7220/7-3 S.

Model	Details	Value
Model X	Iteration	14
	Training loss	0.0459
	Validation loss	0.0403
	Test loss	0.0417

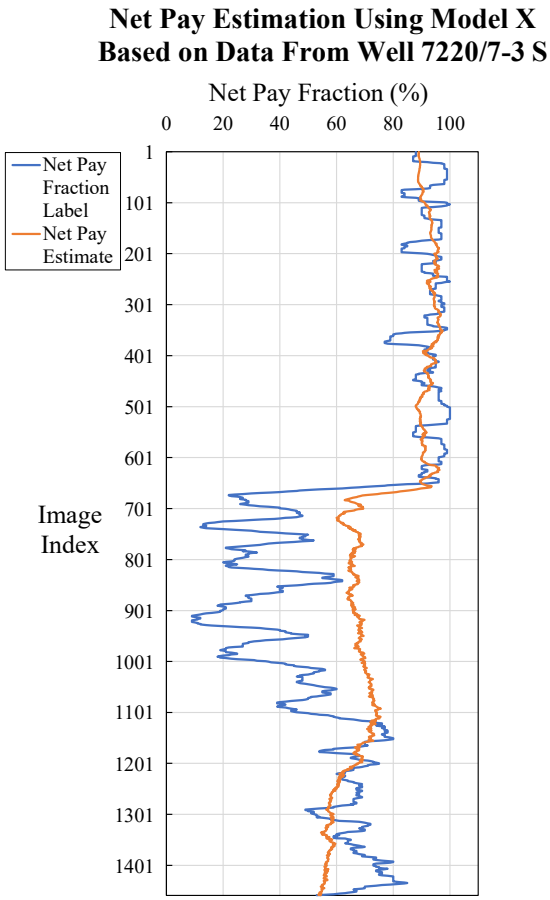


Figure 2.37: Net pay estimation based on data from well 7220/7-3 S in the preceding study.

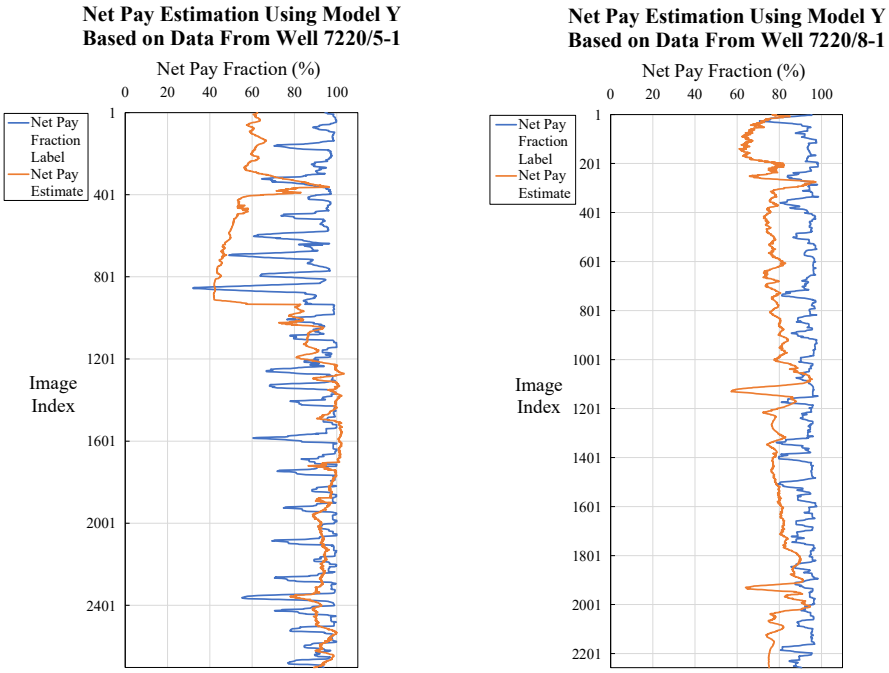
Net Pay Estimation on Data From a Different Well

In this net pay estimation, the training and validation samples were taken from well 7220/7-3 S and 7220/7-1. Two test sets were generated, one with samples from well

7220/5-1 and the other with samples from well 7220/8-1. The model used in this net pay estimation was referred to as Model Y. Table 2.7 outlines the details of the training process and also the test loss of estimating samples from well 7220/5-1 and 7220/8-1. The result of net pay estimation based on data from well 7220/5-1 and 7220/8-1 in the preceding study is displayed in Figure 2.38.

Table 2.7: Details of the training process of Model Y and the test loss achieved in estimating net pay based on data from well 7220/5-1 and 7220/8-1 .

Model	Details	Value
Model Y	Iteration	8
	Training loss	0.0459
	Validation loss	0.0276
	Test loss (7220/5-1)	0.0457
	Test loss (7220/8-1)	0.0740



(a) Well 7220/5-1.

(b) Well 7220/8-1.

Figure 2.38: Net pay estimation based on data from well 7220/5-1 and 7220/8-1 in the preceding study.

Chapter 3

Methodology

3.1 General Workflow

In the preceding study, a net pay estimation method from borehole image logs based on a machine learning principle was developed. The object of study was several sets of image logs and core data from exploration wells in the Johan Castberg field (c.f. Section 3.2). Image datasets comprised of image log samples served as the input for the machine learning technique, while core UV photographs were utilized to generate net pay fraction labels for the input. A set of processing procedures were developed to generate the image datasets from the raw image logs and net pay fraction labels from the core UV photographs. The machine learning technique itself was designed and developed in Python based on convolutional neural network (CNN).

The preliminary net pay estimations conducted in the preceding study provided several insights on how the machine learning-based net pay estimation method could be optimized. Based on those insights, several optimization approaches were formulated in this study. The optimization approaches consist of optimization of the image logs and core UV photographs processing procedures (c.f. Section 3.4 and 3.5) and an optimized machine learning technique implementation (c.f. Section 3.6). The goal of implementing those optimization approaches was to improve the estimation results in terms of accuracy and reliability.

The optimization of the image logs and core UV photograph processing was achieved by implementing modification on the existing processing procedures (c.f. Section 2.5.1 and 2.5.2) and introducing novel processing procedures. For the machine learning part, the optimization was attained by implementing modification in the convolutional neural network design; a better curation of samples in the training set; and introduction of an alternative data augmentation technique.

As seen in Figure 3.1, after the optimized data processing and machine learning implementation, a set of net pay estimations was conducted using the processed data and CNN model. One of the goals of the estimations was to observe and analyze the capability of the optimized CNN model in estimating the net pay of unseen image logs in the test set, in terms of accuracy and estimation profile. The reliability of the model was also evaluated through a series of estimations on test sets with a horizontal variation. Statistical analysis based on the calculation of standard deviation was performed to ensure that the estimation was independent of horizontal variation within the image logs. The optimized net pay estimation method was also demonstrated in estimating net pay of reservoir intervals without core photographs from one of the wells in the Johan Castberg field. Finally, based on the results evaluation and observed limitation of the net pay estimation method, several recommendations for the continuation of the study was formulated.

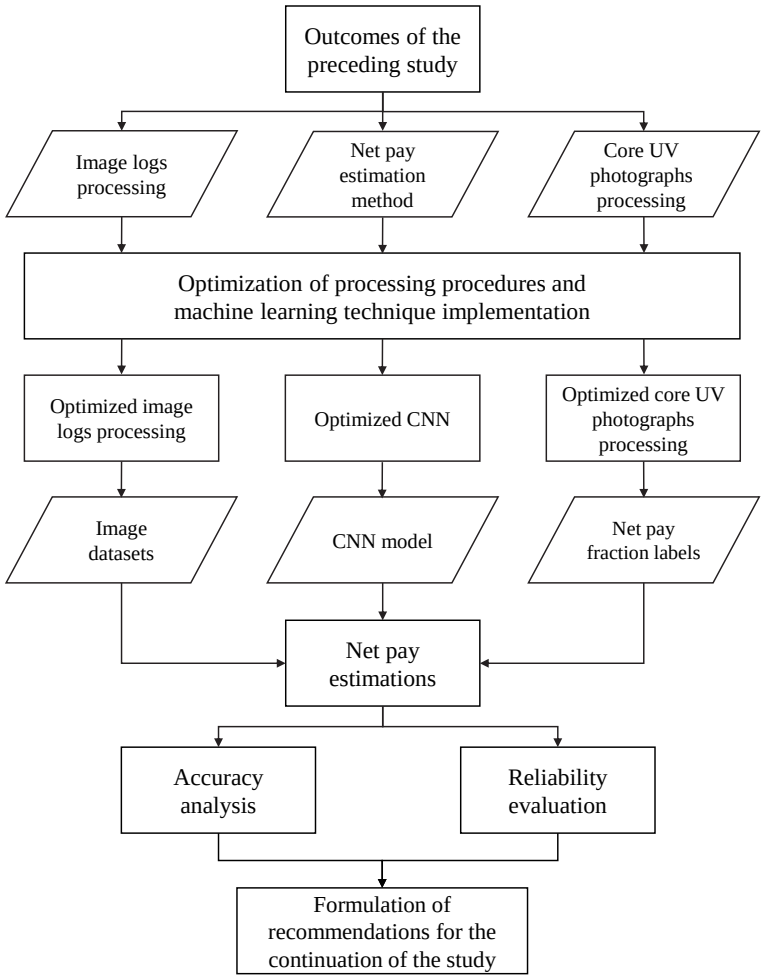


Figure 3.1: General workflow of the study conducted in this study.

3.1.1 Software

The software and programming languages which were utilized in this study were outlined below:

Techlog

Techlog is a software developed by Schlumberger to aid in petrophysical interpretation and analysis. Techlog enables basic and advanced petrophysical interpretation on all wellbore data types, including log, core, images, photos, and thin sections (Schlumberger, 2019b). Techlog allows the processing of all common data types from various logging tools and vendors. A Techlog suite, Advanced Borehole Geology Suite, was used to process image logs in this study. Advanced Borehole Geology Suite is designed for processing and interpretation of borehole image logs. It contains eleven plugins to perform interpretation and processing on borehole image logs, e.g. basic image log processing, facies analysis, rock texture, and fractures analysis, rock matrix conductivity analysis, extraction of formation dip, structural cross-section computation, and generation of 360-degree images (Schlumberger, 2019a). In this study, Techlog was used to perform the following tasks:

- Image log processing.
- Depth correlation between image logs and core UV photographs.
- Converting image logs into tabular array format in comma-separated values (CSV) files.

The version of Techlog used in this study was Techlog 2018.2.

ImageJ

ImageJ is an open-source image processing program developed by the National Institutes of Health and Laboratory for Optical and Computational Instrumentation (LOCI). ImageJ is a Java-based program and designed for standard image processing functions such as sharpening, smoothing, edge detection, median filtering, binarization, geometric transformation, contrast alteration, convolution, and Fourier analysis (NIH, 2019). ImageJ version 1.52a was utilized in this study for the following tasks:

- Core UV photographs scaling.
- Binarization of core UV photographs.

Python

Python is a high-level programming language developed by Python Software Foundation. It enables multiple programming applications, including functional and object-oriented programming. Machine learning scripts in this study were written in Python programming language. One of the Python packages, which was utilized to implement machine learning techniques in this study, was Keras. Keras is an open-source high-level neural networks application programming interface (API) developed by François Chollet, a Google software engineer. Python 3 and Keras 2.2.5 was used in this study for the following tasks:

- Converting core UV images into CSV files and Python array.
- Data augmentation on image logs and core UV photographs.
- Application of machine learning technique for net pay estimation.

3.2 Object of Study

3.2.1 Data Criteria and Composition

In the preceding study, several sets of data from exploration wells in Norway were acquired from the NPD Diskos database. Diskos is a Norwegian national data repository for exploration and production-related data, which is shared by both authorities and oil companies. It was initiated and designed by the Norwegian Petroleum Directorate (NPD) and oil companies represented on the Norwegian continental shelf in 1995 (NPD, 2019a). There were several criteria which were considered during the selection of the wells for data acquisition:

- Exploration wells with clean sand and shaly sand reservoir intervals.
- Exploration wells with hydrocarbon discovery.
- Exploration wells with available core and image log data.

The composition of the data from each well is outlined in Table 3.1.

Table 3.1: Data composition and file format for each dataset.

Data	Format	Details
Core UV photographs	Tagged Image File Format (TIFF)	Served as the ground truth for machine learning estimation. Labels for the machine learning algorithm were generated from this data.
Core laboratory report	Portable Document Format (PDF)	Utilized in the identification of core photo interval which contained no core photograph due to whole core preservation (seal peel).
Borehole image log	Digital Log Information Standard (DLIS) and PDF	Served as the input for the machine learning. Net pay estimation was performed based on the image log.

3.2.2 Johan Castberg Field

The information in this paragraph was summarized from the Johan Castberg webpage in the Norsk Petroleum website organized by the Norwegian Ministry of Petroleum and Energy and NPD (NPD, 2019b). The Johan Castberg field is an oil field operated by Equinor Energy AS and situated in the Barents Sea, Norway (Figure 3.2). It is located 100

kilometers northwest of the Snøhvit field. Johan Castberg consists of three discoveries: Skrugard, Havis, and Drivis. The discoveries were proven between 2011 and 2013. The reservoirs contain oil with gas caps in three separate sandstone deposits from Late Triassic to Middle Jurassic age in the Tubåen, Nordmela, and Stø Formations. They lie at depths of 1350 to 1900 m. The field is currently under development, and production is scheduled to start in late 2022.

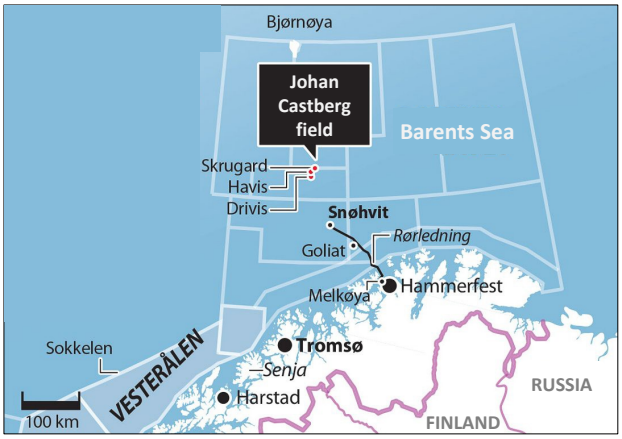


Figure 3.2: Location of Johan Castberg field in Barents Sea, Norway (Ifinnmark, 2019).

Data from four exploration wells in Johan Castberg was chosen as the object of this study. Johan Castberg field was selected due to its laminated sandstone reservoir characterized by ripple lamination and silty layers (Knaust, 2017). Two of the wells, 7220/7-3 S and 7220/7-1, contain both clean and shaly sand sequences in the reservoir interval. The rest of the wells, 7220/5-1 and 7220/8-1, contain mostly clean sand in the reservoir interval. Wells with both clean and shaly sand sequences were chosen as the main training data. This was done to ensure that the neural network model learns both clean and shaly sand sequences during training.

In this study, image logs and core UV photographs were curated from the oil zone, which is the interval between gas-oil contact (GOC) and oil-water contact (OWC). In the core UV photographs, oil bearing sands fluoresce within the oil zone. Oil fluorescence is essential for the generation of net pay fraction labels from core UV photographs. Further information and details on the selected wells are outlined in Table 3.2.

Table 3.2: Selected exploration wells from Johan Castberg field

Well Name	Discovery	Year	Total Depth	Oil Zone Interval
7220/8-1	Skrugard	2011	2222 m	43 m
7220/7-1	Havis	2011	1740 m	126 m
7220/5-1	Skrugard	2012	2097 m	32 m
7220/7-3 S	Drivis	2014	2230 m	78 m

3.3 Implemented Optimization Approaches

Several optimization approaches which were formulated in the preceding study were implemented in this study. These optimization approaches were implemented in several elements of this study, e.g. data processing procedures and machine learning implementation. Table 3.3 outlines the changes in the data processing procedures and the machine learning implementation due to the introduction of the optimization. Each of the optimization approaches is discussed further in Section 3.3.1 to 3.3.3.

Table 3.3: Changes in the data processing procedures and in the machine learning implementation due to the introduction of the optimization approaches.

Element	Preceding Study	This Study
Image logs processing	Button harmonization performed after image based speed correcton.	Button harmonization performed before and after image based speed correcton.
	Image log sub-intervals were stitched together during augmentation. Image log augmentation was performed on the stitched image.	Image log sub-intervals were not stitched together during augmentation. Image log augmentation was performed on each individual sub-intervals
	Orientation augmentation procedures: vertical and horizontal flip.	Orientation augmentation procedures: vertical flip, horizontal flip, and pixel based horizontal shift.
Core UV photographs processing	Image artefacts on core UV photographs were still present.	Image artefacts on core UV photographs were not present.
	Core UV photographs sub-intervals were stitched together during augmentation. Core UV image augmentation was performed on the stitched image.	Core UV photographs sub-intervals were not stitched together during augmentation. Core UV image augmentation was performed on each individual sub-intervals
Machine learning implementation	Less complex CNN design: 5 convolutional and pooling layers; 2 fully connected layers.	More complex CNN design: 6 convolutional and pooling layers; 6 fully connected layers.
	Calcite cemented sand intervals were present in the training set.	Calcite cemented sand intervals were removed in the training set.

3.3.1 Optimization Approaches for Image Log Processing

The optimization approaches which were introduced in the image log processing consist of the introduction of an additional button harmonization in the processing workflow; modification of the image log augmentation procedures; and introduction of an additional image log orientation augmentation parameter.

Introduction of an Additional Button harmonization

Generally, button harmonization is an effort to regularize the response of button electrodes or correct missing values due to faulty buttons (c.f. Section 2.2.2). However, in the case of excessive buttons misalignment, an additional button harmonization may be introduced to rectify the artefacts (Schlumberger, 2019a).In this study, an additional button harmonization was performed before image based speed correction. Image based correction itself is a part of image enhancement procedure which attempts to rectify offsets in the buttons (sawtooth effect) or pads misalignment (c.f. Section 2.2.2). By performing a button harmonization before image based speed correction, the response of button electrodes was regularized before the image log was corrected for sawtooth effect. Furthermore, another button harmonization was performed after image based speed correction, which further amplified the regularization of button electrodes response. Further details on button harmonization and image based speed correction are discussed in Section 3.4.5 and 3.4.6.

Modification of the Image Log Augmentation Procedures

One of the purposes of image log augmentation was to increase the number of samples from the available image log by introducing overlaps between each sample image. One of the steps in image log augmentation was the removal of image log intervals which did not possess corresponding core UV photographs due to seal peel intervals. Seal peel intervals were shown as blank sections in the core UV photographs; therefore, net pay fraction labels could not be generated from those intervals. This implies that there were some missing intervals within the image log, and the image log was divided into several unconnected sub-intervals. In the preceding study, those sub-intervals were stitched together during image log augmentation. Stitching those sub-intervals together produced an unrealistic structure as two distinctive features cut by the seal peel intervals were stitched together (Figure 3.3). This might cause CNN to learn unrealistic structures and thus result in an inaccurate estimation.

A solution to avoid this issue was by treating each sub-interval as an individual image log and performing image augmentation on each sub-interval. The image samples generated from each sub-interval were then combined in the image dataset. This ensured that each sample image did not contain combined structures, which was unrealistic. Further details on image log augmentation are discussed in Section 3.4.9.

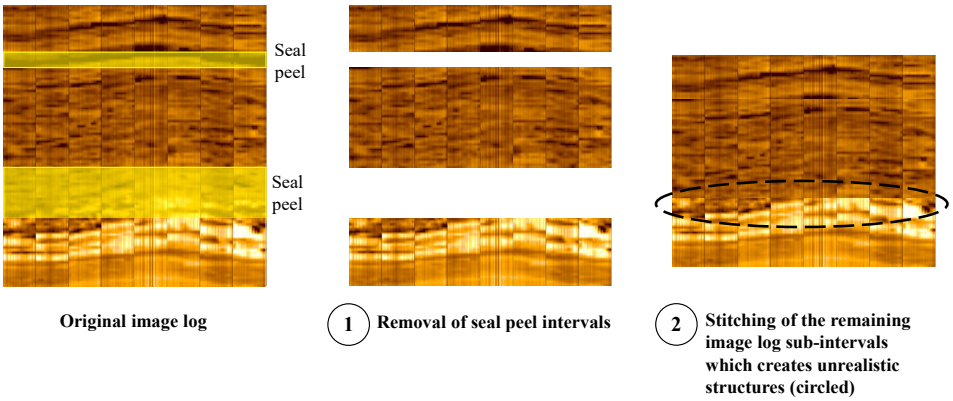


Figure 3.3: An example of how image concatenation during image log augmentation produces an unrealistic image log in the preceding study.

Introduction of An Additional Orientation Augmentation Procedure

Besides increasing the number of samples in an image dataset, image log augmentation was also purposed for introducing more variation within training samples, especially in terms of image orientation. By implementing image augmentation, image samples were generated with a higher degree of variation in the orientation. More variation on the orientation leads to a network which learns that the net pay was invariant to the orientation of the sand-shale layer. Orientation augmentation was achieved by implementing random orientation shift on each sample image, e.g. vertical shift, zooming, and horizontal flip.

