Doctoral theses at NTNU, 2021:384

Mikkel Leite Arnø

At-bit estimation of formation properties, lithology classification, and pressure control using machine learning

Doctoral thesis

NTNU
Norwegian University of Science and Technology
Thesis for the Degree of
Philosophiae Doctor
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Mikkel Leite Arnø

At-bit estimation of formation properties, lithology classification, and pressure control using machine learning

Thesis for the Degree of Philosophiae Doctor

Trondheim, December 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU

Norwegian University of
Science and Technology

NTNU
Norwegian University of Science and Technology

Thesis for the Degree of Philosophiae Doctor

Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

ISBN 978-82-326-5550-2 (printed ver.)
ISBN 978-82-326-5884-8 (electronic ver.)
ISSN 1503-8181 (printed ver.)
ISSN 2703-8084 (online ver.)

ITK report 2021-7-W

Doctoral theses at NTNU, 2021:384

Printed by NTNU Grafisk senter

Summary

The petroleum drilling process involves an array of complicated operations necessary to create circular wellbores to allow the exploration and recovery of hydrocarbons. Examples of these are overcoming the resistance, crushing, and removal of rock, maintaining wellbore stability, and avoiding fracture and influx of formation fluids. As wellbores can extend for several kilometers underground, much of the operation is hidden from the driller, which must rely on topside and downhole sensor measurements for indications of which actions are preferable, and also to understand the state of the process. The amount of available measurements has increased gradually, and it is of interest to convert these measurements to value. However, analysis of large amounts of data is challenging and time-consuming for humans, and too much information could even be counterproductive.

This Ph.D. thesis is concerned with the use of machine learning (ML) methods to make use of the data available during drilling. A deep reinforcement learning agent is trained to control bottomhole pressure during pipe connection in a simulated study. Also, a streaming learning method using deep neural networks (DNNs) is presented in order to both estimate logging while drilling (LWD) sensor readings, as well as classifying lithology. This method uses drilling parameters as inputs, and learns continuously from the available data stream to adapt to case-specific and changing conditions during operation. This approach is applied to real drilling data from the reservoir section of a field operated by Equinor. The results herein presented indicate that such a configuration can be used to provide preliminary at-bit indications of lithology characteristics, which can be used by the driller as a tool in the decision-making process. Such predictions can contribute to a better foundation for selection of suitable drilling parameters, leading to a safer

and more efficient drilling operation.

Preface

This thesis is submitted in partial fulfillment of the requirements for the degree of Philosophiae Doctor (Ph.D.) at the Norwegian University of Science and Technology (NTNU). The research has been conducted from August 2018 to 2021 at the Department of Engineering Cybernetics (ITK) under the supervision of Professor Ole Morten Aamo and the co-supervision of Dr. John-Morten Godhavn, Equinor.

Acknowledgments

First and foremost, I would like to thank my supervisors, Professor Ole Morten Aamo and Dr. John-Morten Godhavn. Thank you for your guidance, and for allowing me the autonomy to select research projects that excite me. I also want to thank Equinor's Real-Time Automated Lithology (REAL) team, for providing valuable insights from their experience in lithology classification, and for their interest in my work. This interest has been a motivating factor for me.

The Covid-19 situation has been challenging for several reasons, but especially in terms of social life. I would like to thank Joakim for our frequent walks during this period. Our work-related and casual conversations during walks have helped keep me sane during this time of home-office and limited social interaction. Next, I would like to thank my family. Your support has meant a lot to me during these last three years. Mom and Dad, thank you for pushing me and believing in me. I guess I turned out allright thanks to you. Last, but definitely not least, Julie. This time has been a rollercoaster, including the highs of exciting results and accepted papers, and the stressful periods of stagnation and emerging deadlines. You have been my rock through it all, the wind in my sails, and so on. Thank you.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

The term automation dates back to around 1946, and is attributed to an engineering manager at the Ford Motor Company [2]. At the time, it was used to describe the use of automatic devices and controls in mechanized production lines, i.e., replacing physical human labor in dull, repetitive and dangerous tasks in the context of manufacturing. Since then, the definition has broadened to include a wider range of processes, as well as inclusion of mental labor and cognition [3]. Data processing, analysis and decision making all fall under this more modern definition of automation, both in full and partial replacement of human labor [4]. New technologies have gradually allowed more computational power, while sensors are installed everywhere, monitoring both private households and industrial processes. At this time, many companies find themselves gathering more data from sensors than can possibly be converted to business value by humans.

Machine learning (ML) is a branch of artificial intelligence (AI) that has recently resurged due to the aforementioned technical advancements, and has been defined as "[...] the study of computer algorithms that allow computer programs to automatically improve through experience" [5]. ML is typically divided into three main subcategories: *supervised learning* - predict outputs y from inputs x for regression and classification problems, *unsupervised learning* - clustering, dimension reduction and feature extrac-

tion, and *reinforcement learning* - learning of optimal behavior to maximize some notion of reward. These methods all learn directly from observations without explicit programming, and can in different ways extract knowledge and create value from the data. Lately, the development of deep learning has been receiving attention due to remarkable ability to extract latent features from input data through multiple layers of neurons in deep neural networks (DNNs), allowing learning of representations with no theoretical limitations of complexity [6]. Exciting research results include deep convolutional neural networks (CNNs) for image classification over 1000 classes [7] and self-driving cars [8], deep reinforcement learning (RL) agents surpassing or performing comparably to professional human game testers at 49 Atari games [9], and solving simulated physics tasks [10].

Also in various industrial sectors, data-driven approaches are subject to extensive research [11], aiming to make use of available data. Among them is the petroleum drilling industry, which is inherently expensive due to high capital and operating expenditures. The commercial viability of hydrocarbon reserves are thus dependent on both these costs, as well as hydrocarbon prices. Due to recent decline in price, it is vital to cut operating costs. Automation has been predicted to contribute significantly in this regard[1], although critics claim that automation will inhibit crews' ability to respond manually due to loss of competence [12]. Over time, several drilling tasks have been successfully automated, although wider adoption may be described as slow, where heterogeneity between wells and rigs, data integration between multiple stakeholders such as operator, service provider, drilling contractor and equipment manufacturer, as well as acceptance of automation in the drilling crew are among contributing factors to this trend [13]. The identification of areas where automation can benefit the drilling industry, as well as determining which level of automation is most appropriate, are not trivial tasks. The long-term goal is to reverse the roles so that machines take over the majority of the work load, while humans possess more of an advisory role. However, it has been found that while higher levels of automation can improve process efficiency, it also leads to an increased risk of operator error [14]. These findings suggest that a gradual transition to fully autonomous drilling systems, along with additional train-

[1] DNV Technology Outlook 2030 document available online at https://www.dnv.com/to2030/impact/impact-on-oil-and-gas.html, (accessed July 12, 2021)

ing of personnel in parallel with these advancements are key for the success of broader adoption of automation in drilling.

The focus of this Ph.D. project has been to explore the use of ML methods in the drilling operation. Primarily, efforts have been targeted at the development of data analysis systems, which pose low operational risk with regards to safety and efficiency, and as such may serve as stepping stones in the transition towards higher autonomy levels. These systems aim to relieve the driller of certain data analyses tasks during drilling, and provide valuable insight into bottomhole conditions. At the same time, decision making responsibility is currently left entirely to the driller. Motivations behind such a human/machine division of labor are to obtain acceptance among crews, to preserve competence, and to allow crews to focus on other tasks. In the future, when such systems have been further validated, increasing levels of advising and autonomy can be layered on top, for example by alerting the driller of events or changes in downhole conditions, suggestions of favorable actions or even automatic updates of drilling parameters.

1.2 Contributions

The research conducted is a contribution to the petroleum drilling industry. The majority of the work herein presented contributes to the at-bit estimation of formation properties and classification of drilled lithology. To this end, a novel streaming learning approach was developed. For these contributions, real surface drilling data and logging while drilling (LWD) measurements from the reservoir section of a field operated by Equinor were used. These contributions, with references to papers in Section 1.3, can be summarized as:

- Development of a divided and prioritized experience replay approach for streaming regression in Paper B, comparing the developed approach to the standard sliding window for density log estimation using a DNN.
- A case study of the developed streaming learning approach for at-bit density log estimation from drilling parameters in Paper A. This paper motivates a streaming learning approach by use of a t-distributed stochastic neighbor embedding (t-SNE) dimension reduction, and compares results to a standard deep learning approach.