In the preceding study, orientation augmentation was performed by implementing vertical and horizontal flip. In this study, the orientation augmentation was optimized with the introduction of pixel based horizontal shift. With pixel based horizontal shift augmentation, each sample was shifted horizontally based on a randomly chosen number of pixels. Further discussion on the orientation augmentation is shown in Section 3.4.9.

3.3.2 Optimization Approaches for Core UV Photographs Processing

The core UV photograph processing was optimized by introducing an additional processing step to improve the quality and accuracy of the net pay fraction labels. One of the drawbacks of core UV photographs is the presence of core plug holes in the image, which act as artefacts in the binarized image (Figure 3.4). Core plug holes were binarized into black color due to their dark shading in the original core UV images. If a core plug hole was present in a clean sand interval, it would be converted into a black circular area in the binarized image, which might reduce the net pay fraction of that interval. Hence a clean sand interval with a significant core plug hole might appear as a shaly interval in the core. By interpolating the interval with core plug holes in the label generation process, the generated net pay fraction label ideally became more accurate. The core plug holes were interpolated with textures of the area surrounding the holes. Beside the interpolation of

core plug holes, seal peel intervals were also removed as in the image log processing procedures. Further discussion on the core UV photographs processing is shown in Section 3.5.

Figure 3.4: An example of how binarization of a core plug hole resulted in a black circular area in the binarized image. The image is taken from well 7220/5-1.

3.3.3 Optimization Approaches for Machine Learning Implementation

The machine learning implementation was optimized by implementing a more complex neural network architecture and a better curation of samples in the training set.

Enhancement in CNN Architecture Complexity

One of the prevalent issues encountered in the preceding study was the model underfitting problem. A possible solution for the underfitting issue is by increasing the complexity of the CNN architecture. This was achieved by increasing the number of layers in the CNN architecture, e.g. convolutional layer, pooling layer, and layers in the fully connected layer. In the preceding study, the CNN architecture was comprised of five convolutional and pooling layers, and also two fully connected layers. In this study, the CNN architecture was comprised of six convolutional and pooling layers, and six fully connected layers.

The introduction of additional convolutional and pooling layers enabled the network to learn more features by employing more convolutional filters and feature maps. With more feature maps, the number of neurons required to perform the computation in the fully connected layer would also increase. The necessity of more neurons to perform computation and generate net pay estimation was fulfilled by introducing more fully connected layers. The CNN achieved better learning through the use of more complex architecture, and underfitting was minimized because the generated model was able to generalize better to unseen samples.

Better Curation of Samples in the Training Set

Generating a CNN model with a good generalization to different variation in the samples is essential. In this study, the image log samples were composed of either clean and or shaly sand intervals. To generate a model which was able to generate a good estimation

on both clean or shaly sand intervals, the composition of training set had to be curated in a way that it had a balanced composition of clean and shaly sand samples. An unbalanced composition might cause the model to generalize better to one type of sample, and thus resulting in underfitting and poor estimation of net pay.

The presence of calcite cemented sands layers might also reduce the accuracy of net pay estimation. As discussed previously in Section 3.5.1, calcite cemented sands commonly possess low porosity and low permeability due to precipitation of calcite cement in the pore space (Worden et al., 2019). As seen in Figure 3.5, a calcite cemented interval in an oil zone is generally displayed as a dark interval in core UV photograph because it does not contain oil. However, it might be displayed as a high resistivity interval (bright orange) in the image log due to the cemented pores inhibiting the flow of electric currents. Thus, calcite cemented sands were displayed with a similar color intensity as oil-bearing sand. The presence of samples with calcite cemented sands might confuse the network as it was trained with samples where bright color correspond to high net pay fraction. This implies that calcite cemented sands intervals must be identified and subsequently excluded from the training set. By excluding samples with calcite cemented sands from the training set, a better CNN learning was achieved, and a more accurate net pay estimation was produced.

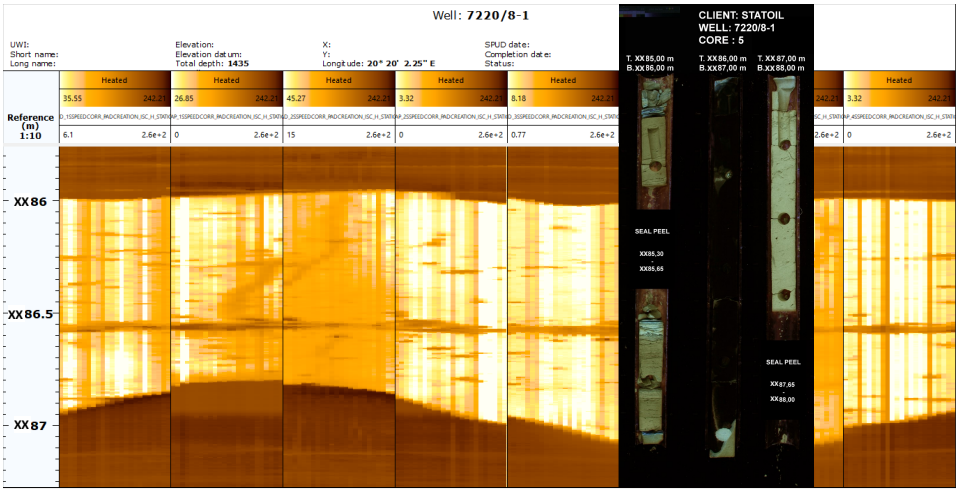


Figure 3.5: An interval of calcite cemented sands in well 7220/8-1 between XX86 – XX87 m with the corresponding core UV photograph as a comparison. The interval was displayed as a dark area in the core UV photograph. However, in the image log, it was displayed with a similar color intensity as oil-bearing sand.

3.4 Optimized Image Logs Processing

In this study, image logs were used as the input for the neural network to perform net pay estimation. These image logs were acquired using a Fullbore Micro-Imager (FMI) tool

by Schlumberger. As mentioned in Section 2.2.1, it consists of four pads with 24 button electrodes in each pad. The button electrodes are arranged into rows of 12 buttons, and each row is arranged with offsets. Each pad has a flap attached to it, and the flap increases the borehole coverage without introducing more caliper arms. The flaps function the same way as pads, and they also contain the same number of buttons. Thus, FMI contains four flaps (flap A-D), four pads (pad A-D) and 192 button electrodes.

The image logs from Diskos database were raw unprocessed image logs. Therefore a series of processing measures were required before further utilization. The image log processing steps were performed in Techlog and aimed at reducing logging environmental effects and acquisition artefacts. Image log processing also produced a standard display of the image log. Raw image log is comprised of 16 unprocessed stripes, and each stripe corresponds to one row of buttons in a pad or flap (Figure 3.6). These stripes were unprocessed, which means acquisition artefacts were still prevalent. The standard display of the image log consisted of 8 processed stripes, and each stripe represents a measurement from one pad or flap (Figure 3.7). Image log depth was also correlated with the depth of core photograph as the reference using Techlog. This ensured a matching depth between image logs and the corresponding core photographs.

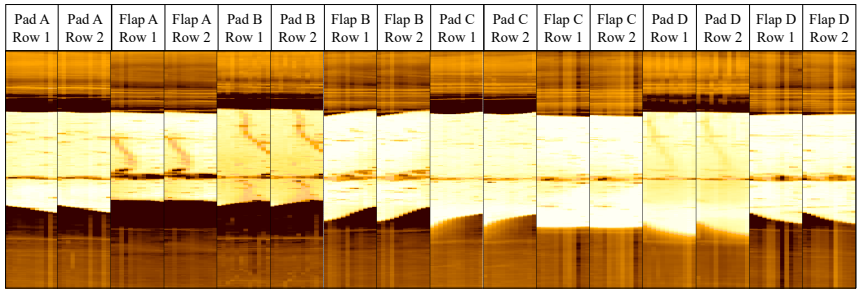


Figure 3.6: Raw image log from well 7220/8-1 showing 16 stripes of image.

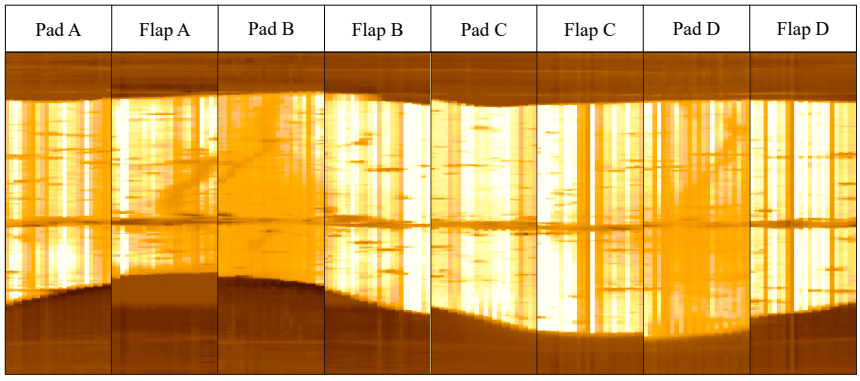


Figure 3.7: Processed image log from well 7220/8-1 showing 8 stripes of image.

Besides processing, an image log augmentation procedure using Python was also performed. Image log augmentation aimed at generating more samples and variation of the image log based on the available data. The number of samples is pivotal during the training process of a convolutional neural network. Ideally, the CNN model will have a better generalization with the introduction of more samples in the training set. The general workflow of image log processing is outlined in Figure 3.8. The details of each processing procedure are outlined in the following sections.

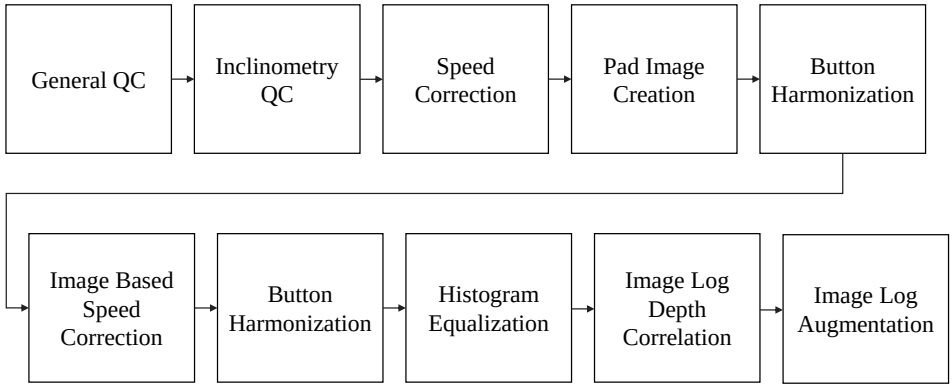


Figure 3.8: General workflow of image logs processing using Techlog and Python.

3.4.1 General Quality Control

The first step of image log processing in Techlog is observing the overall quality of the log and ensuring all necessary measurements were available. Generally, this step includes the following tasks:

- Extraction of necessary measurements from the Digital Log Information Standard (DLIS) file.
- General overview of the image log to ensure there were no missing intervals.
- Identification of zones with excessive noise or artefacts.
- Identification of zones with poor hole condition (washout or swelling).
- Caliper reading verification.

Individual measurements are referred to as channels in Techlog. The required channels for image log processing are outlined in Table 3.4. Caliper readings were also verified during general QC. Caliper correction is usually performed in the field by the oilfield service company. The correction was verified by looking at the correction value in the log remarks section to ensure that the correction had been performed.

Table 3.4: Required channels for image log processing.

Channel	Description
C1	Caliper 1
C2	Caliper 2
DEVIM	Memorized Deviation
EV	Electric voltage
FBGA	Electronic gain
FCA1	Electrode buttons measurement from pad A, row 1
FCA2	Electrode buttons measurement from pad A, row 2
FCA3	Electrode buttons measurement from flap A, row 1
FCA4	Electrode buttons measurement from flap A, row 2
FCAX	High resolution x-axis acceleration
FCAY	High resolution y-axis acceleration
FCAZ	High resolution z-axis acceleration
FCB1	Electrode buttons measurement from pad B, row 1
FCB2	Electrode buttons measurement from pad B, row 2
FCB3	Electrode buttons measurement from flap B, row 1
FCB4	Electrode buttons measurement from flap B, row 2
FCC1	Electrode buttons measurement from pad C, row 1
FCC2	Electrode buttons measurement from pad C, row 2
FCC3	Electrode buttons measurement from flap C, row 1
FCC4	Electrode buttons measurement from flap C, row 2
FCD1	Electrode buttons measurement from pad D, row 1
FCD2	Electrode buttons measurement from pad D, row 2
FCD3	Electrode buttons measurement from flap D, row 1
FCD4	Electrode buttons measurement from flap D, row 2
FTIM	High resolution acquisition time
FX	X-Axis magnetometer
FY	Y-Axis magnetometer
FZ	Z-Axis magnetometer
HAZIM	Memorized hole azimuth
P1AZ	Pad 1 azimuth in horizontal plane
RB	Relative bearing

3.4.2 Inclinometry Quality Control

As mentioned in Section 2.2.2, inclinometry quality control (QC) attempts at verifying and repairing inclinometer measurements. In Techlog, inclinometry QC is performed by plotting accelerometer and magnetometer channels and observing the plot. Ideally, the accelerometer readings should lie in the middle of the plot, and the magnetometer readings should form a circular shape in a vertical well. Techlog would attempt at calculating the correction needed if erratic readings were observed. It also recomputes the tool orientation channels. The required channels for inclinometry QC are outlined in Table 3.5.

Table 3.5: Required channels for inclinometry QC

Channel	Description
DEVIM	Memorized Deviation
FCAX	High resolution x-axis acceleration
FCAY	High resolution y-axis acceleration
FCAZ	High resolution z-axis acceleration
FX	X-Axis magnetometer
FY	Y-Axis magnetometer
FZ	Z-Axis magnetometer
HAZIM	Memorized hole azimuth
P1AZ	Pad 1 azimuth in horizontal plane
RB	Relative bearing

The results from inclinometry QC were several reliable inclinometer channels, which was utilized in the subsequent processing steps.

3.4.3 Speed Correction

After a series of QC, the next processing step is speed correction. Speed correction corrects for the effect of tool sticking on the image log. Tool sticking happens when the tool stops momentarily and move upward in an abrupt manner. Tool sticking might cause some intervals of the log to appear smeared.

Speed correction was performed by utilizing a sticking detection threshold based on the z-axis acceleration channel (FCAZ). If the variation on the FCAZ falls below this threshold, the interval was considered to be potentially stuck. This was verified by checking for deceleration before the stuck interval, and acceleration after the stuck interval. The default value of stuck detection threshold is 0.01 m/s^2. If sticking was detected, the tool velocity would be set to zero over the stuck intervals, and the measurements acquired over the stuck interval were shifted to the depth where the sticking started. The required channels for speed correction are outlined in Table 3.6.

Table 3.6: Required channels for speed correction

Channel	Description
FCA1	Electrode buttons measurement from pad A, row 1
FCA2	Electrode buttons measurement from pad A, row 2
FCA3	Electrode buttons measurement from flap A, row 1
FCA4	Electrode buttons measurement from flap A, row 2
FCAZ	High resolution z-axis acceleration
FCB1	Electrode buttons measurement from pad B, row 1
FCB2	Electrode buttons measurement from pad B, row 2
FCB3	Electrode buttons measurement from flap B, row 1
FCB4	Electrode buttons measurement from flap B, row 2

Table 3.6 (Continued)

Channel	Description
FCC1	Electrode buttons measurement from pad C, row 1
FCC2	Electrode buttons measurement from pad C, row 2
FCC3	Electrode buttons measurement from flap C, row 1
FCC4	Electrode buttons measurement from flap C, row 2
FCD1	Electrode buttons measurement from pad D, row 1
FCD2	Electrode buttons measurement from pad D, row 2
FCD3	Electrode buttons measurement from flap D, row 1
FCD4	Electrode buttons measurement from flap D, row 2
FTIM	High resolution acquisition time

The output from this processing step was image log channels, which had been depth shifted based on the speed correction mechanism. Ideally, the application of speed correction generates image logs with a reduced smearing effect.

3.4.4 Pad Image Creation

Pad based image logging tools have a variety of hardware designs that differs from one vendor to another. Some tools have four caliper arms, while others have six arms. The number and placement of the electrode buttons may vary. As a result, raw data measured from the tools are stored using different conventions. As an example, Fullbore Micro-Imager (FMI) tool by Schlumberger produces 16 image arrays, which corresponds to 16 rows of button electrodes (Figure 3.6). Meanwhile, Simultaneous Acoustic and Resistivity Imager (STAR) tool from Baker Hughes produces eight image arrays in which one array corresponds to two rows of button electrodes in a pad. Raw image logs still display offsets between rows of button electrodes and offsets between pads and flaps. Therefore, a standardized display was required, and it was achieved through pad image creation.

Pad image creation generally converts raw image logs into image logs with a standardized display. Pad image creation also enables visualization of the image log before the application of orientation information into the log. Techlog uses the following conventions to display image logs:

- All pads are ordered clockwise around the borehole when looking down the borehole
- All buttons are ordered clockwise around the borehole when looking down the borehole
- All button and pad measurements are corrected for any depth offsets during acquisition
- Processed image logs are displayed with eight stripes of image

The required channels for pad image creation are outlined in Table 3.7.

Table 3.7: Required channels for pad image creation.

Channel	Description
C1	Caliper 1
C2	Caliper 2
EV	Electric voltage
FBGA	Electronic gain
FCA1_S	Speed corrected electrode buttons measurement from pad A, row 1
FCA2_S	Speed corrected electrode buttons measurement from pad A, row 2
FCA3_S	Speed corrected electrode buttons measurement from flap A, row 1
FCA4_S	Speed corrected electrode buttons measurement from flap A, row 2
FCB1_S	Speed corrected electrode buttons measurement from pad B, row 1
FCB2_S	Speed corrected electrode buttons measurement from pad B, row 2
FCB3_S	Speed corrected electrode buttons measurement from flap B, row 1
FCB4_S	Speed corrected electrode buttons measurement from flap B, row 2
FCC1_S	Speed corrected electrode buttons measurement from pad C, row 1
FCC2_S	Speed corrected electrode buttons measurement from pad C, row 2
FCC3_S	Speed corrected electrode buttons measurement from flap C, row 1
FCC4_S	Speed corrected electrode buttons measurement from flap C, row 2
FCD1_S	Speed corrected electrode buttons measurement from pad D, row 1
FCD2_S	Speed corrected electrode buttons measurement from pad D, row 2
FCD3_S	Speed corrected electrode buttons measurement from flap D, row 1
FCD4_S	Speed corrected electrode buttons measurement from flap D, row 2

In this study, the raw image log data consists of 16 image arrays, and each consists of 12 array columns, which corresponding to 12 buttons. During pad image creation, they were converted into eight image arrays, which consists of 24 array columns, which corresponds to 12 buttons (Figure 3.7). The depth offset between the pads and button electrodes were corrected through the process of pad image correction. Since orientation information was not required for net pay estimation, the standard display from pad image creation was sufficient for net pay interpretation.

3.4.5 Button Harmonization

Button electrodes might give irregular responses due to tool problems or borehole conditions, such as poor pad contact or mudcake smears on the pads. Faulty button electrodes might give no responses, which creates empty values in the image log array. Button harmonization corrects issues of button electrodes responses and missing measurement values due to faulty button electrodes.

If one or more faulty button electrodes displayed inconsistent responses that were not caused by lithological variation, a button harmonization was implemented to regularize the response of the faulty buttons by matching them to a global response of all the buttons. Because the algorithm applied a shift and a gain to the data; and there was no interpolation or filtering; there was only a little loss of resolution or data fidelity. In the case of faulty

button electrodes, the algorithm attempts to find dead button electrodes by looking for intervals where the button electrode measurements remain below a threshold over a defined interval. If such values were found, they would be replaced by interpolation with their neighbors.

In the case of excessive buttons misalignment or sawtooth effect, button harmonization can also be performed twice, before and after image based speed correction. This implies that the button electrodes response were regularized before supplying the image log to image based speed correction, which attempted to correct buttons misalignment. Performing button harmonization twice also amplified the regularization of button electrodes response during the processing workflow. Button harmonization requires the following channels as input:

Table 3.8: Required channels for button harmonization.

Channel	Description
FLAP_1	Image array from flap A
FLAP_2	Image array from flap B
FLAP_3	Image array from flap C
FLAP_4	Image array from flap D
PAD_1	Image array from pad A
PAD_2	Image array from pad B
PAD_3	Image array from pad C
PAD_4	Image array from pad D

The result of button harmonization was an updated pad or flap image arrays, which were used in the next processing step.

3.4.6 Image Based Speed Correction

Speed correction might not always manage to entirely correct for irregular tool movement. Since button electrodes are geometrically arranged with offset, irregular tool movement might display the offset as a sawtooth effect or buttons misalignment in the image log. Irregular tool movement might also create a depth mismatch between pads along with a certain depth. To correct those artefacts, image based speed correction was performed in Techlog.

Image based speed correction utilizes a depth correlation algorithm based on the standardized image arrays from pad image creation. The term ”image-based” refers to how the algorithm attempts to correct sawtooth effect or pads misalignment based solely on image arrays, without utilizing speed or accelerometer measurements. It requires the following channels as input:

Table 3.9: Required channels for image based speed correction.