- An application of the developed approach to at-bit neutron log estimation from drilling parameters in Paper C.
- An extension of the LWD estimation method to classify at-bit lithology by layering a DNN lithology classifier on top of the LWD models in Paper D. This work was initially covered in the conference paper, Paper E, which was invited for peer-review in the SPE Drilling & Completion journal.

Also, one instance of deep reinforcement learning has been applied in a simulated study for bottomhole pressure control:

- A deep reinforcement learning agent was embedded in a managed pressure drilling (MPD) system to control the bottomhole pressure using a topside choke valve with nonlinear characteristics during pipe connection in a simulated study in Paper F.

1.3 List of publications

This thesis is based on the papers listed below. Paper B has been accepted at the MethodsX journal. Papers C and D are accepted at the SPE/IADC International Drilling Conference and Exhibition, Galveston, Texas, 2022, and the SPE Drilling & Completion journal, respectively, but are yet to be published. In addition to these papers, the work on LWD estimation and lithology classification has resulted in the incipient development of a prototype application aimed for use internally at Equinor.

Paper A

> **Arnø ML**, Godhavn J-M, Aamo OM. At-bit estimation of rock density from real-time drilling data using deep learning with online calibration. In *Journal of Petroleum Science and Engineering*, page 1009006, 2021 https://doi.org/10.1016/j.petrol.2021.109006 [1]

Paper B

> **Arnø ML**, Godhavn J-M, Aamo OM. A divided and prioritized experience replay approach for streaming regression. *MethodsX (Accepted)* [15]

Paper C

Arnø ML, Godhavn J-M, Aamo OM. At-bit virtual neutron log estimated from real-time drilling data. *SPE/IADC International Drilling Conference and Exhibition, Galveston, Texas (Accepted).* March 8-10, 2022. [16]

Paper D

Arnø ML, Godhavn J-M, Aamo OM. Classification of drilled lithology in real-time using deep learning with online calibration. *SPE Drilling & Completion (Accepted)* [17]

Paper E

Arnø ML, Godhavn J-M, Aamo OM. Real-time classification of drilled lithology from drilling data using deep learning with online calibration. In *proceedings of the SPE/IADC International Drilling Conference and Exhibition, Virtual.* March 8-12, 2021. SPE-204093-MS https://doi.org/10.2118/204093-MS [18]

Paper F

Arnø ML, Godhavn J-M, Aamo OM. Deep reinforcement learning applied to managed pressure drilling. In *proceedings of the SPE Norway Subsurface Conference, Virtual.* November 2-3, 2020. SPE-200757-MS https://doi.org/10.2118/200757-MS [19]

1.4 Outline of thesis

The remainder of this thesis is structured as follows: Chapter 2 provides background on the areas to which the methods are applied, along with existing work in literature. Also, preliminaries for the methods used are given. Chapters 3 - 8 contain the papers on which this thesis is based. Finally, chapter 9 offers concluding remarks and suggestions for further work.

Chapter 2

Background

This chapter provides background on the areas of the drilling process that are the foci of this thesis. Previous work and methodology used are given as well. The aim is to put these into context and expand on them beyond what is done in the publications.

2.1 Lithology characterization

Lithology refers to the physical characteristics of rocks, including their mineral composition and texture. As downhole lithology is a central part of the environment that the drilling system is in interaction with, it is of great interest to answer the question of drilled lithology. Evaluation of well placement and reservoir structure, for example, depend on this. Also, selection of suitable drilling parameters are dependent on lithology [20], both in terms of safety and efficiency. For example, in transitions from soft to hard lithology, the risk of buckling, washout, severe doglegs and unwanted vibrations is increased if proper actions are not taken. Another common dysfunction is loss of circulation which could leak to pipe sticking [21]. The risk of loss of circulation is increased in inherently fractured and cavernous lithologies, and thus, correct classification of these lithologies is of importance to allow the driller to take proper actions to reduce this risk.

Traditionally, determining lithology has been done manually by geologists through interpretation of logging while drilling (LWD) logs [22]. LWD tools

are a family of downhole sensors that communicate to the rig via mudpulse telemetry at a sampling period of $\sim$ 20 s. Each log provides insight into the physical properties of lithology, and examples of these are the density log, neutron porosity log, gamma ray log, resistivity log, and the caliper log. The density log measures the bulk density of rock, which can separate softer lithologies from stringers, while the neutron porosity log measures the porosity, which can be used to detect oil and gas. The gamma ray log measures gamma radiation, which can be used to separate claystones from other sedimentary rocks. Resistivity is used to determine water saturation. Lastly, the caliper log provides a measure of the borehole quality. Alone, each of these provide valuable downhole information, and in combination, they can be used to obtain an accurate understanding of lithology. However, as illustrated in Figure 2.1, they are typically installed some distance behind the bit, the rotary steerable system (RSS) and the mud motor. This distance could be 0.5-30 meters, typically 7-30 meters, which, depending on rate of penetration (ROP) can amount to delays of approximately 20-120 minutes.

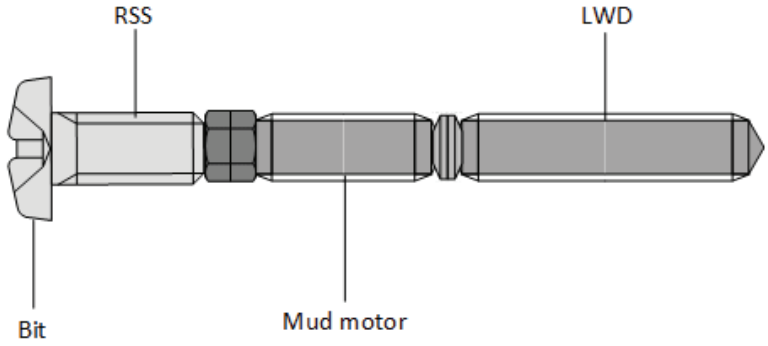

Figure 2.1: Illustration showing where the LWD sensors are placed compared to the bit.

Supervised learning in the context of LWD tools has been the focus of several previous publications. As LWD tools are expensive, several publications have focused on the estimation of one log from other logs, both to cut cost of equipment, and in case of log failure. Examples include the use of DNNs [23], long short-term memory models (LSTM) [24], conditional variational autoencoders (CVAE) [25], and Bayesian neural networks [26]. Studies to classify lithology from drilling data have also been conducted. Since LWD measurements are established as good predictors for lithology,

many of these studies rely on LWD data as inputs to their models. Scalable gradient boosted decision trees [27], support vector machines (SVMs) [28], and bi-directional gated recurrent units (GRUs) [29, 30] are among the methods applied to this end. Since these publications all use LWD data as input, they do not solve the problem of the delay due to log placement. Due to this, a driller making decisions based on estimates or predictions produced by such systems, might in fact act on outdated information, as drilled lithology could very well have changed in the last 20-120 minutes.

Mechanical drilling parameters, unlike LWDs, are measured on the rig, and provide information at the bit. These are typically obtained at a sampling period of ~ 2 s. Examples of these are ROP, weight on bit (WOB), drillstring rotation (RPM), and torque (T). These measurements are the earliest indicators of downhole changes. It is known that different lithologies result in different bit-rock interactions [20], meaning that the ROP resulting from a set of mechanical inputs depends on rock type. As RPM is directly controlled by the driller, and autodrillers can be set to maintain constant ROP or WOB, combinations of drilling parameters are influenced by the actions of the driller in addition to lithology. For this reason, mechanical specific energy (MSE) is of interest. MSE defines the input energy necessary to remove a unit volume of rock [31], and removes the driller's actions from the "equation" by relating ROP to input parameters RPM, WOB and T in the following manner:

$$\mathrm{MSE} = \frac{\mathrm{WOB}}{A_b} + \frac{120\pi \cdot \mathrm{RPM} \cdot \mathrm{T}}{A_b \cdot \mathrm{ROP}}, \tag{2.1}$$

where A_b is the bit area (in^2), and other units are WOB (lb), T (lb-ft), and ROP (ft/hr). Still, the drilling parameters are highly dependent on equipment used, well trajectory and uncertain parameters such as bit wear. The fact that no two wells are the same make direct classification of lithology from these measurements difficult for humans. Also a fixed model trained to perform this task will likely not perform well on a wide range of cases due to non-constant dependencies between drilling parameters and lithology.