Channel	Description
FLAP_1_H	Image array of flap A from button harmonization
FLAP_2_H	Image array of flap B from button harmonization
FLAP_3_H	Image array of flap C from button harmonization
FLAP_4_H	Image array of flap D from button harmonization
PAD_1_H	Image array of pad A from button harmonization
PAD_2_H	Image array of pad B from button harmonization
PAD_3_H	Image array of pad C from button harmonization
PAD_4_H	Image array of pad D from button harmonization

The result from image based correction was an updated pad or flap image arrays, which were used in another button harmonization. The second button harmonization was performed similarly as the first one (c.f. Section 3.4.5). It requires the following channels as input:

Table 3.10: Required channels for the second button harmonization.

Channel	Description
FLAP_1_H_ISC	Image array of flap A from image based speed correction
FLAP_2_H_ISC	Image array of flap B from image based speed correction
FLAP_3_H_ISC	Image array of flap C from image based speed correction
FLAP_4_H_ISC	Image array of flap D from image based speed correction
PAD_1_H_ISC	Image array of pad A from image based speed correction
PAD_2_H_ISC	Image array of pad B from image based speed correction
PAD_3_H_ISC	Image array of pad C from image based speed correction
PAD_4_H_ISC	Image array of pad D from image based speed correction

The result from second button harmonization was used in histogram equalization.

3.4.7 Histogram Equalization

Image log interpretation generally requires good image contrast to highlight the features or layers in the formation. Image contrast can be enhanced by normalizing the image log through the use of the histogram equalization technique. Histogram equalization accomplishes this by spreading out the most frequent color intensity values, i.e. stretching out the intensity range of the image. This enables a contrast enhancement of the logs, which aids visual interpretation by petrophysicists or geologists.

In histogram equalization, each measurement value is mapped to a position on a normalized scale, e.g. a scale that ranges from 0 to 255. There are two types of normalization in histogram equalization:

- **Static normalization**: Each measurement value is mapped to a scale normalized for the entire logging interval. Global contrasts are enhanced as a result.

- **Dynamic normalization**: Each measurement value is mapped to a scale defined by the range in a window with pre-defined size (usually 1 m). Local contrasts are enhanced as a result.

The required channels for histogram equalization are outlined in Table 3.11.

Table 3.11: Required channels for histogram equalization.

Channel	Description
FLAP_1_H_ISC_H	Image array of flap A from the second button harmonization
FLAP_2_H_ISC_H	Image array of flap B from the second button harmonization
FLAP_3_H_ISC_H	Image array of flap C from the second button harmonization
FLAP_4_H_ISC_H	Image array of flap D from the second button harmonization
PAD_1_H_ISC_H	Image array of pad A from the second button harmonization
PAD_2_H_ISC_H	Image array of pad B from the second button harmonization
PAD_3_H_ISC_H	Image array of pad C from the second button harmonization
PAD_4_H_ISC_H	Image array of pad D from the second button harmonization

One of the drawbacks of dynamic normalization is it might show thin layers in the image log, which are not physically present in the formation. That might lead to an incorrect net pay estimation due to additional layers of sand or shale which are not present in the corresponding core UV photograph. The advantage of using static normalization is to ensure a universal color assignment for the whole interval of the log, instead of local color assignment in dynamic normalization. In static normalization, a specific color is assigned to a particular pixel value, which is independent of depth. Thus, static normalization was used in this study.

The result from histogram equalization was a set of updated image arrays that were ready to be displayed for interpretation. The arrays can be displayed as a complete image log, as shown in Figure 3.9.

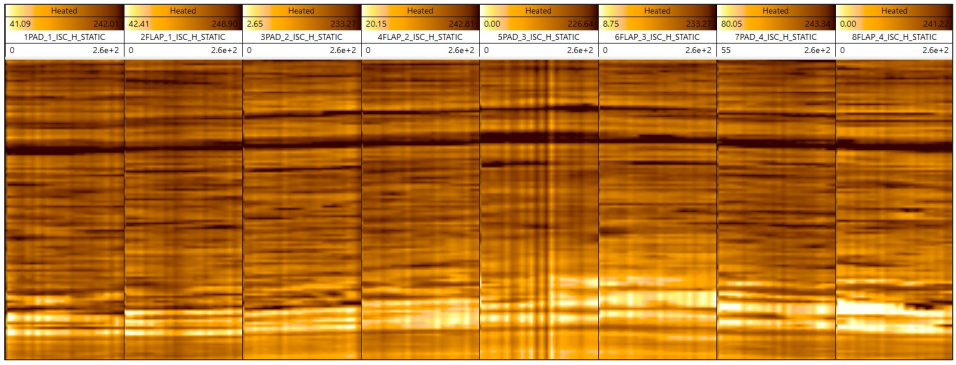


Figure 3.9: An example of image log processed with static normalization from well 7220/7-3 S.

3.4.8 Image Log Depth Correlation

Image logs and the corresponding core UV photographs had depth offsets, which had to be corrected. The offset was corrected by correlating the depth of image logs with the depth of core UV photographs as the reference. Depth correlation was performed by matching the depth of visually distinct geological features in the oil zone (Figure 3.10). Table 3.12 outlines the geological features which were utilized for depth correlation in this study.

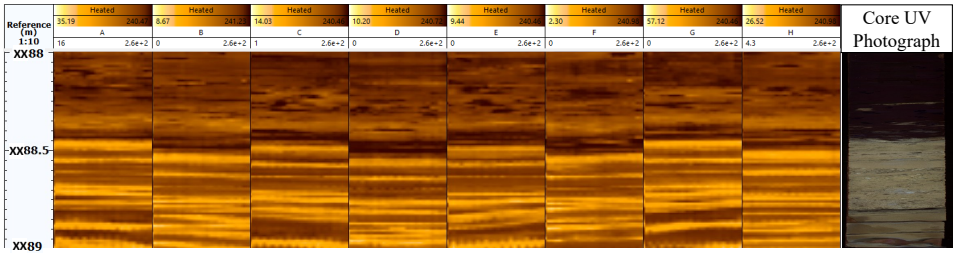


Figure 3.10: An example of depth correlation based on the top of clean sand interval at XX88.45 m from well 7220/7-1.

Table 3.12: Geological features utilized in depth correlation and their characteristics in image logs and core UV photographs

Geological features	Characteristic in image logs	Characteristic in core UV photographs
Clean sand interval	High resistivity interval (bright orange or yellow)	Bright interval
Clean shale interval	Low resistivity interval (dark brown)	Dark interval
Shaly sand sequence	Alternating layers of high and low resistivity interval	Alternating layers of bright and dark interval
Calcite cemented sand interval	High resistivity interval (bright orange or yellow)	Dark interval

After depth matching the top or bottom of a certain geological feature, a depth correction value was computed for each well. Image log from each well was depth shifted by applying the depth correction value in Techlog. Image log arrays from one well was combined into an image array with 192 columns, and certain number of rows which varies for each well. It was then exported from Techlog into a comma separated value (CSV) file for the final processing step in Python.

3.4.9 Image Log Augmentation

In order to generate more samples with more variation within the image dataset, an image augmentation procedure was a necessity during the generation of an image dataset. Image log augmentation is an attempt to generate more image log samples to populate the

image dataset, only with the available image logs. In this study, there were two goals of implementing image log augmentation: image dataset enlargement and enrichment of the variation within the dataset.

Image Dataset Enlargement

Before being used as the input for the machine learning technique, each processed image log went through a set of image augmentation procedures, in which it was sliced into smaller sized images with an overlapping portion between each image. By introducing image overlaps, a larger image dataset with more images was achieved by using only the available image logs. This implies that a particular layer might appear multiple times in a series of consecutive images, albeit with different vertical position. Image overlaps also enhance the CNN ability to learn the net pay of an image log sample regardless of the layer position. A larger dataset enables the model to generalize better to unseen samples and thus reducing model fitting problems such as overfitting and underfitting.

Image augmentation was conducted in Python and generated a dataset with a multitude of images. The procedure of image augmentation is shown in Figure 3.11, and each step is explained below:

1. Interval above the GOC and below the OWC was removed from the image log. This was to remove image log intervals with no oil fluorescence in the corresponding core UV photographs.

2. Image log intervals which correspond to seal peel intervals in the core UV photograph were removed. Seal peel intervals appear as blank sections in the core UV photographs (Figure 3.12). Therefore, net pay fraction labels could not be computed from those intervals. Image log sub-intervals were generated.

3. Each sub-interval was sliced into smaller sized images through the use of an image slicing window. The slicing window size was 100 pixels long and 192 pixels wide. The length of 100 pixels corresponds to 25 cm depth. Thus each image represents a 25 cm interval of the formation.

4. The slicing window moved 4 pixels (1 cm) down after slicing an image. The window sliced another image after moving 4 pixels down. This implies that the top depth of an image was not the same as the previous image bottom depth. As a result, each image was overlapping with the subsequent image by 96%.

5. Each image was flattened from a 2D array of 100 x 192 pixels into a 1D array of 1 x 19200 pixels. Each array was stored as a single row in the final array, which served as the input for the machine learning technique. The number of rows in the final array corresponds to the number of images in each well.

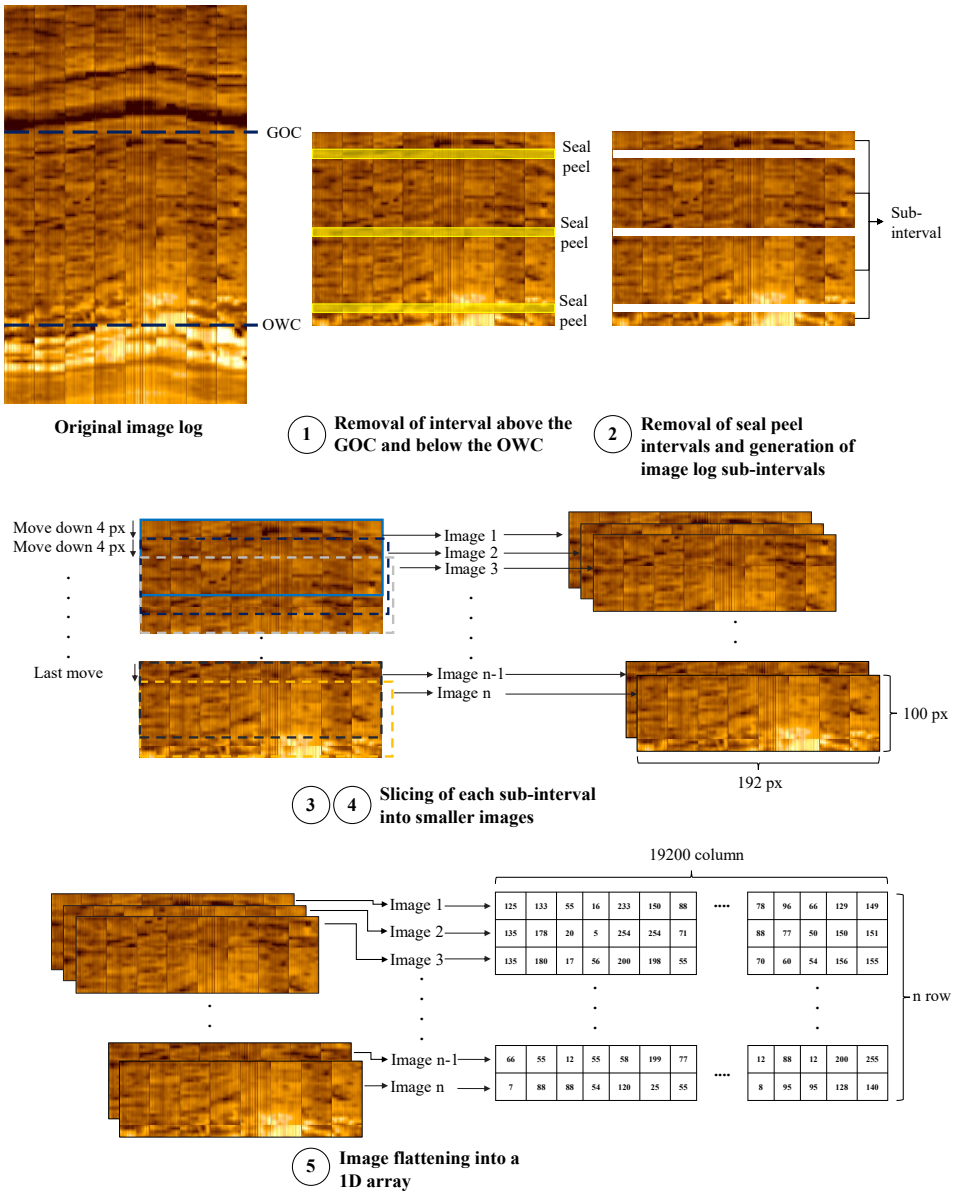


Figure 3.11: An illustration of processing steps for image logs augmentation.

Figure 3.12: An example of a core photograph with seal peel interval from well 7220/7-3 S. The interval contains no core image; therefore, a net pay fraction label cannot be generated from that interval.

Enrichment of the Variation Within a Dataset

Once an image dataset was generated with a sufficiently large number of samples, the orientation of samples was altered and augmented by implementing another image augmentation procedure. By augmenting the orientation of samples, more variation would be embedded into the dataset. This was essential for the samples in the training set as it taught the network that net pay was invariant to the orientation of the sand-shale layer. Orientation variation within the training set was achieved by using an image augmentation function in Python called Image Data Generator. This augmentation was implemented on samples in training set during the training process.

The orientation procedures considered in this study were vertical flip, horizontal flip, and pixel based horizontal shift. Image Data Generator randomly selected which procedure would be applied to a sample image. This implies that each sample image in the training set might have a different orientation, further enhancing the variation within the dataset. Vertical flip flipped the samples along the x-axis, while the horizontal flip flipped the samples along the y-axis. With pixel based horizontal shift, each sample was shifted horizontally based on a randomly chosen number of pixels. In this study, the range of pixels was specified between -96 to +96 pixels. Several examples of images with orientation variation implemented using Image Data Generator are shown in Figure 3.13.

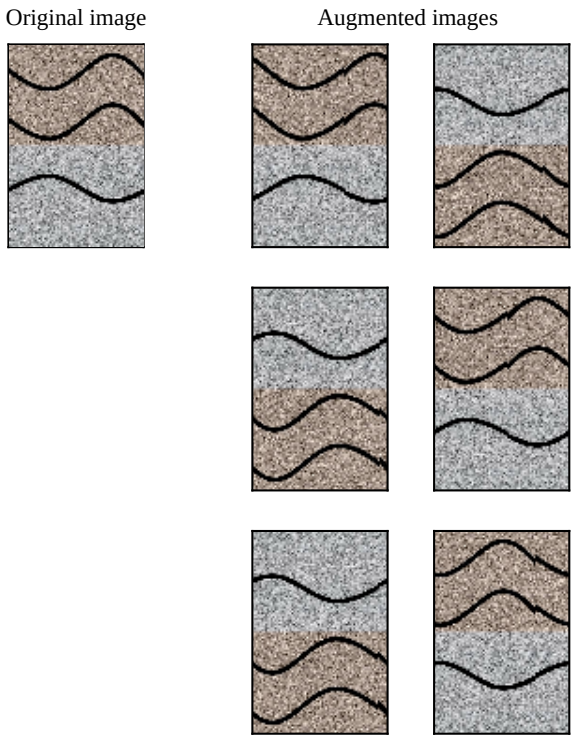


Figure 3.13: Several examples of artificial images with orientation variation implemented using Image Data Generator, e.g. vertical flip, horizontal flip and pixel based horizontal shift.

3.5 Optimized Core UV Photographs Processing

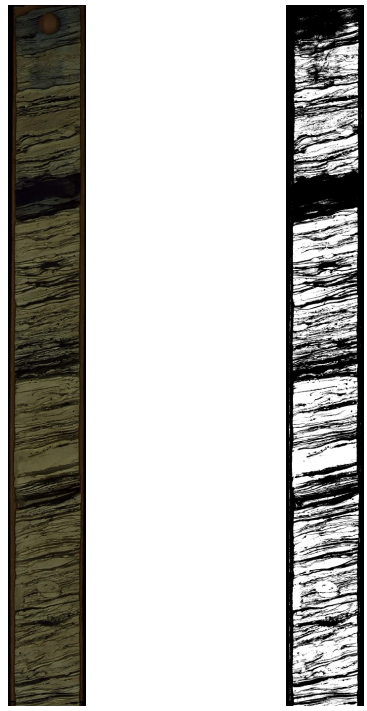
Core UV photographs were used in this study to generate net pay fraction labels for the machine learning technique. To generate net pay fraction labels, several processing steps must be performed. In general, core UV photographs were binarized using ImageJ. After core UV photographs binarization, data augmentation was performed similarly with image log augmentation. The final step was the net pay fraction calculation for each binarized image based on the color of the image. The net pay fraction values were used as the label for the machine learning technique.

3.5.1 Binarization of Core UV Photographs

To generate the net pay fraction for a specific interval of core UV photograph, a visual interpretation of sand and shale in the oil zone was needed (c.f. Section 2.3.3). Interpretation of sand and shale based on core UV photographs was performed by observing the color intensity of the image. In the oil zone, intervals with dark color were interpreted as shales or calcite cemented sands, while bright intervals were interpreted as oil bearing sands. Calcite cemented sands commonly possess low porosity and low permeability due

to precipitation of calcite cement in the pore space (Worden et al., 2019). Hydrocarbons such as oil generally do not migrate into layers of calcite cemented sands, and thus it is shown with dark color in the core UV photographs.

Instead of using a manual interpretation to determine which interval was considered as dark or bright, a binarization technique using ImageJ was utilized to convert the color intensity of the image into two colors, black and white (Figure 3.14). In this case, black pixels represent shale, and white pixels represent oil bearing sand. This enabled a straightforward net pay estimation for a specific interval, as the number of white pixels was only summed up and divided by the total number of pixels of the interval to obtain net pay fraction.



(a) Original **(b)** Binarized

Figure 3.14: Examples of (a) an original core UV photograph; and (b) a binarized core UV photograph from well 7220/7-3 S.

Core UV photographs from the Diskos database consist of multiple images, and each image represents 1 m interval of the core. The original dimension of each core UV image from the Diskos database was 9000 x 992 pixels, and it was reduced to 1000 x 110 pixels in ImageJ. This was to compress the size of the image and to enable a straightforward depth matching as 1 pixel corresponds to 0.1 cm after compression. Prior to binarization, the UV images were concatenated vertically into a single image using ImageJ. ImageJ binarized

the concatenated image by automatically setting a threshold based on the maximum and minimum pixel value in the image. Any pixel values below the threshold were converted to black, and pixel values above the threshold were converted to white. ImageJ binarization assigns pixel value of 255 for black and 0 for white. The result was a concatenated image for each well, with pixel values of 0 (oil bearing sand) or 255 (shale).

3.5.2 Core UV Images Augmentation

Similar to image logs processing, a set of image augmentation procedures were performed on the binarized core UV images. Each binarized core UV image underwent an image augmentation process where it was sliced into smaller sized images with overlaps (Figure 3.15). Image augmentation generated a dataset that contains a multitude of images using Python. The processing steps for image augmentation is listed below:

1. Interval above the GOC and below the OWC was removed from the core UV image. This was performed to remove the intervals with no oil fluorescence.

2. Seal peel intervals were removed as they were shown as blank sections. Net pay fraction labels could not be computed from those intervals. Core plug holes were also interpolated with textures of the area surrounding the holes. After the removal of seal peel intervals and interpolations of core plug holes, core UV image sub-intervals were generated.

3. Each sub-interval was sliced into smaller sized images through the use of an image slicing window. The slicing window size was 250 pixels long and 110 pixels width. The length of 250 pixels corresponds to 25 cm depth. Thus each image represents a 25 cm interval of the core.

4. The slicing window moved 10 pixels (1 cm) down after slicing an image. The window sliced another image after moving 10 pixels down. This implies that the top depth of an image was not the same as the previous image bottom depth. As a result, each image was overlapping with the subsequent image by 96%. The purpose of the overlap was to enlarge the existing dataset as an improvement in neural network model learning.

5. Each image was flattened from a 2D array of 250 x 110 pixels into a 1D array of 1 x 27500 pixels. Each array was stored as a single row in the final array. The number of rows in the final array corresponds to the number of the image in a dataset. Each well was represented by a dataset with a certain number of images.

The result from image augmentation was an array saved in CSV format.

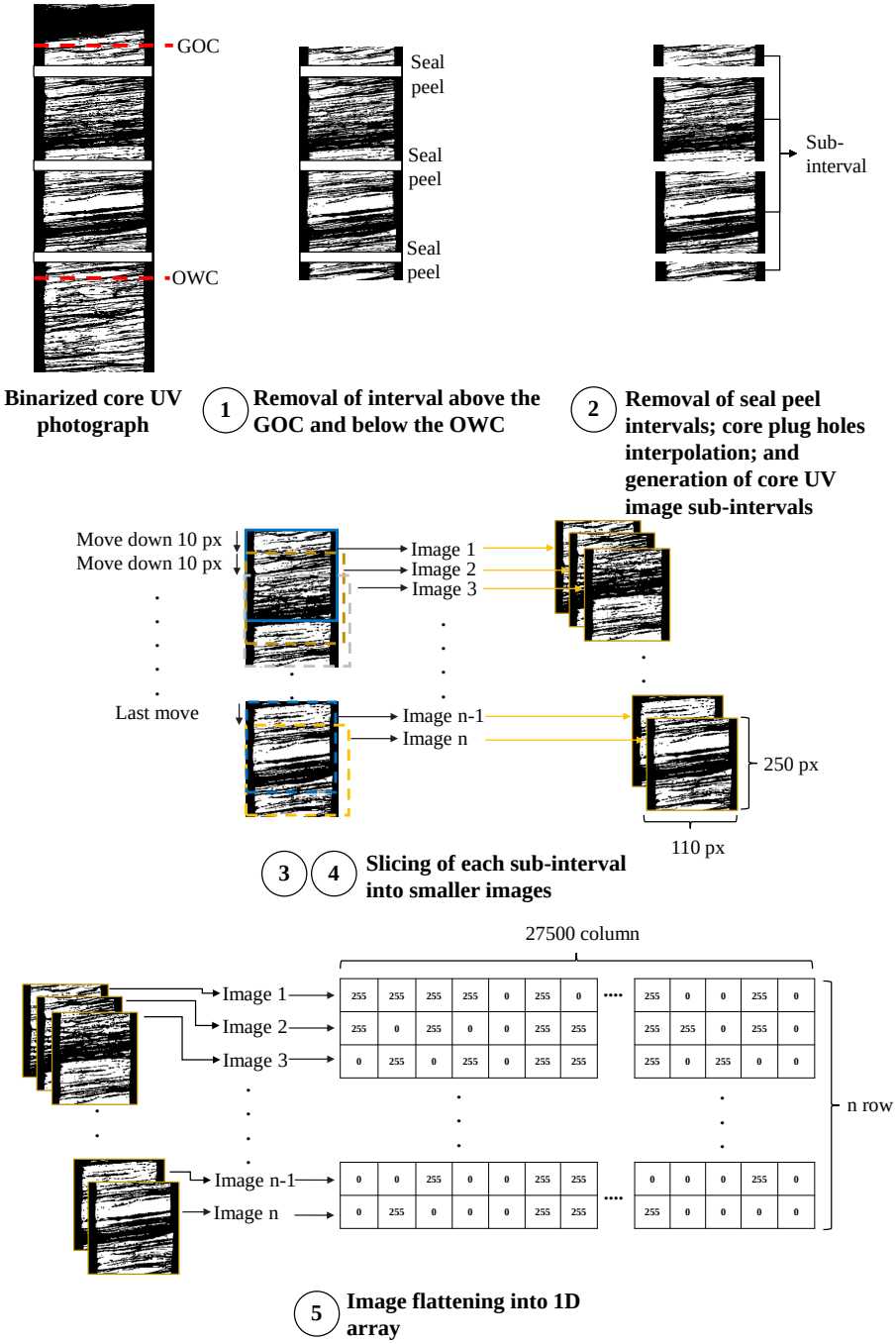
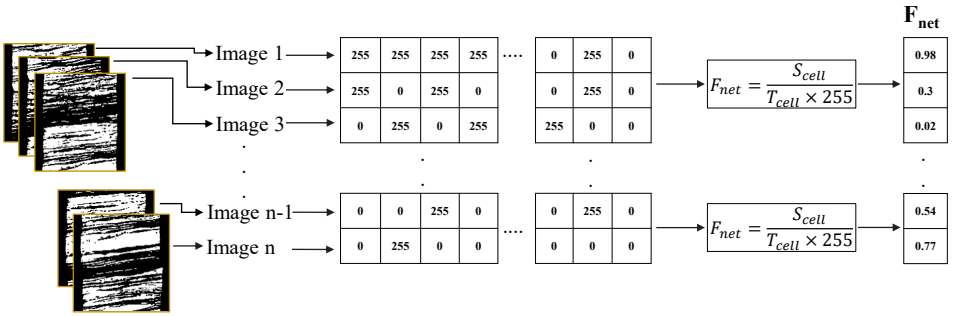


Figure 3.15: An illustration of processing steps for core UV images augmentation.

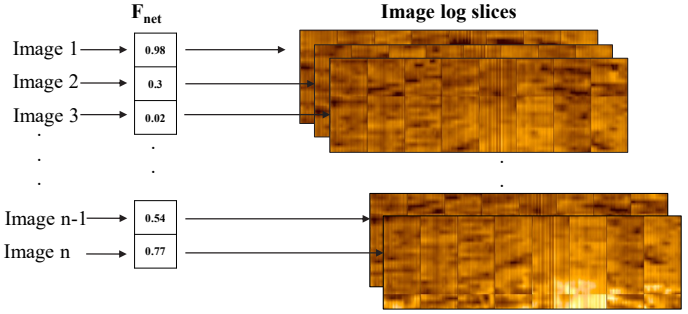
3.5.3 Net Pay Fraction Label Generation

The result from core UV images augmentation was an array with 27500 columns and multitude rows, in which each row represents one image. Each cell in the row had either a pixel value of 0 (oil bearing sand) or 255 (shale). Therefore by a simple averaging, the fraction of the cell with a value of 255 was determined. The fraction corresponds to the fraction of image pixel, which represents oil bearing sand, and therefore it corresponds to the net pay fraction of the image (Figure 3.16a). Since the cell values were either 0 or 255, net pay fraction (F_{net}) was computed by summing up the cell values in a single row (S_{cell}) and dividing it by the number of cells in the row (T_{cell}) multiplied by 255 as shown by Equation (3.1).

$$F_{net} = \frac{S_{cell}}{T_{cell} \times 255} \times 100\% \tag{3.1}$$



(a) Net pay fraction label generation.



(b) Label assignment to image log slices.

Figure 3.16: An illustration of (a) a net pay fraction label generation from core UV images and (b) labels assignment to image log slices.

The net pay fraction values of each row were utilized as the label for the machine learning training process. These values were saved in a separate 1D array in Python. Each image log slice was assigned with a net pay fraction label from the label array (Figure 3.16b). Each well was represented by a dataset that consisted of images from an image log and their corresponding labels.

3.6 Optimized Machine Learning Implementation

Machine learning implementation for net pay estimation was achieved by developing a convolutional neural network (CNN) algorithm.

3.6.1 Convolutional Neural Network Design

A series of configuration and hyperparameter tuning were performed to find a network design which gives an optimum result in terms of loss minimization, model fit, and estimation accuracy. The design of CNN is shown in Figure 3.17 and outlined in Table 3.13.

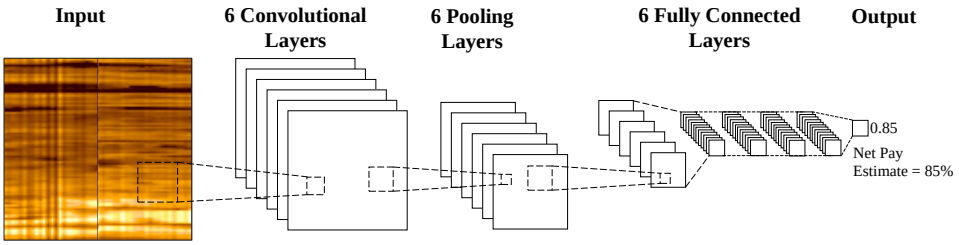


Figure 3.17: An illustration of convolutional neural network design in this study.

Table 3.13: Configuration and hyperparameter setting of the convolutional neural network in this study.

Configuration	Description
Convolutional layer	Number of layer: 6 Number of filter: 16,32,64,128, 256, 512 (from layer 1 to 6) Filter size: 3 x 3 Activation function: ReLu Padding: Enabled
Pooling layer	Number of layer: 6 Pooling window size: 2 x 2 Padding: Enabled
Fully connected layer	Number of layer: 6 Number of neuron: 512, 256, 128, 64, 32 and 1 Activation function: ReLu and linear
Loss	Type: Mean squared error (MSE)
Optimization algorithm	Type: Adam Learning rate: 0.0001 Batch size: 64
Other parameter	Early stopping: Stop training if validation loss was not minimized after 25 iterations.

The convolutional neural network used in this study consisted of the following elements which are discussed in the following subsections.

Dataset

The dataset for each well was split into training, validation, and test set. The training and validation sets were composed of samples from well 7220/7-3 S as it has a significant presence of both clean and shaly sand intervals in its reservoir. There were two different types of net pay estimation: estimation on samples from the same well as the training set; and estimation on samples from a different well. Therefore two different types of test set exist in this study.

The first estimation was performed to observe the capability of the method in estimating samples with the same logging environment as the samples in the training set. Even though the test set in that estimation was composed of samples from the same well as the training set, the test samples were unseen to the network during training. For the estimation on samples from a different well, the test sets were composed of samples from well 7220/5-1, 7220/7-1 and 7220/8-1. This was an attempt to simulate the practical application of the net pay estimation method, in which the CNN is trained on wells with core data and used to estimate net pay on wells without core data.

For the reliability evaluation, a multitude of additional test sets were generated from the existing test sets. Every sample in a test set was shifted horizontally between 24 to 168 pixels, thus generating seven different test sets from one test set (Figure 3.18). The horizontal shift by 24 pixels corresponds to a shift by one pad/flap, as a pad or flap is composed of 24 pixels. In the reliability evaluation, eight different test sets were utilized for estimation on each well. Those test sets contain different horizontal shift, and horizontal variation exists among the test sets. Since a borehole image log is essentially an unwrapped cylinder representing the borehole (c.f. Section 2.2.3), horizontal shifting did not alter the interpretation of the log. Thus, the net pay estimates across the different test sets should ideally be similar.

A statistical analysis was performed by generating a standard deviation band based on the estimates from the eight different test sets. Since the estimation across the eight different test sets should ideally be similar, the width of standard deviation band should not be wide. Through the generation and evaluation of the standard deviation band, the reliability of the net pay estimation method was evaluated.

Input

The input was a multitude of images from the image log, which were passed into the first convolutional layer. The pixel values of each image were also normalized from a greyscale value between 0 to 255, to a scale between 0 and 1. Normalization was aimed to transform features to be on a similar scale, to improve the performance and training stability of the model.

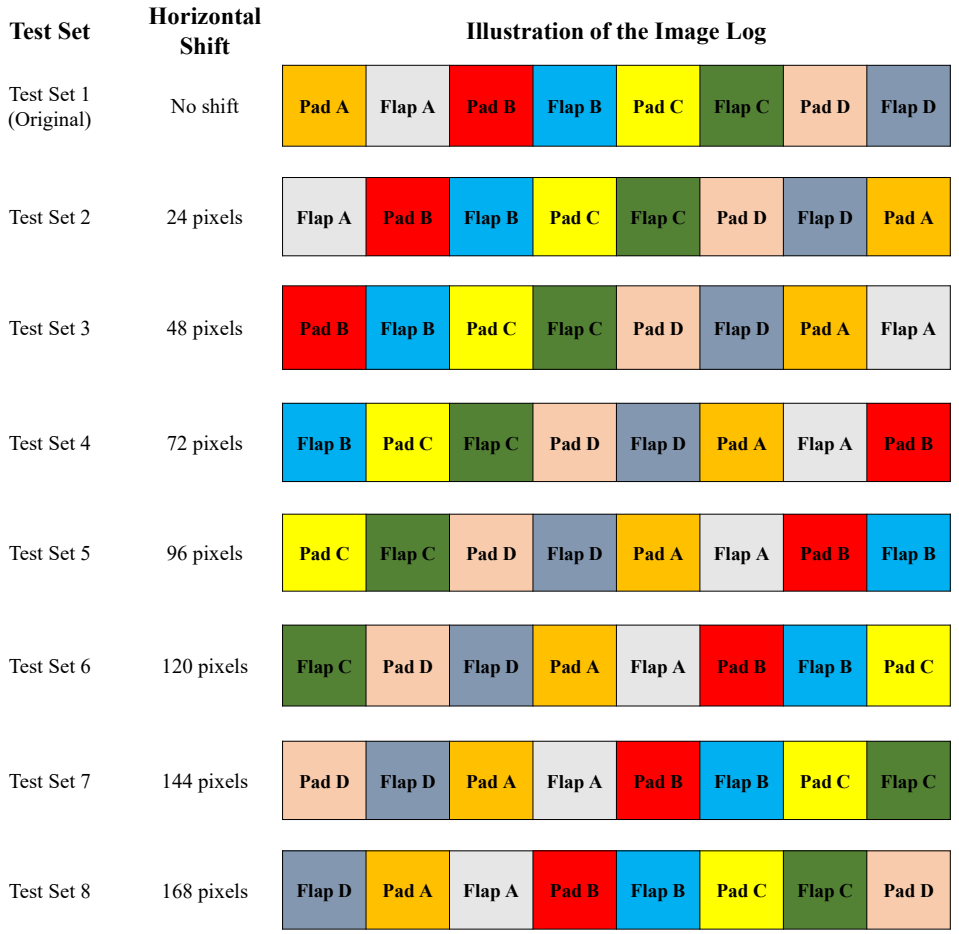


Figure 3.18: An illustration on how the test sets with horizontal variation were generated.

Convolutional and Pooling Layer

Convolutional and pooling layers performed a convolutional and pooling operation on the input to learn important features within the images. Each convolutional layer was assigned with the following hyperparameters: a filter with 3x3 size, a stride of 1, padding operation enabled and ReLU activation function. As an exception, the first convolutional layer was assigned with a stride of 2, to reduce the computational time. The filters were initialized with random values, and those values were updated during the training process. Adding a larger stride for the first layer resulted in a faster training process without a degradation in the estimation quality. Each pooling layer consisted of the following hyperparameters: a pooling window with 2x2 size and padding operation enabled. The number of layers, filters, windows, and hyperparameter settings were tuned to ensure all essential features were learned, and minimization of loss during training is achieved.

Fully Connected Layer

The fully connected layer performed computations on the flattened pooled feature map. The first layer was assigned with 512 neurons and ReLU activation function. The subsequent layers were assigned with a decreasing number of neurons and also a ReLU activation function. The final layer consisted of a single neuron which output net pay estimation for each image. A linear activation function was assigned in the final layer in order to output a continuous value. Similar to convolutional and pooling layers, the number of layers and neurons were tuned to ensure all essential features were learned, and minimization of loss during training is achieved.

Training Process and Output

During training, the CNN model was evaluated with the validation samples, and the validation loss was calculated as a mean squared error (MSE) between estimates on the validation samples and the corresponding net pay fraction labels (c.f. Section 2.4.2). Through a set of iterations during training, validation loss was minimized, and the model was updated using an Adam optimizer. Adam is an optimization algorithm based on mini-batch gradient descent method (c.f. Section 2.4.4). A learning rate of 0.0001 and a batch size of 64 were used to ensure that the loss was minimized properly without a long computation process.

An early stopping mechanism was implemented in the training process to prevent overfitting problem. With early stopping, the training process was terminated based on the monitoring of validation loss. If validation loss was not minimized within 25 iterations, the training was stopped. The model with the minimum loss was used for estimation on the test set. A test loss based on the test set was also computed as an MSE between estimates on the validation samples and the corresponding net pay fraction labels. A loss plot consisted of training, and validation loss was also generated. The output of the trained CNN model was an array containing net pay estimates for each image in the test set, which was compared with test set labels to assess the model estimation capability.

3.7 Statistical Analysis of the Estimation Results

3.7.1 Statistical Analysis for Accuracy Evaluation

In the accuracy analysis of estimation results, an estimation error was calculated for each net pay estimate. Estimation error (e) is an error between a net pay estimate (y) and its net pay fraction label ($\hat{y}$), as seen in Equation (3.2).

$$e = y - \hat{y} \tag{3.2}$$

Based on the estimation error calculation, a box plot was generated to plot the distribution of estimation errors in each estimation (Figure 3.19). Box plot is a type of chart commonly used to show the distribution of numerical data through displaying the data quartiles, median and mean (Lane, 2013). A box plot gives an indication of the spread or range of the

data based on the width of the box. It also shows the presence of outliers within a data set.

A box plot is also used to display the skewness of data. Skewness is a measure of the lack of symmetry in the data distribution. In data with positive skew, the mean is greater than the median. In data with negative skew, the mean is smaller than the median. Skewness gives an indication of where the majority of the data lies, either on the lower or higher side of the distribution. The measures of data shown in a box plot are explained as follows:

- Minimum: The lowest value of a data, excluding outliers.
- Lower quartile: Twenty-five percent of the data fall below the lower quartile value
- Median: The median marks the mid-point of the data and is shown by the line that divides the box into two parts. Half of the data are greater than or equal to this value, and half are less.
- Upper quartile: Seventy-five percent of the data falls below the upper quartile value. Therefore, 25% of the data are above this value.
- Maximum: The highest value of the data, excluding outliers.
- Whiskers: The upper and lower whiskers represent data which are located outside the middle 50% of the data.
- Inter quartile range: This box shows the middle 50% of the data.

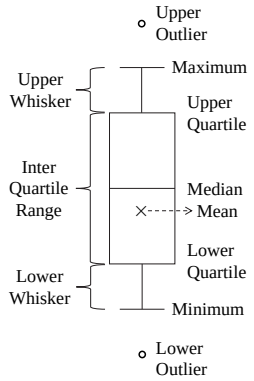


Figure 3.19: Measures of a data shown in a box plot.

3.7.2 Statistical Analysis for Reliability Evaluation

The capability of the net pay estimation method in generating estimations which were independent of horizontal orientation was evaluated through a set of net pay estimations on eight test sets with varying horizontal orientation (c.f. Section 3.6.1). The variability within the eight estimates from the test sets were observed by calculating the standard

deviation of the estimates. Variability is a statistical terminology which refers to the extent in which values differ from one another (Lane, 2013). Standard deviation (σ) is a measure of variability and it is calculated using the following formula:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2} \tag{3.3}$$

In Equation (3.3), the number of observation is represented by N; individual value is represented by x_i; and the mean of the population is represented by μ.

Since the net pay estimates were a continuous value, a standard deviation band was generated to give a better visualization of the variability (Figure 3.20). The standard deviation band (B) is basically an interval defined by the following equation:

$$B = \Big[\mu - \sigma\,,\ \mu + \sigma\Big] \tag{3.4}$$

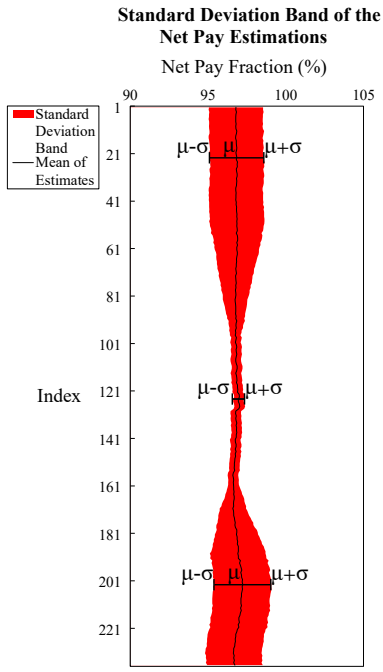


Figure 3.20: An example of a standard deviation band showing it as an interval with the population mean at its center. The wider portion of the band indicates a higher variability within the population.

The width of a standard deviation band gives a hint on the variability within the population. A population with large variability within its values has a wider band compared to a population with small variability (Srivastava, 2008).

Chapter 4

Results and Discussions

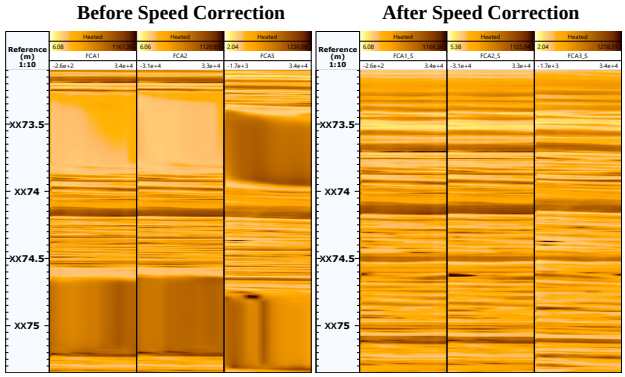
4.1 Results of the Optimized Data Processing

In this study, optimizing the data processing procedures is one of the objectives in the optimization of the machine learning-based net pay estimation method. The image logs and core UV photographs processing procedures were aimed to generate the necessary datasets which serve as input and labels for the machine learning technique. The following two sections outline the results of the optimized image logs and core UV photographs processing.

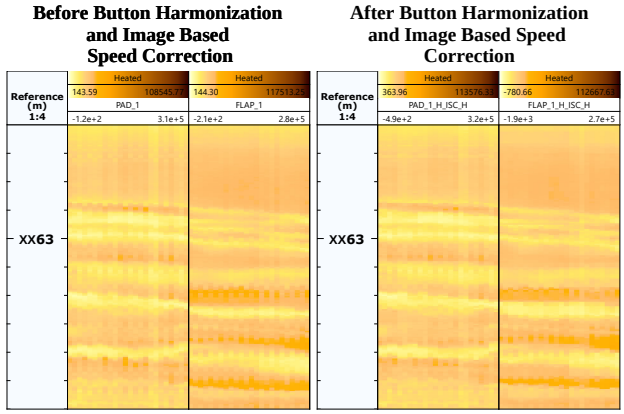
4.1.1 Image Logs Processing Results

Image logs processing began with a general quality control of the logs and also an inclinometry quality control (QC) in Techlog. Based on the general quality control, the DLIS files obtained from Diskos contained image logs with all of the required channels and without any missing interval. Having a complete interval is essential as the image logs should ideally cover the intervals present in the corresponding core UV photographs. Visual observation of image logs did not find any interval with exceptionally poor borehole condition or excessive noise in the reservoir zone. In all of the wells, the accelerometer readings formed an ideal plot as the values of the x-axis and z-axis acceleration were close to zero. The magnetometer readings also showed a circular shape in all of the wells. This implies that based on the inclinometry QC, there is no correction needed for the inclinometer measurements.

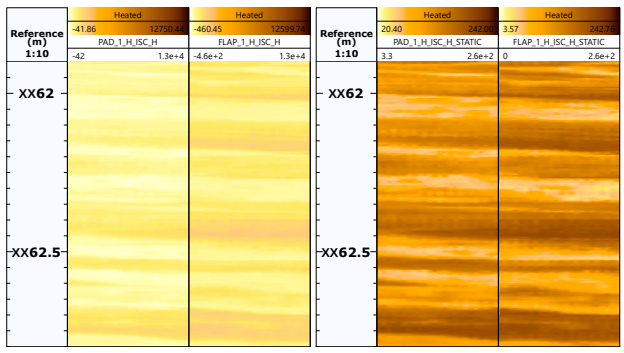
The optimized image logs processing procedures managed to rectify image artefacts present in the raw image logs. The processing corrected several artefacts such as pads misalignment, sawtooth effect, and pads smearing effect. Although some of the image artefacts remained present in some intervals, the quality was still sufficient for net pay fraction estimation purposes. Several examples of image artefacts correction and other processing results are shown in Figure 4.1.



(a) Pad smearing effect corrected through speed correction in an interval from well 7220/7-3 S.



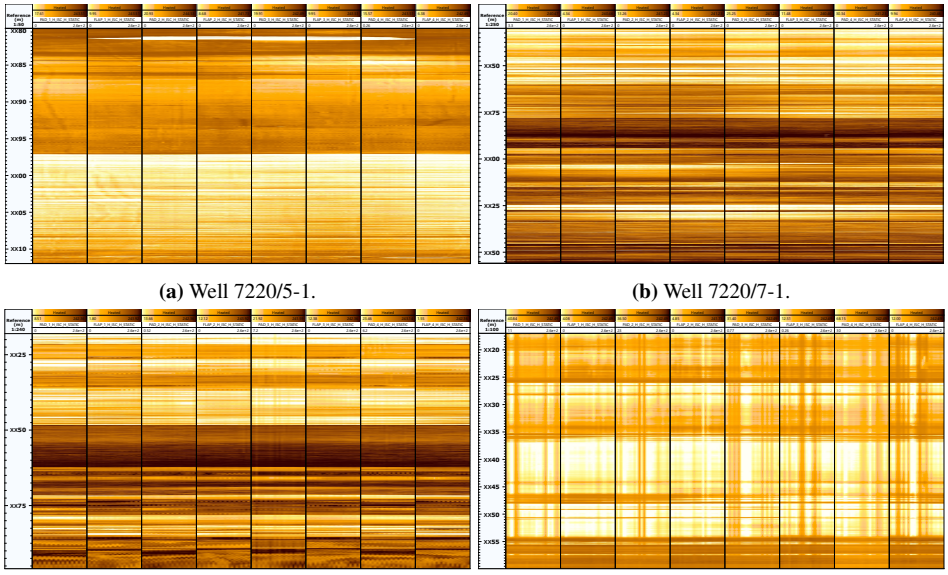
(b) Reduction of sawtooth effect with button harmonization and image based speed correction from well 7220/8-1. The pixel value in the color scale were altered due to the button harmonization.



(c) Contrast enhancement with static normalization in an interval from well 7220/7-1.

Figure 4.1: Several examples of image artefact corrections and contrast enhancement performed during image logs processing.

As shown by Figure 4.2, image logs with static normalization in a standardized display were generated by the image logs processing. Through the depth correlation procedure, it was observed that depth shifts were required to remove the offsets between the image logs and core UV photographs (Table 4.1). The final result of image logs processing was four image datasets from four exploration wells in Johan Castberg.



(a) Well 7220/5-1. **(b)** Well 7220/7-1.

(c) Well 7220/7-3 S. **(d)** Well 7220/8-1.

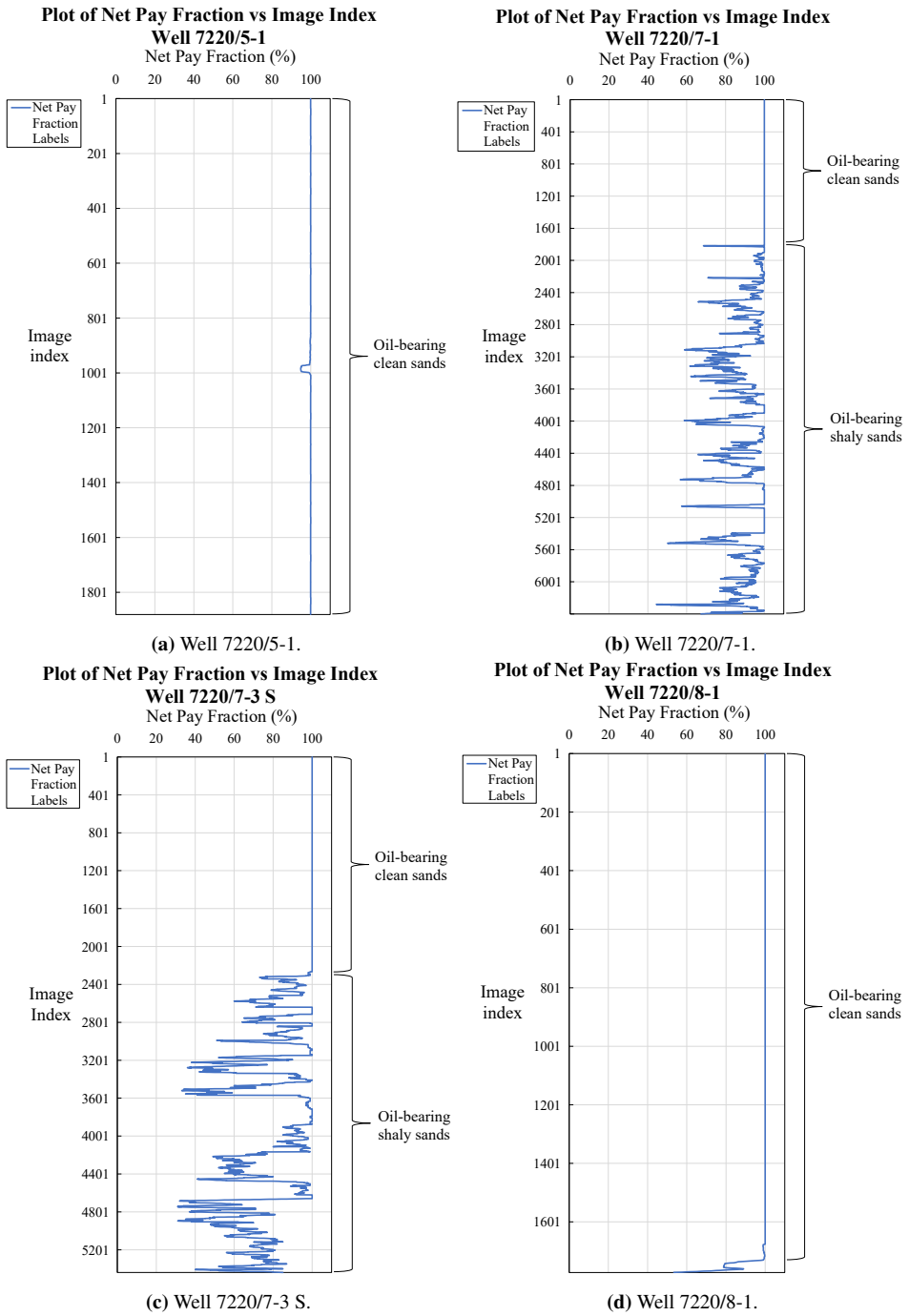
Figure 4.2: Fully processed image logs from each exploration well in this study. Each image stripe within the image log represents a measurement from one pad or flap.

Table 4.1: Details of depth shift values and number of samples in the image dataset for each well.

Well	Depth shift	Number of Samples
7220/5-1	0.2 m down	1881
7220/7-1	1.05 m down	6400
7220/7-3 S	0.9 m down	5439
7220/8-1	1.5 m down	1773

4.1.2 Core UV Photographs Processing Results

Core UV photographs processing generated label datasets comprised of net pay fraction values. The number of labels in each label dataset corresponds to the number of images in the image dataset. Since the image dataset was composed of overlapping images, the labels were displayed in a plot against image indices instead of depth, which represents the number of labels or images in the dataset (Figure 4.3).



(a) Well 7220/5-1.

(b) Well 7220/7-1.

(c) Well 7220/7-3 S.

(d) Well 7220/8-1.

Figure 4.3: Plot of net pay fraction labels of each exploration well in the study.

As seen in Figure 4.3, well 7220/5-1 and well 7220/8-1 are composed of mostly clean sand samples, while well 7220/7-1 and 7220/7-3 S are comprised of both clean and shaly sand samples. Samples with a clean sand interval are indicated by the smooth curve with net pay fraction values between 95-100%, while samples with a shaly sand interval are indicated the fluctuating curve with net pay fraction values between 30-95%.

4.2 Net Pay Estimation Based on Data From The Same Well

A series of net pay estimations were performed using the processed data, to evaluate the capability of the optimized net pay estimation method. The optimized convolutional neural network was trained based on data from one of the wells in the Johan Castberg field. The optimized network generated a model that was used to estimate the net pay of unseen image log samples. These unseen image log samples were acquired from either the same well as the training and validation samples or from a different well (c.f. Section 3.6.1). The accuracy and reliability of the developed net pay estimation method were evaluated through a set of statistical analyses. The accuracy was evaluated by observing the loss plot and analyzing the error on some specific lithologies, e.g. clean sand and shaly sand. The reliability of the estimation was evaluated by performing estimation on test sets with different horizontal orientation, and conducting a statistical analysis on the results (c.f. Section 3.7).

For the first net pay estimation, the training, validation, and test samples were acquired from the same well. The purpose of performing net pay estimation based on data from the same well is to observe the capability of the optimized method in estimating samples from the same logging environment. The dataset from well 7220/7-3 S was selected due to the significant presence of both clean and shaly sand intervals in its reservoir. That was done to ensure that both types of lithology were involved in the training and estimation process.

The dataset split was conducted according to this rule: 64% of the samples for the training set, 18% of the samples for the validation set, and 18% of the samples for the test set. The dataset curation was performed carefully to ensure no overlap between the training, validation, and test set. This implies that the test set remained unseen to the network during training, and thus keeping the integrity of the training process. Each set also contained both clean and shaly sand samples to ensure that the network was trained and generated estimation on both types of lithology. This net pay estimation was conducted with data listed in Table 4.2.

Table 4.2: Details of the datasets used in the net pay estimation based on data from the same well. The training, validation and test samples were taken from well 7220/7-3 S.

Dataset	Number of Samples	Source
Training set	3486	7220/7-3 S
Validation set	971	7220/7-3 S
Test set	982	7220/7-3 S

During the training process, the training and validation loss were calculated as a Mean Squared Error (c.f. Section 2.4.2 and 3.6.1) and recorded. Once the minimum validation loss had been reached, a net pay estimation was conducted on the test set. The test loss was also calculated using MSE. The model generated by the trained network in this estimation is referred to as Model A. Table 4.3 outlines the details of the training process, and also the test loss of estimating samples from well 7220/7-3 S. The training and validation loss of Model A were plotted as shown in Figure 4.4.

Table 4.3: Details of the training process of Model A and the test loss achieved in estimating net pay based on data from well 7220/7-3 S.

Model	Details	Value
Model A	Iteration	52
	Training loss	0.0149
	Validation loss	0.0081
	Test loss	0.0127

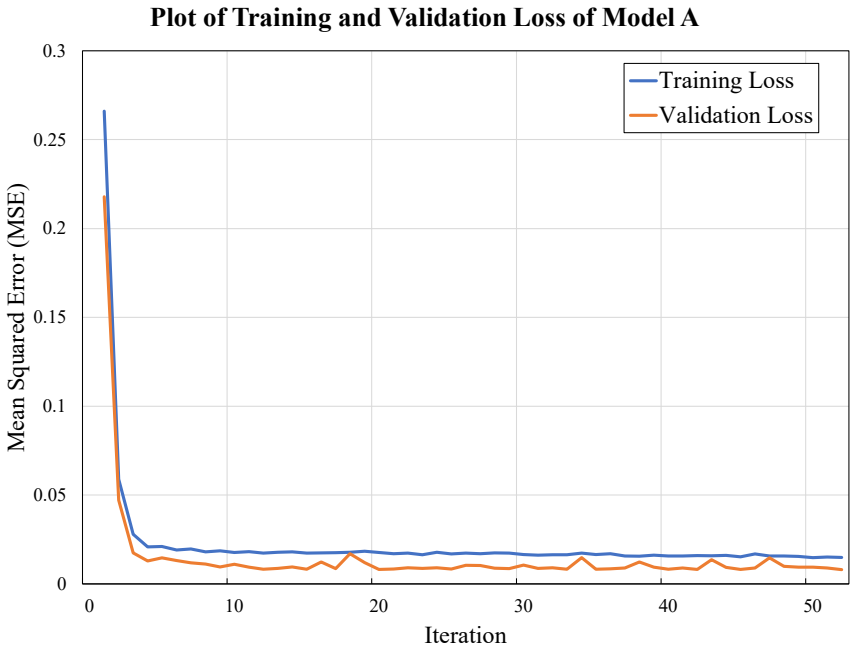


Figure 4.4: Plot of Training and Validation Loss of Model A.

The net pay estimates were plotted together with the test set labels to observe the estimation quality. The result of net pay estimation based on data from well 7220/7-3 S is displayed in Figure 4.5.

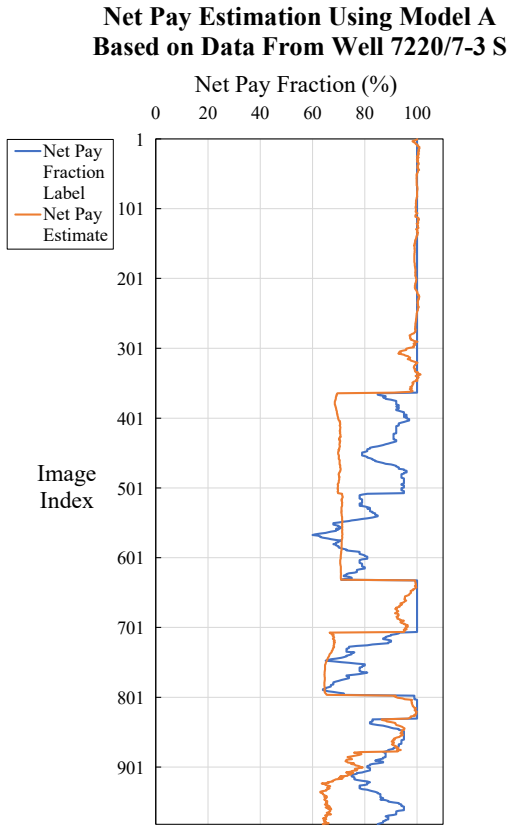


Figure 4.5: Net pay estimation based on data from well 7220/7-3 S.

Result Overview

The results showed that the optimized net pay estimation method was able to perform net pay estimation based on data from the same well, and hence the same logging environment. The test set consisted of both clean and shaly sand samples, and the network generated an estimation with a test loss of 0.0127. The network was able to generate estimates which followed the general trend of the net pay fraction labels. Through visual observation, Figure 4.5 shows that the estimates on clean sand samples were relatively closer to the label values compared to the estimates on shaly sand samples.

Generally, the network tended to generate estimates which underestimated the net pay fraction in the shaly sand intervals. The network managed to estimate the decline of net pay fraction in a shaly sand interval. However, it did not generate estimates with fluctuation of net pay fraction. This was possibly caused by a limitation in the logging tool resolution. The resolution of the borehole image logging tool might not be sufficient in identifying thin layers of sand or shale, which were identifiable in the core UV photographs (Figure 4.7).

Analysis of Loss Plot

Based on the training process, a loss plot which consists of training and validation loss was generated. The loss plot provides an insight of the model fit. It shows that the training and validation loss were gradually decreasing through the iterations. Figure 4.4 and Table 4.3 show that after 52 iterations, the training loss was minimized to 0.0149, and the validation loss was minimized to 0.0081. Since both of the losses were minimized and converged to a certain minimum, it implies that there was no model fit problem in this estimation. The early stopping mechanism prevented overfitting by stopping the training after 25 iterations without further minimization of validation loss (c.f. Section 2.4.4). The absence of model fit problem indicates that the model was able to generalize to unseen data.

The loss plot also shows that the validation loss was slightly lower than the training loss, which indicates that the model was able to generate a more accurate estimation on the validation set compared to the training set. This was possibly caused by the intense image augmentation mechanism, which was implemented only on the training samples, but not on the validation samples (c.f. Section 3.4.9). During training, image augmentation induced more orientation variation in the training set. Therefore the training set would contain samples that were more difficult to estimate compared to the validation set.

Evaluation of the Optimization Approaches

Compared to the results of the preceding study (c.f. Section 2.5.4), the optimized net pay estimation method produced relatively more accurate results, when estimating net pay based on data from the same well. In this study, the optimized network generated a model with a significantly lower training and validation loss compared to the losses of Model X from the preceding study (Table 4.4). Despite having more iterations in the training process, the optimized network in this study generated an estimation with a lower test loss than the network in the preceding study. Through visual observation, it can also be observed that the estimates in this study followed the trend of the net pay fraction labels better than the estimates in the preceding study. This implies that the optimization approaches introduced in this study managed to generate a network with a better loss minimization and thus producing a more accurate estimation.

Table 4.4: A comparison between the result of the preceding study and the result of this study in net pay estimation based on data from well 7220/7-3 S.

Result of Net Pay Estimation Based on Data From Well 7220/7-3 S					
Preceding Study			**This Study**		
Model	**Details**	**Value**	**Model**	**Details**	**Value**
Model X	Iteration	14	Model A	Iteration	52
	Training loss	0.0459		Training loss	0.0149
	Validation loss	0.0403		Validation loss	0.0081
	Test loss	0.0417		Test loss	0.0127

Accuracy Analysis

The estimation error for each sample was computed, and the distribution of the error was plotted in a box plot (c.f. Section 3.7.1). Figure 4.6 shows the distribution of the estimation error for each sample in the test set. Based on the net pay fraction label, the samples were grouped into a certain lithology, i.e. clean sand and shaly sand. Samples with a net pay fraction label above 95% were classified as clean sand samples. On the other hand, samples with a net pay fraction label between 30% to 95% were classified as shaly sand samples.

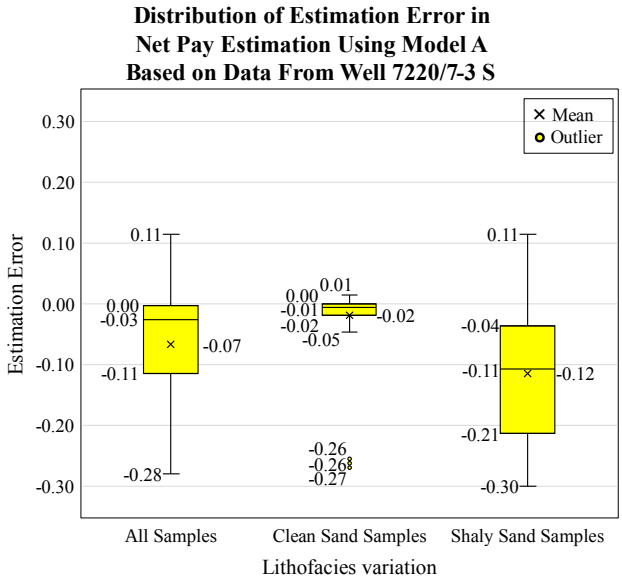


Figure 4.6: Net pay estimation based on data from well 7220/7-3 S.

The box plot shows that the box of clean sand samples was significantly narrower compared to the box of shaly sand samples. The wider box of shaly sand samples indicates that the estimates in the shaly sand samples had a wider range of errors. In the clean sand samples, the median of errors was closer to zero (-0.01) than the shaly sand samples (-0.12). The errors in shaly sand samples had a significantly larger maximum and minimum values at the end of the whiskers. This implies that the net pay estimation method was generally more accurate in estimating clean sand samples than shaly sand samples. The overall median of errors from the all of samples was close to zero (-0.03), which means the estimation was generally accurate throughout the samples. The outliers in the clean sand box possibly represent samples with a calcite cemented sand interval.

The poor performance of the net pay estimation method in shaly sand samples highlights the limitation of borehole image logging tool resolution in differentiating thin layers of sand or shale. Figure 4.7 shows that the thin layers of sand and shale in a shaly sand interval is more visible in the core UV photographs compared to the corresponding image log. A borehole image logging tool can identify a sand or shale layer as thin as 0.2 inch

or 5.08 cm (c.f. Section 2.2.1), while a core UV photograph can clearly visualize a thinner layer. Therefore, the net pay fraction labels fluctuated in the shaly sand interval due to more identifiable thin layers of sand or shale. The image log samples in the shaly sand interval did not have much variation in terms of color contrast and pixel value as it should be due to less identifiable thin layers of sand or shale. However, they still had a contrasting difference compared to the clean sand samples; therefore, the network still managed to estimate the decline of net pay fraction in a shaly sand interval.

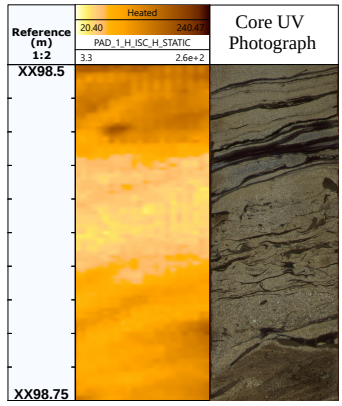


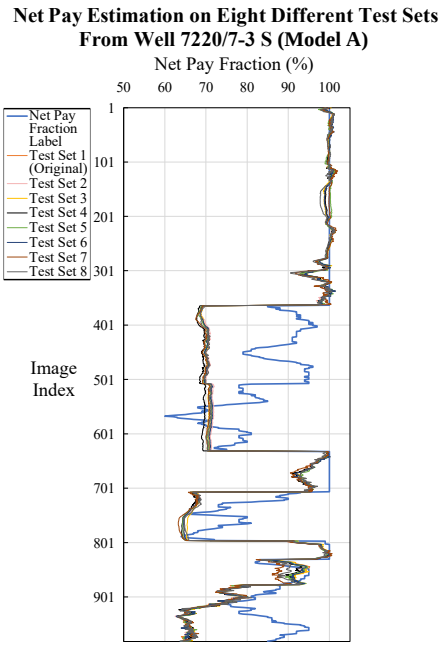
Figure 4.7: A comparison of a 25 cm interval of borehole image log and core UV photograph from well 7220/7-1. The core UV photograph possesses a resolution advantage as it managed to visualize a layer of sand or shale thinner than 5.08 cm.

In the box plot, a positive error value represents an overestimation of a sample's net pay fraction. On the other hand, a negative error value represents an underestimation of the net pay fraction value of a sample. Figure 4.6 shows that the box of *All Samples* is located below zero. This implies that generally, the net pay estimation method underestimated the net pay fraction of a sample. The *All Samples* box also shows a negative skew, indicated by a mean lower than the median. A wider lower skewer compared to the upper skewer, also indicates a negative skew in the errors. This indicates that most of the errors are negative, further confirming the tendency of net pay fraction value underestimation by the network.

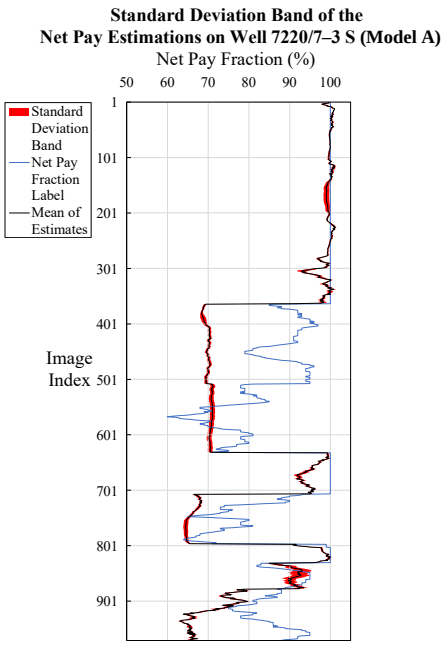
Reliability Evaluation

Besides analyzing the accuracy of the estimation, the estimation's reliability was evaluated to ensure that the estimation was independent of horizontal variation in the image logs. As previously discussed in Section 3.6.1, the reliability of the net pay estimation method was evaluated by performing net pay estimation on eight test sets with varying horizontal orientation. The variability within the eight estimates was observed by calculating the standard deviation of the estimates in each image index. Since the net pay estimates were a continuous value, a standard deviation band was generated to give a better visualization of the variability (c.f. Section 3.7.2).

Seven additional test sets were generated from the original test set of well 7220/7-3 S. Those test sets differed only in a horizontal orientation as each test set was shifted horizontally by an increment of 24 pixels, corresponding to a shift by one pad or flap. An interval of image log is essentially an unwrapped cylinder representing the borehole (c.f. Section 2.2.3). Therefore horizontal shifting ideally does not alter the value of net pay of the interval. The number of samples in every additional test set is the same as the number of samples in the original test set. Model A was utilized to generate estimation on the eight different test sets. The estimates from the eight test sets were shown in Figure 4.8a, and the calculated standard deviation was plotted in Figure 4.8b.



(a) Net pay estimation on eight different test sets from well 7220/7-3 S.



(b) Standard deviation band on the net pay estimation on eight different test sets from well 7220/7-3 S.

Figure 4.8: Net pay estimation on eight different test sets and the standard deviation band from well 7220/7-3 S.

The plot of net pay estimation (Figure 4.8a) shows that the estimates from each test set were visually similar in terms of trend and value. The test loss of each estimation is outlined in Table 4.5, and the value of each test loss was close to each other, with a test loss average of 0.0128. The plot of the standard deviation band also shows that the band was generally narrow for most of the samples. An interval with a narrow standard deviation band indicates that the variability within the eight different estimates on that interval was low. This implies that the net pay estimation method was generally able to generate an estimation independent of horizontal orientation in the image logs.

Table 4.5: Test losses from net pay estimation on eight test sets of well 7220/7-3 S.

Model	Test Set	Test Loss
Model A	Test Set 1 (Original)	0.0127
	Test Set 2	0.0125
	Test Set 3	0.0125
	Test Set 4	0.0135
	Test Set 5	0.0128
	Test Set 6	0.0126
	Test Set 7	0.0133
	Test Set 8	0.0125
Average		0.0128

Although the network was not affected by horizontal orientation within the image log samples, there were still several shaly sand intervals with varying net pay estimates. Figure 4.8b shows that the shaly sand intervals between 510 to 620 and 820 to 860 were having slightly wider standard deviation bands than the rest of the samples. This was possibly due to the limitation of the network performance in estimating shaly sand samples. However, the variability on those intervals was still not high enough to render a wide standard deviation band. Thus, it indicates that the estimation method was generally not affected by horizontal orientation in the image logs.

4.3 Net Pay Estimation on Data From a Different Well

This net pay estimation was an attempt to simulate the practical application of the machine learning-based net pay estimation method. The network was trained with samples from one well, and the generated model was utilized to estimate unseen samples from a different well. This implies that the image log samples in the test set might be acquired from a well with a different logging environment compared to the training and validation samples (Table 4.11). Similar to the previous net pay estimation, the accuracy and reliability of the net pay estimation method was evaluated through a set of statistical analysis.

In this estimation, the training and validation samples were taken from well 7220/7-3 S, and the test samples were taken from either well 7220/7-1, 7220/5-1, or 7220/8-1. Well 7220/7-1 is composed of both clean and shaly sand intervals, while well 7220/5-1 and 7220/8-1 are composed of mostly clean sand (c.f. Section 4.1.2). Each well was represented by a test set; therefore, three different test sets were utilized in this estimation.

Two different models were utilized to perform net pay estimation on three test sets. The two models differed in the number of training samples used during the training process. The first model was trained with 3486 samples. It was the same model which was used in the previous net pay estimation, Model A. By utilizing Model A, the performance of the model in estimating samples from a different well was compared with the performance in estimating samples from the same well which was done in the previous section (c.f. Section 4.2). The second model, Model B, was trained with 4468 samples. The training

set of Model B was constructed by incorporating samples from the test set of well 7220/7-3 S into the training set. This implies that all the available samples of well 7220/7-3 were utilized in the training process of Model B. Table 4.6 outlines the datasets used in this net pay estimation for both models.

Table 4.6: Details of the datasets used in the net pay estimation on data from a different well. The training and validation samples were taken from well 7220/7-3 S, while the test samples were taken from well 7220/7-1, 7220/5-1 and 7220/8-1.

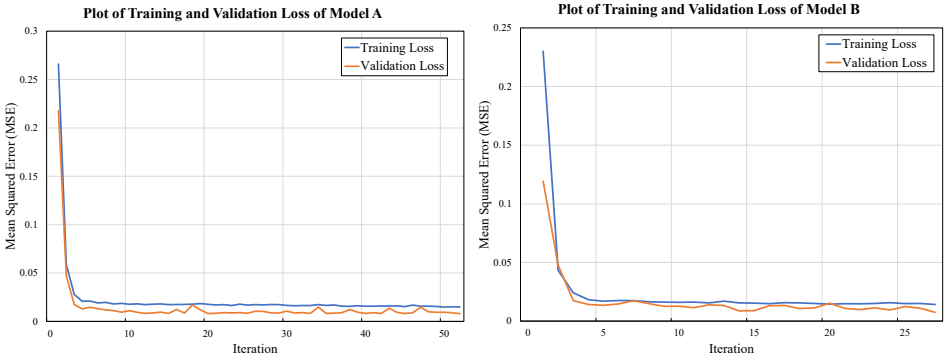
Model	Dataset	Number of Samples	Source
Model A	Training set	3486	7220/7-3 S
	Validation set	971	7220/7-3 S
	Test loss (7220/7-1)	6400	7220/7-1
	Test loss (7220/5-1)	1881	7220/5-1
	Test loss (7220/8-1)	1773	7220/8-1
Model B	Training set	4468	7220/7-3 S
	Validation set	971	7220/7-3 S
	Test loss (7220/7-1)	6400	7220/7-1
	Test loss (7220/5-1)	1881	7220/5-1
	Test loss (7220/8-1)	1773	7220/8-1

Each model generated three results from the three test sets. The details of the training process, including the number of iteration, training, validation, and test loss for each model is outlined in Table 4.7.

Table 4.7: Details of the training process of the two models and the test losses for the three test sets.

Model	Details	Value
Model A	Iteration	52
	Training loss	0.0149
	Validation loss	0.0081
	Test loss (7220/7-1)	0.0123
	Test loss (7220/5-1)	0.0009
	Test loss (7220/8-1)	0.0107
Model B	Iteration	27
	Training loss	0.0141
	Validation loss	0.0075
	Test loss (7220/7-1)	0.0090
	Test loss (7220/5-1)	0.0006
	Test loss (7220/8-1)	0.0036

The training and validation loss of each model were plotted and shown in Figure 4.9. The net pay estimates of each well were plotted together with the corresponding test set labels to observe the quality of the estimation. The results of net pay estimation from each well are shown in Figure 4.10 to 4.12.



(a) Plot of Training and Validation Loss of Model A. **(b)** Plot of Training and Validation Loss of Model B.

Figure 4.9: Plot of Training and Validation Loss of Model A and B.

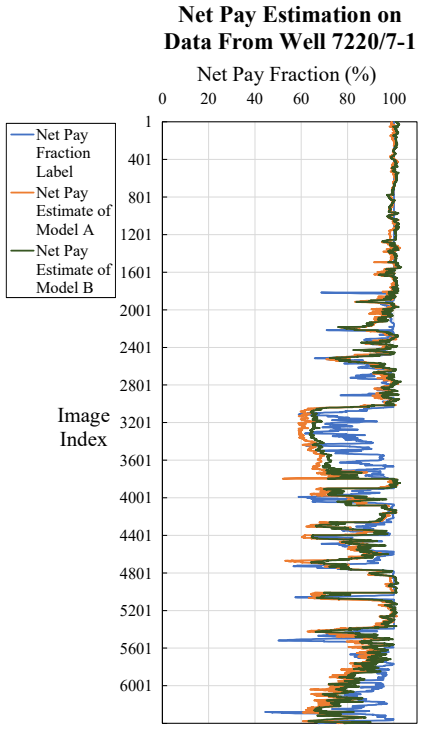


Figure 4.10: Net pay estimation using Model A and B on data from well 7220/7-1.

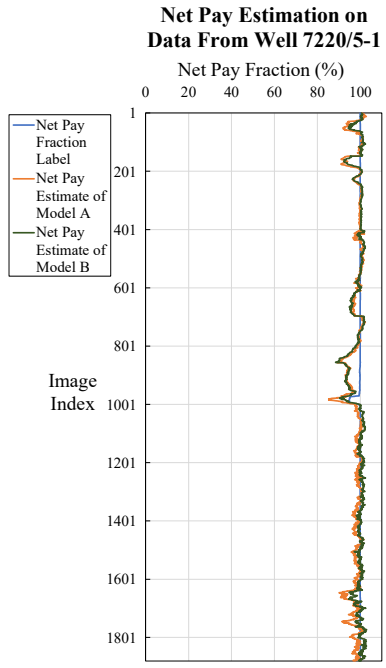


Figure 4.11: Net pay estimation using Model A and B on data from well 7220/5-1.

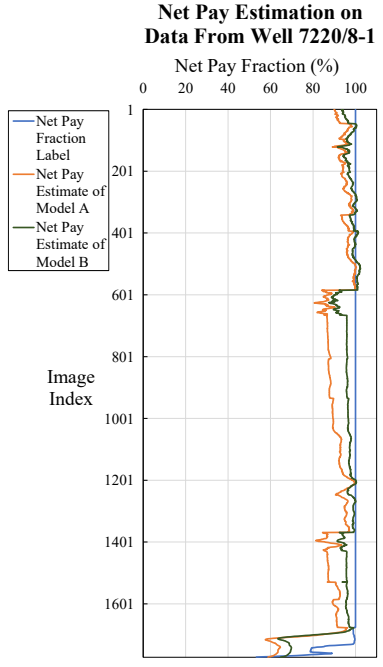


Figure 4.12: Net pay estimation using Model A and B on data from well 7220/8-1.

Result Overview

The result shows that generally, the optimized net pay estimation method was able to perform net pay estimation on data from a different well, which might have a different logging environment with the training data. Through a visual observation of the net pay estimates plots across the three different wells, the results showed that the estimates were generally following the trend of net pay fraction labels.

The test set of well 7220/7-1 comprises both clean and shaly sand samples, similar to the test set of well 7220/7-3 S in the previous net pay estimation. However, net pay estimation on well 7220/7-1 (Figure 4.10) shows that the estimates on shaly sand samples were better compared to the shaly sand sample estimates in the previous net pay estimation (c.f. Section 4.2). One of the possible cause is the shaly sand samples in the test set of well 7220/7-1 is more similar to the shaly sand samples in the training set, in terms of formation properties such as resistivity or lithology. This implies that for a model used on different test sets with similar composition, the performance might vary. This limitation conforms with the axiom that no model is universally suitable for every problem (Gómez and Rojas, 2015). The introduction of additional training samples with a larger degree of variation might generate a model which is able to work better for a wider variety of image logs.

The test set of well 7220/5-1 and 7220/8-1 is comprised of clean sand samples. The estimations on the test set of well 7220/5-1 (Figure 4.11) were quite accurate, with a relatively low test loss compared to the test losses of the other estimations. The estimation on the test set of well 7220/8-1 (Figure 4.12) shows that the network generally underestimated the net pay of samples, especially on the result of Model A. For a test with mostly clean sand samples. The estimates of well 7220/8-1 were not as smooth as the estimates of well 7220/5-1. This was possibly caused by a difference in the logging environment between those two wells. The borehole fluid salinity in well 7220/5-1 was 35508 ppm, while the borehole fluid salinity in well 7220/8-1 was 57921 ppm (Table 4.11). The significant difference of borehole fluid salinity might influence the response of the image logging tool, as resistivity-based measurements are generally sensitive to borehole fluid salinity. Due to the difference in salinity, the image logging tool might measure a different resistivity value for rocks with a similar lithology. The difference in measured resistivity values was reflected by the range of pixel values. This difference in pixel values range was interpreted differently by the network, although the image logs might visually look similar to the human eye.

In general, the performance of Model B was better compared to Model A based on the results across the three wells. Model B managed to produce better net pay estimations in terms of loss and estimates trend. Table 4.7 shows that the validation loss of Model B was lower than the validation loss of Model A. A lower validation loss indicates that Model B achieved a better generalization to unseen data compared to Model A, which was also reflected by the lower test losses of Model B in all the results. Besides having smaller losses, Model B also generated net pay estimates which were closer to the net pay fraction labels and thus having a more similar trend with the labels. Model B was trained with more training samples; therefore it implies that the introduction of more training samples

improved the loss minimization during training which led to a more accurate estimation.

Analysis of Loss Plot

Figure 4.9 shows that the losses were minimized and converged to a certain minimum during the training process of both models. This implies that there was no model fit problem in this estimation. With the implementation of early stopping mechanism, overfitting was avoided by terminating the training on iteration 52 for Model A and iteration 27 for Model B. It can also be observed that the training process of Model B was shorter indicated by a smaller number of iterations. This implies the training process of Model B achieved an earlier minimization of the validation loss. The loss plot of Model B shows a slightly lower validation loss than the training loss, similar to model A. As previously discussed in Section 4.2, a lower validation loss compared to training loss was possibly caused by the intense image augmentation mechanism which was implemented only on the training samples.

Evaluation of the Optimization Approaches

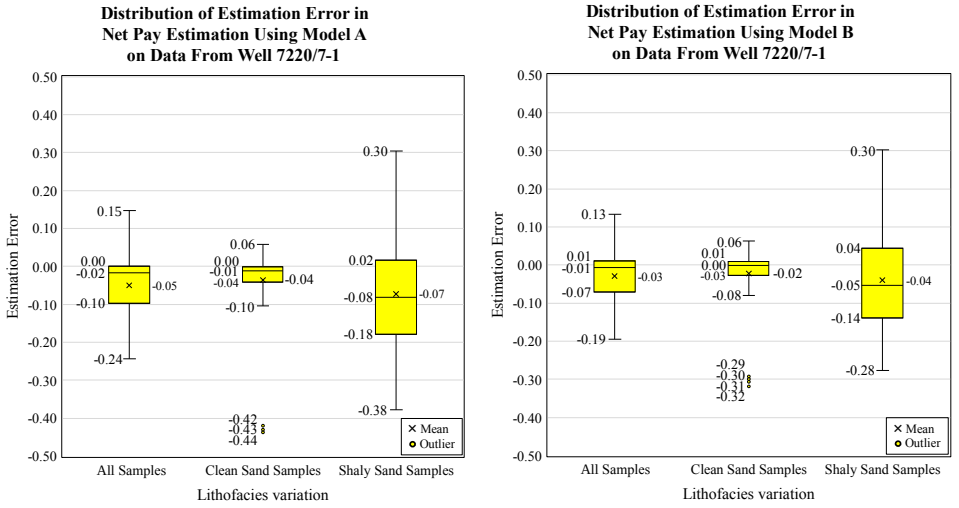
Compared to the results of the preceding study (c.f. Section 2.5.4), the optimized net pay estimation method generated a relatively more accurate result, when estimating net pay on data from a different well. In this study, the optimized network generated a model with a significantly lower training and validation loss compared to the losses of Model Y from the preceding study (Table 4.8). Although the training process was longer in this study, the optimized network generated estimations with a lower test loss than the network in the preceding study. In this study, the estimates on data from a different well followed the net pay fraction labels' trend better than the estimates in the preceding study. This confirms the founding in the previous net pay estimation (c.f. Section 4.2), in which the optimization approaches introduced in this study generated a network with a better minimization of loss and thus producing a more accurate estimation.

Table 4.8: A comparison between result of the preceding study and result of this study in estimating net pay on data from a different well.

Result of Net Pay Estimation on Data From a Different Well				
Preceding Study		**This Study**		
Details	**Model Y**	**Details**	**Model A**	**Model B**
Iteration	8	Iteration	52	27
Training loss	0.0459	Training loss	0.0149	0.0141
Validation loss	0.0276	Validation loss	0.0081	0.0075
Test loss (7220/5-1)	0.0457	Test loss (7220/5-1)	0.0009	0.0006
Test loss (7220/8-1)	0.0740	Test loss (7220/8-1)	0.0107	0.0036

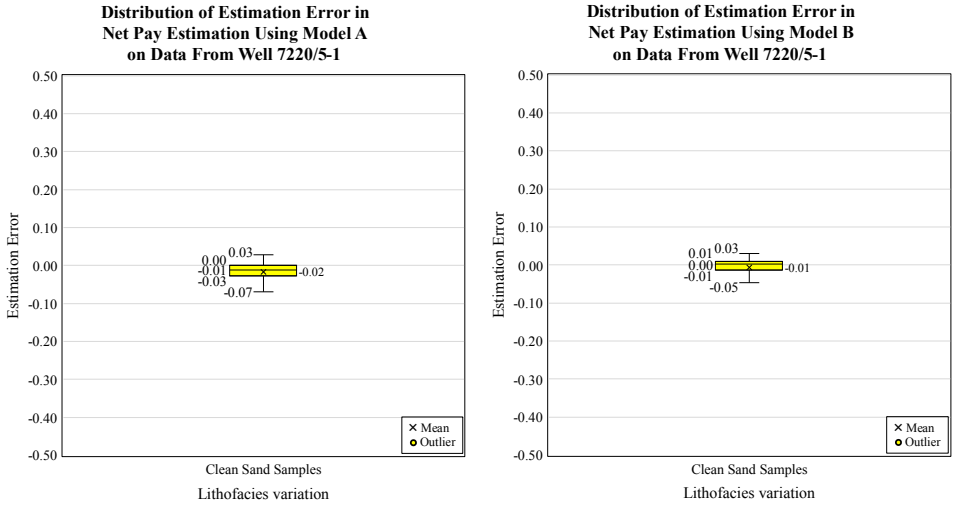
Accuracy Analysis

A set of box plots were generated to plot the distribution of estimation errors of the results across the three wells. Figure 4.12 to 4.14 show the distribution of the estimation errors for each test set from net pay estimation using Model A and B.



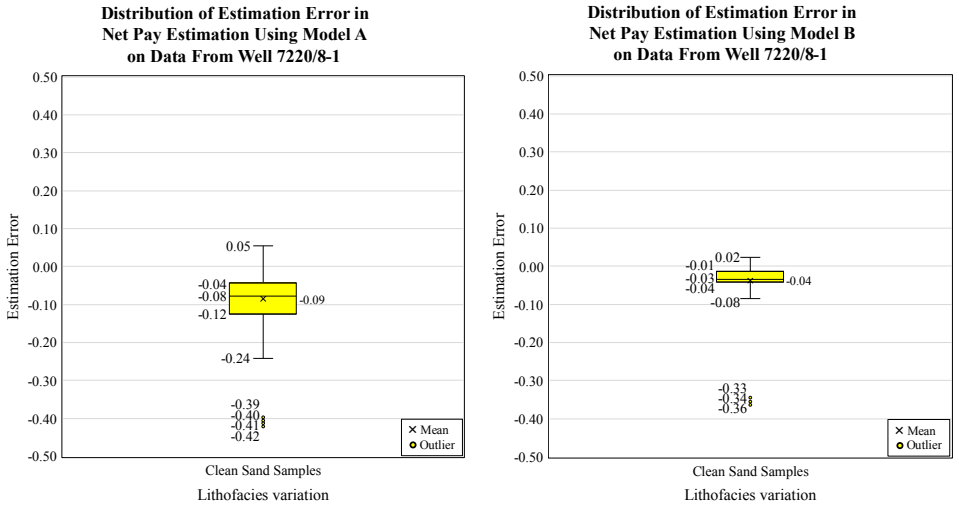
(a) Model A. **(b)** Model B.

Figure 4.13: Distribution of estimation error in net pay estimation on data from well 7220/7-1.



(a) Model A. **(b)** Model B.

Figure 4.14: Distribution of estimation error in net pay estimation on data from well 7220/5-1.



(a) Model A. **(b)** Model B.

Figure 4.15: Distribution of estimation error in net pay estimation on data from well 7220/8-1.

The box plots of well 7220/7-1 (Figure 4.13) show that for both models, the box of shaly sand samples was wider than the box of clean sand samples. This indicates a wider range of errors in the shaly sand samples, similar to the distribution of errors in the shaly sand samples of well 7220/7-3 S (c.f. Section 4.2). The median of errors in the clean sand samples was closer to zero (-0.01) compared to the median of errors in the shay sand samples (-0.08), which implies that the net pay estimation method was generally more accurate in estimating clean sand samples compared to shaly sand samples. The maximum and minimum values at the end of the whiskers were smaller in the clean sand samples than the shaly sand samples. This confirms the network's tendency to generate a less accurate estimation in shaly sand samples, which might be caused by a limitation on logging tool resolution.

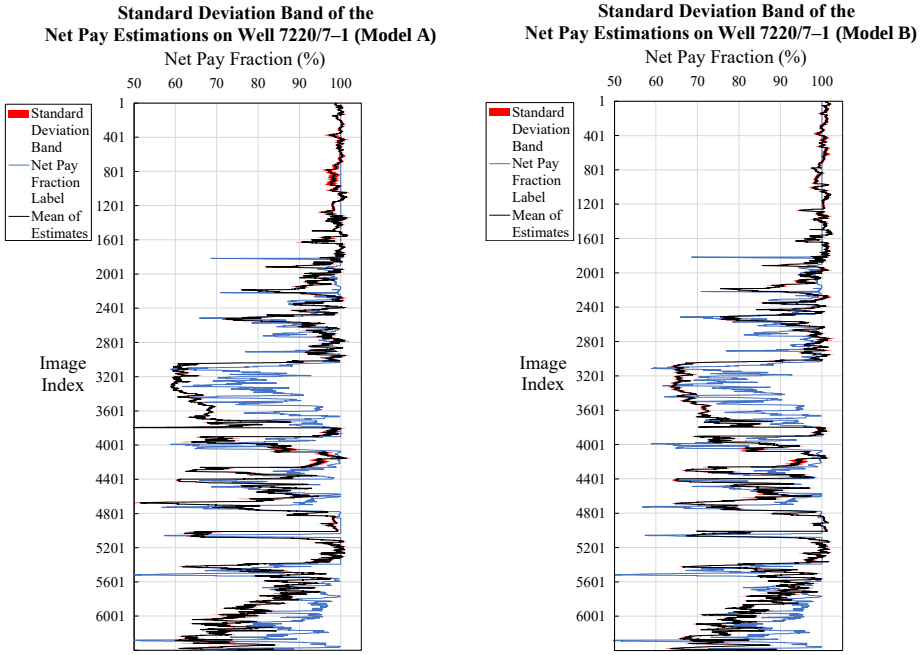
Well 7220/5-1 and 7220/8-1 contain mostly clean sand samples; therefore, their box plots (Figure 4.14) only display the distribution of estimation error in the clean sand samples. The box plots of well 7220/5-1 show that for both models, the errors were mostly close to zero, indicated by the median and width of the boxes. The box plots of well 7220/8-1 (Figure 4.15) shows a relatively wider box with larger maximum and minimum values compared to the boxes of well 7220/5-1. This was possibly due to the influence of the logging environment, which affected the logging tool response. Generally, the plots of both wells show that the range of errors in clean sand samples was not wide. This was an expected result as the network was generally more accurate in estimating clean sand samples.

Across the three wells, the box plots showed a distribution of error with a negative skew. This was indicated by a mean which was lower than the median and by the wider lower skewer. This confirms the founding in the previous net pay estimation (c.f. Section 4.2),

in which the network had a tendency of underestimating the net pay value of a sample. Several outliers were present in the error distribution of clean sand samples. As in the previous net pay estimation, these outliers might represent samples with a calcite cemented sand interval.

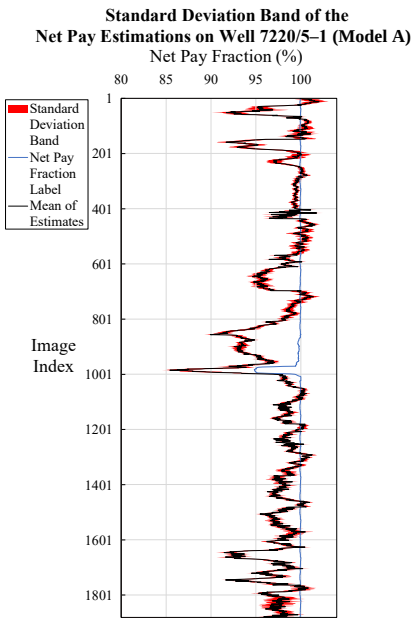
Reliability Evaluation

As in the previous net pay estimation (c.f Section 4.2), the reliability of estimation was evaluated to ensure that the estimation was independent of horizontal variation in the image logs. The reliability of the net pay estimation method was evaluated by performing net pay estimation on eight test sets with varying horizontal orientation. The variability within the eight estimates was observed by calculating the standard deviation of the estimates in each image index. Since the net pay estimates were a continuous value, a standard deviation band was generated to give a better visualization of the variability (c.f. Section 3.7.2). The standard deviation of each well based on the net pay estimations using Model A and B are shown in Figure 4.16 to 4.17.

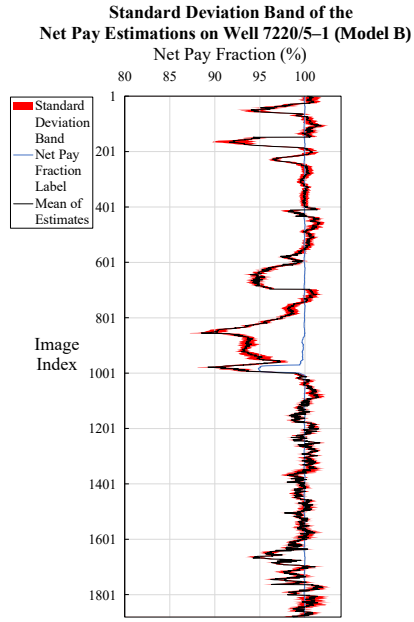


(a) Model A. **(b)** Model B.

Figure 4.16: Standard deviation band of the net pay estimations using Model A and B on data from well 7220/7-1.

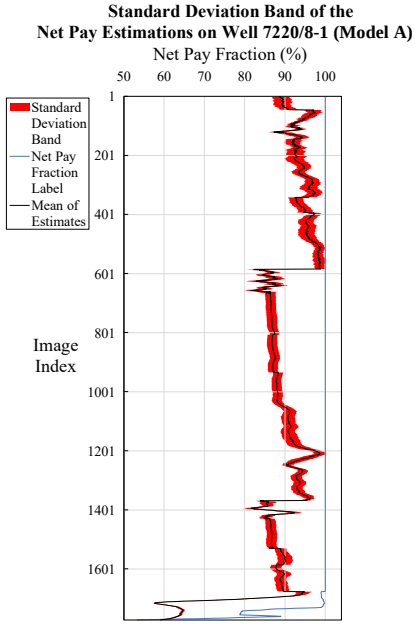


(a) Model A.

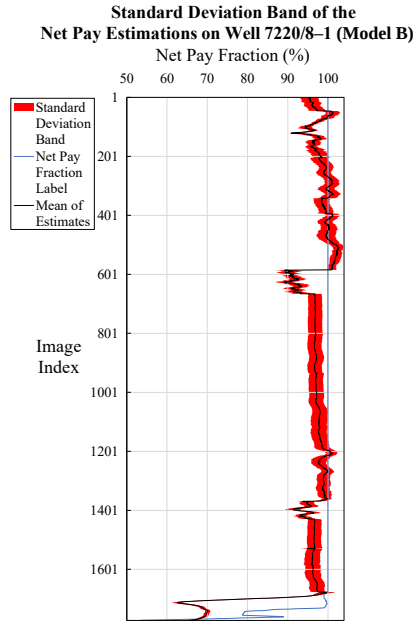


(b) Model B.

Figure 4.17: Standard deviation band of the net pay estimations on data from well 7220/5-1.



(a) Model A.



(b) Model B.

Figure 4.18: Standard deviation band of the net pay estimations on data from well 7220/8-1.

The standard deviation bands across the three different well show that the variability within the eight estimates for each well was generally low, indicated by the absence of excessively wide standard deviation band. However, the standard deviation bands of well 7220/8-1 were relatively wider compared to the rest two wells. This was possibly caused by the influence of a distinct structure such as fault, fractures, or a tilted layer boundary in the formation. These structures were usually displayed as sinusoidal lines in the image log. As the test sets were shifted by a certain pixel, the sinusoidal also got shifted and ended up with a different sinusoidal shape compared to the original shape (Figure 4.19). Since the network was trained based on the original shape, it might be confused by the presence of a different sinusoidal shape, and thus generating net pay estimates with a slightly higher variability on test sets with varying horizontal orientation. However, the eight estimates on well 7220/8-1 were not exceptionally different from each other in terms of trend and test losses. Therefore, the estimation on well 7220/8-1 was still regarded as a reliable estimation.

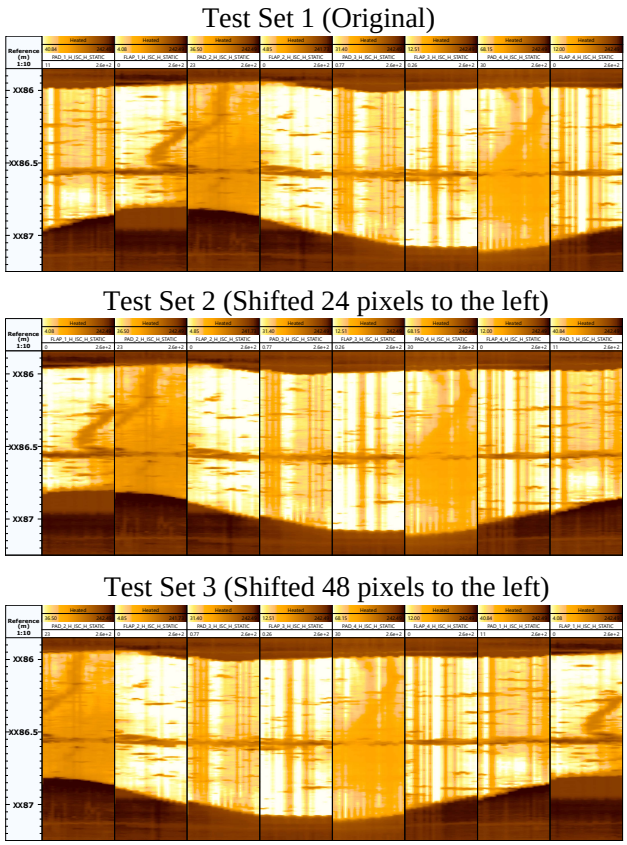


Figure 4.19: An example of a tilted layer boundary from an interval in well 7220/8-1, which is represented by the sinusoidal line in the image log. As the test set was shifted horizontally, the shape of the boundary changed.

A relatively narrow standard deviation band indicates that the estimates from each test within the same well were relatively close to each other in terms of trend and values. This is supported by the test losses of the eight estimations for each well, outlined in Table 4.9. The test losses among the eight estimations on each well were generally close to each other. By being able to generate net pay estimations with low variability in a horizontally shifted test sets, it implies that the estimation method was generally not affected by horizontal orientation in the image logs.

Table 4.9: Test loss from net pay estimation on eight test sets of well 7220/7-1, 7220/5-1 and 7220/8-1.

	Test Loss of Well 7220/7-1		**Test Loss of Well 7220/5-1**		**Test Loss of Well 7220/8-1**	
Test Set	**Model A**	**Model B**	**Model A**	**Model B**	**Model A**	**Model B**
Test Set 1 (Original)	0.0123	0.0090	0.0009	0.0006	0.0107	0.0036
Test Set 2	0.0123	0.0090	0.0010	0.0007	0.0101	0.0028
Test Set 3	0.0121	0.0087	0.0009	0.0006	0.0138	0.0035
Test Set 4	0.0134	0.0089	0.0008	0.0007	0.0096	0.0027
Test Set 5	0.0136	0.0092	0.0009	0.0008	0.0133	0.0034
Test Set 6	0.0128	0.0100	0.0010	0.0009	0.0130	0.0047
Test Set 7	0.0121	0.0086	0.0009	0.0008	0.0173	0.0047
Test Set 8	0.0122	0.0082	0.0007	0.0009	0.0132	0.0040
Average	0.0126	0.0090	0.0009	0.0008	0.0126	0.0037

4.4 Net Pay Estimation on Reservoir Intervals Without Core Photographs

In this net pay estimation, the capability of the optimized net pay estimation method was demonstrated in estimating net pay of a selected interval from one of the wells in the Johan Castberg field. This was an attempt to simulate the practical application of the machine learning-based net pay estimation method in estimating the net pay of an uncored reservoir interval. The selected interval was comprised of several sub-intervals with or without core photographs. The sub-intervals without core photographs refer to the seal peel intervals, which were shown as blank spaces in the core UV photographs. The estimates in the sub-intervals with core photographs were plotted alongside the net pay fraction label from the corresponding core UV photograph of that portion. Therefore the accuracy of the estimation can also be observed in the sub-intervals with core photographs.

A 15 m interval from well 7220/7-1 was selected in this net pay estimation. The selected interval was comprised of both clean sand and shaly sand section. Model B from the previous net pay estimation was utilized in generating the estimation, and the training and validation samples were taken from well 7220/7-3 S. Model B was selected instead of Model A because it had a better performance in terms of accuracy and estimates trend. Net

pay estimation was performed every 25 cm of the interval; therefore, the selected image log interval from well 7220/7-1 was split into 60 smaller intervals without overlap. The details of the net pay estimation are outlined in Table 4.10.

Table 4.10: Details of the datasets used in the net pay estimation of reservoir intervals without core photographs.

Dataset	Number of image	Source
Training set	4468	7220/7-3 S
Validation set	971	7220/7-3 S
Test set	60	7220/7-1

In this net pay estimation, the image log and core UV photograph were placed side by side to visualize the similarity between them. The blank spaces in the core UV photograph represent the sub-intervals without core photographs. The net pay fraction labels and estimates were plotted together on the same track to observe the estimates' accuracy in intervals with core photographs. The net pay fraction label curve possessed several missing values, which corresponds to missing core UV photograph sections due to seal peel preservation. The absence of a core UV photograph in the seal peel interval implies that the net pay fraction label could not be generated in that interval, thus creating a missing value in the net pay fraction label curve. The result of the estimation is shown in Figure 4.20.

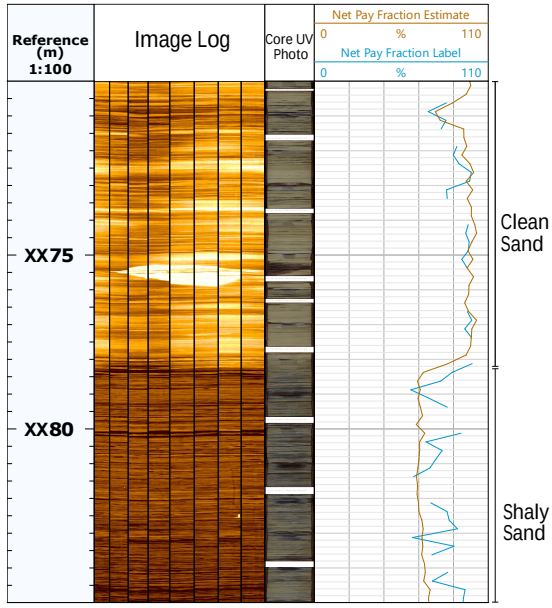


Figure 4.20: Net pay estimation of a reservoir interval in well 7220/7-1. The interval was comprised of a clean and shaly sand section. The blank spaces in the core UV photograph represent sub-intervals without core photographs.

The result shows that the net pay estimates were generally accurate in estimating the net pay of the image log. The estimates were able to follow the trend of the net pay fraction labels in the clean and shaly sand section. Although in the shaly sand section the estimates did not reproduce the fluctuation in the labels, the estimates still show the net pay was lower than in the clean sand section without excessively underestimating the values.

The result also shows the practical application of the net pay estimation method in estimating a reservoir interval without core data. In the plot of net pay estimates and labels, it is shown that the estimates were able to fill the missing net pay value due to the absence of a core photograph. The estimation also did not involve any core comparison process or resistivity threshold to estimate the net pay. This implies that the machine learning-based net pay estimation method has the potential to be implemented as an alternative net pay estimation method in a situation where core information was not available or not reliable. Such situations may arise when drilling in an unconsolidated formation, which might lead to the acquisition of poorly consolidated cores which are easy to break apart (Bajsarowicz, 1992). The net pay estimation method may also be implemented to estimate net pay on development wells, where cores are not regularly acquired due to cost.

4.5 Limitation and Recommendations

4.5.1 Limitation of the Net Pay Estimation Method

In this study, the machine learning-based net pay estimation method was successfully optimized. Compared to the preceding study, the results showed improvements in terms of loss minimization and estimation accuracy. Based on the accuracy analysis and reliability evaluation of the results in Section 4.2 and 4.3, several limitations of the net pay estimation method were observed. The limitations are outlined in this section.

Borehole Image Logging Tool Resolution

The resolution of the borehole image logging tool, which was used to generate the image logs, clearly influenced the estimation quality and accuracy. This was indicated by the estimation in shaly sand samples, which were generally less accurate than the clean sand samples. The Fullbore Micro-Imager (FMI) tool used in this study has a resolution of 0.2 inch or 5.08 cm, which means structures or layers smaller than 5.08 cm will not be clearly identified in the image log. On the other hand, the core UV photographs were able to identify the layers thinner than 5.08 cm. Therefore, it was generally more difficult to achieve a perfect correlation between image log samples and the net pay values in the shaly sand interval.

Influence of Logging Environment on the Response of Image Logging Tool

The wells in the Johan Castberg field were drilled the same borehole fluid type with a different fluid salinity. Table 4.11 shows that the borehole fluid salinity of well 7220/7-1 is significantly higher than the rest of the wells, and it possibly influenced the image log measurement of that well. Although well 7220/8-1 was comprised of mostly clean sand

samples, the estimations on its samples were generally less accurate than the estimations on samples from well 7220/5-1, which was also comprised of clean sand samples.

Table 4.11: The logging environments of the Johan Castberg wells showing varying borehole fluid salinity.

Well	Borehole Fluid Type	Borehole Fluid Salinity (ppm)
7220/5-1	Water Based Mud	35508
7220/7-1	Water Based Mud	45372
7220/7-3 S	Water Based Mud	44801
7220/8-1	Water Based Mud	57921

A resistivity-based image logging tool is generally sensitive to the characteristic of the borehole fluid. For example, the FMI tool used in this study was more suitable for wells drilled with a water-based mud than wells drilled with an oil-based mud (Schlumberger, 2013). Measurement of formation resistivity in a less conductive environment, such as an oil-based mud environment, is generally less accurate than in a water-based mud environment. Since the net pay estimation method relies heavily on image logs, a significantly different image log response for a similar lithology might inhibit the generation of a more accurate estimation.

Model Compatibility

One of the models generated by the optimized network, Model A, had a relatively better estimation on the shay sand samples of well 7220/7-1 compared to the shaly sand samples of well 7220/7-3 S. This indicates that there was a limitation in the model compatibility. Although the model generated a relatively good estimation in samples from a particular well, it does not imply that it will always be compatible for estimating any type of image log samples. This agrees with the axiom which states that there is no model which is universally suitable for every problem (Gómez and Rojas, 2015).

A model generates estimations or predictions based on some assumptions formulated during the training process. The effectiveness of a model is dependent on how well the assumptions fit the true nature of the data. Therefore a good knowledge of the nature of the data is essential as it will help determine which model will be compatible for some datasets with some shared characteristics. In this net pay estimation case, several different models can be generated with different specialization, e.g. models specialized for datasets with mostly shaly sand samples or models specialized for datasets from a high salinity well. Although the model generated in this study might not be universally compatible with any image logs, the formulated workflow, optimization approaches, and insights gained from the results might be useful for similar research.

4.5.2 Recommendations for the Net Pay Estimation Method

Based on the results evaluation and observed limitations of the optimized net pay estimation method, several recommendations were formulated for the continuation of this study.

Utilization of Image Logs With a Better Quality

More recent image logging tools with a better specification have been utilized by the industry. They can deliver image logs with better quality and resolution. One of the tools is the Quanta Geo tool by Schlumberger, which is equipped with eight pads that can cover up to 98% borehole coverage in a standard 8-in well (Schlumberger, 2014). The tool also has a higher vertical resolution (0.24 inch) than the conventional image logging tool for an oil-based mud environment (1.2 inches). Image logs with better borehole coverage and better resolution might give a better correlation with the net pay value.

Introduction of Additional Training Samples From Different Wells or Fields

The net pay estimation results show that the introduction of additional image log samples led to a better loss minimization and more accurate net pay estimations. The introduction of additional samples with different characteristics from different wells or fields may generate a model with a better generalization to unseen data. This is due to the network's exposure to more variations in the image log samples, e.g. lithology or logging environment variation.

Net Pay Estimation on Data From a Different Field

The reservoir formation in wells from the same field may have a similar depositional environment, which leads to a similar lithology or geological property. However, reservoir intervals across different fields may have a different depositional environment, sedimentary structures, or other geological properties. Wells across different fields might be different in terms of logging environment, deviation, or other characteristics due to different drilling program. By performing net pay estimation using the optimized method on samples from a different field, the method's capability in estimating image log samples with a wider variation can be observed and evaluated.

Chapter 5

Conclusion

An alternative net pay estimation method based on the implementation of machine learning on borehole image logs interpretation was successfully developed and optimized in this study. Machine learning technique was implemented through the use of a convolutional neural network (CNN), which will attempt to map correlations between the training data (borehole image logs) and their corresponding labels (net pay fraction from core ultraviolet photographs). The network was trained with data from one well in the Johan Castberg field, and it was able to generate an accurate and reliable net pay estimation on unseen data from the other wells in the field.

The data processing procedures from a preceding study were optimized with several optimization approaches to improve the quality of the image log samples and net pay fraction labels. The machine learning implementation was also optimized by implementing a more robust CNN design and better curation of training samples. These optimization approaches were aimed to improve the net pay estimation from borehole image logs.

Through a set of net pay estimations, the capability of the method was evaluated. The results showed that the estimates were generally accurate with matching trends with the net pay fraction labels. The estimations were also generally reliable, indicated by a low variability in the estimates with varying horizontal orientation. Evaluation of the results proved that the optimization managed to generate a model with more accurate estimations compared to the preceding study. The net pay estimation method has the potential to be implemented in a situation where core information was not available or not reliable.

The net pay estimations also highlighted several limitations of the net pay estimation method, which were caused by image logs resolution, model compatibility, or variations in the logging environment. Based on the limitations and results evaluation, several recommendations for the continuation of the study were formulated, e.g. utilization of image logs with better quality, the introduction of additional training samples from different wells or fields, and net pay estimations on data from a different field.

Bibliography

Al-Saddique, M.A., Hamada, G.M., Al-Awad, M.N.J., 2000. State of the Art: Review of Coring and Core Analysis Technology. Journal of King Saud University - Engineering Sciences 12, 117–137. URL: http://www.sciencedirect.com/science/article/pii/S1018363918307098, doi:10.1016/S1018-3639(18)30709-8.

Anyaehie, J., 2015. Regional (Niger Delta) Facie Based Vsh Cut-Off For Net Reservoir Sand Definition, Society of Petroleum Engineers. doi:10.2118/178322-MS.

Archer, J.S., Wall, C.G., 1986. Characteristics of Reservoir Rocks, in: Archer, J.S., Wall, C.G. (Eds.), Petroleum Engineering: Principles and Practice. Springer Netherlands, Dordrecht, pp. 62–91. URL: https://doi.org/10.1007/978-94-010-9601-0_5, doi:10.1007/978-94-010-9601-0_5.

Bajsarowicz, C.J., 1992. Core Alteration and Preservation: Part 3. Wellsite Methods 95, 127–130. URL: http://archives.datapages.com/data/specpubs/methodo1/data/a095/a095/0001/0100/0127.htm. publisher: AAPG Special Volumes.

Brownlee, J., 2017. Long Short-Term Memory Networks With Python. URL: https://machinelearningmastery.com/lstms-with-python/#packages.

Dertat, A., 2019. Applied Deep Learning - Part 4: Convolutional Neural Networks. URL: https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2. retrieved on 4 Dec 2019.

Donselaar, M.E., Schmidt, J.M., 2005. Integration of outcrop and borehole image logs for high-resolution facies interpretation: example from a fluvial fan in the Ebro Basin, Spain. Sedimentology 52, 1021–1042. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1365-3091.2005.00737.x, doi:10.1111/j.1365-3091.2005.00737.x.

Endres, H., Lohr, T., Trappe, H., Samiee, R., Thierer, P., Krawczyk, C., Tanner, D., Oncken, O., Kukla, P., 2008. Quantitative fracture prediction from seismic data. Petroleum Geoscience 14, 369–377. doi:10.1144/1354-079308-751.

Fanchi, J.R., 2010. Chapter 2 - Fluid Properties, in: Fanchi, J.R. (Ed.), Integrated Reservoir Asset Management. Gulf Professional Publishing, Boston, pp. 17–32. URL: http://www.sciencedirect.com/science/article/pii/B9780123820884000025, doi:10.1016/B978-0-12-382088-4.00002-5.

Firdaus, I., 2010. Image Log Processing and Interpretation for Fracture and litho-facies Characterization; Case Study of Ujung Pangkah Field, East Java, in: Proc. Indon Petrol. Assoc., 34th Ann. Conv., Indonesian Petroleum Association (IPA). URL: http://archives.datapages.com/data/ipa_pdf/081/081001/pdfs/IPA10-G-011_Po.htm, doi:10.29118/IPA.2401.10.G.011.

Gong, B., Keele, D., Toumelin, E., Clinch, S., 2019. Estimating Net Sand From Borehole Images in Laminated Deepwater Reservoirs With a Neural Network, Society of Petrophysicists and Well-Log Analysts. doi:10.30632/PJV60N5-2019a4.

Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. MIT Press. URL: http://www.deeplearningbook.org.

Gómez, D., Rojas, A., 2015. An Empirical Overview of the No Free Lunch Theorem and Its Effect on Real-World Machine Learning Classification. URL: https://www.mitpressjournals.org/doix/abs/10.1162/NECO_a_00793, doi:10.1162/NECO_a_00793. publication Title: Neural Computation Publisher: MIT Press One Rogers St., Cambridge, MA 02142-1209 USA journals-info@mit.edu.

Ifinnmark, 2019. Statoil dropper elektrifisering på Castberg-feltet. URL: https://www.ifinnmark.no/olje-og-energi/oljebransjen/nyheter/statoil-dropper-elektrifisering-pa-castberg-feltet/s/5-81-333577. retrieved on 14 Dec 2019.

Kalukal, F., 2014. FMI & UBI Image QC Report. URL: https://geoscience.nt.gov.au/gemis/ntgsjspui/bitstream/1/83802/2/PR2015-0030_Tanumbirini_1_FMI_and_UBI_QC_Processed_Report.pdf.

Kennedy, M., 2015. Chapter 3 - Core and Other Real Rock Measurements, in: Kennedy, M. (Ed.), Developments in Petroleum Science. Elsevier. volume 62 of *Practical Petrophysics*, pp. 73–88. URL: http://www.sciencedirect.com/science/article/pii/B9780444632708000037, doi:10.1016/B978-0-444-63270-8.00003-7.

Knaust, D., 2017. Selected Trace Fossils in Core and Outcrop, in: Knaust, D. (Ed.), Atlas of Trace Fossils in Well Core : Appearance, Taxonomy and Interpretation. Springer International Publishing, Cham, pp. 27–206. URL: https://doi.org/10.1007/978-3-319-49837-9_5, doi:10.1007/978-3-319-49837-9_5.

Lane, D., 2013. Introduction to Statistics: An Interactive eBook. Rice University. URL: https://books.apple.com/us/book/introduction-to-statistics-an-interactive-e-book/id684001500.

Leonard, Z.S., 2016. Development of a downhole ultrasonic transducer for imaging while drilling, in: 2016 IEEE International Ultrasonics Symposium (IUS), pp. 1–4. doi:10.1109/ULTSYM.2016.7728541. iSSN: 1948-5727.

Li, Y., 2018. Deep Reinforcement Learning. arXiv:1810.06339 [cs, stat] URL: http://arxiv.org/abs/1810.06339. arXiv: 1810.06339 version: 1.

Luthi, S.M., 2001. Electrical Borehole Imaging, in: Luthi, S.M. (Ed.), Geological Well Logs: Their Use in Reservoir Modeling. Springer, Berlin, Heidelberg, pp. 74–123. URL: https://doi.org/10.1007/978-3-662-04627-2_5, doi:10.1007/978-3-662-04627-2_5.

Marsland, S., 2014. Machine Learning: An Algorithmic Perspective, Second Edition. 2nd ed., Chapman & Hall/CRC.

McPhee, C., Reed, J., Zubizarreta, I., 2015. Core Analysis: A Best Practice Guide. volume 64. Elsevier.

Metwalli, F.I., Shendi, E.A.H., Fagelnour, M.S., 2017. Reservoir Petrophysical Modeling and Risk Analysis in Reserve Estimation; A Case Study from Qasr Field, North Western Desert, Egypt. IOSR Journal of Applied Geology and Geophysics 05, 41–52. doi:10.9790/0990-0502014152.

Miralles, L., Rosso, D., Jiménez, F., Garcia, J., 2016. A methodology based on Deep Learning for advert value calculation in CPM, CPC and CPA networks. Soft Computing 21, 1–15. doi:10.1007/s00500-016-2468-4.

MissingLink, 2019a. Fully Connected Layers in Convolutional Neural Networks: The Complete Guide. URL: https://missinglink.ai/guides/convolutional-neural-networks/fully-connected-layers-convolutional-neural-networks-complete-guide/. retrieved on 4 Dec 2019.

MissingLink, 2019b. Using the Keras Flatten Operation in CNN Models with Code Examples. URL: https://missinglink.ai/guides/keras/using-keras-flatten-operation-cnn-models-code-examples/. retrieved on 4 Dec 2019.

Mæland, A.O., 2014. Shaly Sand Petrophysics and its Impact on Full Field Reservoir Simulation. URL: http://hdl.handle.net/11250/2445665.

NIH, 2019. ImageJ. URL: https://imagej.nih.gov/ij/features.html. retrieved on 12 Dec 2019.

NPD, 2019a. Diskos. URL: https://www.npd.no/en/about-us/organisation/collaboration-projects/diskos/. retrieved on 12 Dec 2019.

NPD, 2019b. Johan Castberg. URL: https://www.norskpetroleum.no/en/facts/field/johan-castberg/. retrieved on 12 Dec 2019.

Revie, D., 2017. Unconventional petroleum resources of the Roper Group, McArthur Basin. Northern Territory Geological Survey. URL: https://geoscience.nt.gov.au/gemis/ntgsjspui/handle/1/85134.

Samanpour, A.R., Ruegenberg, A., Ahlers, R., 2018. The Future of Machine Learning and Predictive Analytics, in: Linnhoff-Popien, C., Schneider, R., Zaddach, M. (Eds.), Digital Marketplaces Unleashed. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 297–309. URL: https://doi.org/10.1007/978-3-662-49275-8_30, doi:10.1007/978-3-662-49275-8_30.

Schlumberger, 2013. FMI-HD High-definition formation microimager. URL: https://www.slb.com/reservoir-characterization/surface-and-downhole-logging/wireline-openhole-logging/fmi-fullbore-formation-microimager.

Schlumberger, 2014. Quanta Geo Photorealistic Reservoir Geology Service. URL: https://www.slb.com/reservoir-characterization/surface-and-downhole-logging/wireline-openhole-logging/quanta-geo-microresistivity-imager.

Schlumberger, 2019a. Advanced Borehole Geology Suite. URL: https://www.ocean.slb.com/en/plug-ins/plugindetails?ProductId=PABG-G1. retrieved on 12 Dec 2019.

Schlumberger, 2019b. Techlog. URL: https://www.software.slb.com/products/techlog/techlog-petrophysics. retrieved on 12 Dec 2019.

Shahinpour, A., 2013. Borehole image log analysis for sedimentary environment and clay volume interpretation. 82 URL: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/240255.

Snedden, J.W., 1984. Validity of the Use of the Spontaneous Potential Curve Shape in the Interpretation of Sandstone Depositional Environments 34. URL: http://archives.datapages.com/data/gcags/data/034/034001/0255.htm.

Srivastava, T.N., 2008. Statistics for Management. Tata McGraw-Hill Education.

Torres, J., 2016. First Contact with TensorFlow: Get Started with Deep Learning Programming. Watch this space collection, Universitat Politècnica de Catalunya, Centro Nacional de Supercomputación. URL: https://books.google.no/books?id=Ws3UwAEACAAJ.

Vieira, D., Paixao, J.A.R., 2018. Vector Field Neural Networks. Ph.D. thesis. Federal University of Rio de Janeiro. doi:10.13140/RG.2.2.32353.35688.

Wang, F., Bian, H.y., Gao, X.h., Pan, B.z., 2016. Application of fullbore formation microimager logging in the evaluation of anisotropic resistivity in a thin interbed reservoir. Journal of Geophysics and Engineering 13, 454–461. URL: https://academic.oup.com/jge/article/13/4/454/5111001, doi:10.1088/1742-2132/13/4/454.

Wheaton, R., 2016. Chapter 6 - Estimation of Reserves and Drive Mechanisms, in: Wheaton, R. (Ed.), Fundamentals of Applied Reservoir Engineering. Gulf Professional Publishing, pp. 127–136. doi:10.1016/B978-0-08-101019-8.00006-5.

Worden, R.H., Morrall, G.T., Kelly, S., Ardle, P.M., Barshep, D.V., 2019. A renewed look at calcite cement in marine-deltaic sandstones: the Brent Reservoir, Heather Field, northern North Sea, UK. Geological Society, London, Special Publications 484. URL: https://sp.lyellcollection.org/content/early/2019/04/09/SP484-2018-43, doi:10.1144/SP484-2018-43. publisher: Geological Society of London Section: Research article.

Worthington, P.F., 2010. Net Pay–What Is It? What Does It Do? How Do We Quantify It? How Do We Use It? SPE Reservoir Evaluation & Engineering 13, 812–822. doi:10.2118/123561-PA.

Yamashita, R., Nishio, M., Do, R.K.G., Togashi, K., 2018. Convolutional neural networks: an overview and application in radiology. Insights into Imaging 9, 611–629. URL: https://doi.org/10.1007/s13244-018-0639-9, doi:10.1007/s13244-018-0639-9.

Appendix

The script can be accessed in the following link:
https://github.com/mikaelyuan/A-Machine-Learning-Approach-for-Net-Pay-Estimation-from-Borehole-Image-Logs

CNN Script

```
from numpy.random import seed
seed(33)
import numpy as np
import keras
from keras.layers import Input, Dense, Conv2D, MaxPooling2D, Flatten,
↪  Reshape, Dropout
from keras.models import Model, Sequential
from keras.callbacks import History
from keras.callbacks import CSVLogger
from keras.callbacks import EarlyStopping, ModelCheckpoint
from keras.preprocessing.image import ImageDataGenerator
from keras import regularizers, optimizers
import matplotlib.pyplot as plt
import matplotlib.ticker
import time
import os
import logging
logging.getLogger('tensorflow').disabled = True
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'
os.system('cls')

t = time.time()

#Loading the dataset
TrainData = np.load('DataTrain.npy')
TrainLabel = np.load('LabelTrain.npy')
print('Training data: ',TrainData.shape)

ValidationData = np.load('DataValidation.npy')
ValidationLabel = np.load('LabelValidation.npy')
print('Validation data: ',ValidationData.shape)

TestData = np.load('DataTest.npy')
TestLabel = np.load('DataTest.npy')
print ('Test data: ',TestData.shape)

#Normalization of the array values
max_value1 = float(TrainData.max())
```

```
max_value2 = float(ValidationData.max())
max_value3 = float(TestData.max())
TrainData = TrainData.astype('float32') / max_value1
ValidationData = ValidationData.astype('float32') / max_value2
TestData = TestData.astype('float32') / max_value3

#Reshaping  & Data augmentation
TrainData = TrainData.reshape(-1, 100, 192, 1)
ValidationData = ValidationData.reshape(-1, 100, 192, 1)
TestData = TestData.reshape(-1, 100, 192, 1)

DataAugmentation = ImageDataGenerator(vertical_flip=True,
↪  horizontal_flip=True, width_shift_range=97,fill_mode="wrap")
DataAugmentation.fit(TrainData, augment=True)

#CNN Architecture
Model = Sequential()
#Convolutional layers
Model.add(Conv2D(16, kernel_size=(3, 3), strides=(2, 2), activation =
↪  'relu', padding='same', input_shape=(100,192,1)))
Model.add(MaxPooling2D(pool_size=(2, 2),padding='same'))
Model.add(Conv2D(32, kernel_size=(3, 3), activation = 'relu',
↪  padding='same'))
Model.add(MaxPooling2D(pool_size=(2, 2),padding='same'))
Model.add(Conv2D(64, kernel_size=(3, 3), activation = 'relu',
↪  padding='same'))
Model.add(MaxPooling2D(pool_size=(2, 2),padding='same'))
Model.add(Conv2D(128, kernel_size=(3, 3), activation = 'relu',
↪  padding='same'))
Model.add(MaxPooling2D(pool_size=(2, 2),padding='same'))
Model.add(Conv2D(256, kernel_size=(3, 3), activation = 'relu',
↪  padding='same'))
Model.add(MaxPooling2D(pool_size=(2, 2),padding='same'))
Model.add(Conv2D(512, kernel_size=(3, 3), activation = 'relu',
↪  padding='same'))
Model.add(MaxPooling2D(pool_size=(2, 2),padding='same'))
Model.add(Flatten())
#Fully connected layers
Model.add(Dense(512, activation='relu'))
Model.add(Dense(256, activation='relu'))
Model.add(Dense(128, activation='relu'))
Model.add(Dense(64, activation='relu'))
Model.add(Dense(32, activation='relu'))
Model.add(Dense(1, activation='linear'))

#CNN hyperarameter
adam_mod = keras.optimizers.Adam(lr=0.0001)
Model.compile(optimizer=adam_mod, loss='mean_squared_error')
Stop = EarlyStopping(monitor='val_loss',
                     mode='min',
                     patience=25,
                     restore_best_weights=True)
Batch = 64
csv_logger = CSVLogger('Loss Log.csv', append=True, separator=';')
Train = Model.fit_generator(DataAugmentation.flow(TrainData, TrainLabel,
↪  batch_size=Batch),
                            epochs=100,
```

```
                                steps_per_epoch=  TrainData.shape[0]//Batch,
                                validation_data= (ValidationData,
                                ↪  ValidationLabel),
                                shuffle = True,
                                callbacks=[Stop, csv_logger])

Prediction = Model.predict(TestData)
np.save("Net Pay Estimation.npy", Prediction)
A=np.load('Net Pay Estimation.npy')
np.savetxt("Net Pay Estimation.csv", A, delimiter=",")

PredictionError = Model.evaluate(TestData, TestLabel)
print('Test loss (MSE):', PredictionError)

elapsed = time.time() - t

#Plot of loss function
loss = Train.history['loss']
val_loss = Train.history['val_loss']
plt.figure()
plt.plot(loss, 'b', label='Training loss')
plt.plot(val_loss, 'r', label='Validation loss')
locator = matplotlib.ticker.MultipleLocator(5)
plt.gca().xaxis.set_major_locator(locator)
formatter = matplotlib.ticker.StrMethodFormatter("{x:.0f}")
plt.gca().xaxis.set_major_formatter(formatter)
plt.title('Training and validation loss of the network')
plt.ylabel('Mean Squared Error (MSE)')
plt.xlabel('Iteration')
plt.legend()
plt.savefig('Loss PLot.jpg')

#Printing elapsed time
print("Elapsed time: %.2f" % (elapsed/60),"min")
```

Mikael Yuan Estuariwinarno

A Machine Learning Approach for Net Pay Estimation from Borehole Image Logs

NTNU
Norwegian University of
Science and Technology