Estimation of logs and classification of lithology from drilling parameters seems to be a less researched area. There are some examples in literature, such as estimation of gamma ray log [32], and density log [33] from drilling

parameters. However, both of these publications are based on data from a single wellbore, and no indications of the models' ability to generalize to other wellbores are given. Another publication [34] classifies lithology directly from drilling parameters. While this work utilizes data from several different wells, the data is split randomly into training, validation and test sets, meaning that observations from each well are likely in all three data sets. Also, case specific parameters like depth and bit type are used as inputs. If the wells are located at different depths, or use different bits, the model could possibly have learned a mapping from depth or bit type to lithology, and may therefore not be applicable to other wellbores. The ability to give accurate predictions on unseen wellbores is the key criterion speaking to the business value of such models, and the experimental setup is paramount to obtain results that accurately reflect this. To facilitate such results, the test set should contain observations from a separate wellbore, from which no observations are seen during the training and validation process.

Among the several model types previously used to characterize lithology and its properties is the DNN, which has been utilized also for the works presented in this thesis. DNNs are a class of very flexible model architectures that through multiple layers of neurons containing nonlinear activation functions are able to approximate any continuous function to an arbitrary accuracy [35]. The multiple layers allow extraction of latent features from the inputs, and the activation functions allow learning of nonlinear representations. The following notation is used for DNNs:

L : number of layers in the DNN
m_b : size of batch
f : number of features
x : features
y : target variable
s_l : number of neurons in layer $l \in 1, ..., L$
(x_i, y_i) : i-th observation, $i \in 1, ..., m_b$
$\mathbf{w}^{[l]}$: trainable weight matrix for layer l
$\mathbf{b}^{[l]}$: trainable bias vector for layer l

where

$$\mathbf{x} = \begin{bmatrix} \vdots & \vdots & & \vdots \\ x_1 & x_2 & \dots & x_{m_b} \\ \vdots & \vdots & & \vdots \end{bmatrix} \in \mathbb{R}^{f \text{ x } m_b}, \tag{2.2}$$

$$\mathbf{w}^{[l]} = \begin{bmatrix} \dots & w_1^{[l]\top} & \dots \\ \dots & w_2^{[l]\top} & \dots \\ & \vdots & \\ \dots & w_{s_{l-1}}^{[l]\top} & \dots \end{bmatrix} \in \mathbb{R}^{s_{l-1} \text{ x } s_l}, \tag{2.3}$$

$$\mathbf{b}^{[l]} = \begin{bmatrix} b_1^{[l]} \\ b_2^{[l]} \\ \vdots \\ b_{s_l}^{[l]} \end{bmatrix} \in \mathbb{R}^{s_l \text{ x } 1}. \tag{2.4}$$

The learning process of DNNs is referred to as *deep learning* and involves three main steps: *forward propagation*, *computation of loss*, and *backpropagation*. Before the learning process, the weights $\mathbf{w}$ and biases $\mathbf{b}$ of the DNN are randomly initialized to provide a starting point for the optimization. Several initialization schemes are available, though a popular method is the He normal random initialization [36], which pulls weights $\mathbf{w}$ from a truncated

normal distribution with mean $\mu = 0$ and standard deviation $\sigma = \sqrt{\frac{1}{s_{[l-1]}}}$, resulting in the following weight initialization for layer l:

$$\mathbf{w}^{[l]} \in \mathbb{R}^{s_{l-1} \text{ x } s_l} \sim \mathcal{N}\big(\big[0, (s_{[l-1]})^{-1}\big]\big). \tag{2.5}$$

The biases, $\mathbf{b}$, are initialized as zeroes. The aim of this initialization is to avoid vanishing and exploding gradients by managing the variance of the activations in the DNN.

The forward propagation involves a sequence of mathematical operations to propagate the inputs x through the layers of the network to obtain an output $\hat{y}$. During training, mini-batches $(\mathbf{x}_b, \mathbf{y}_b)$ of size m_b are used, for which the forward propagation through each layer l is described by:

$$\mathbf{z}^{[l]} = \mathbf{w}^{[l]\top}\mathbf{a}^{[l-1]} + \mathbf{b}^{[l]}\mathbf{1} \quad l = 1, ..., L, \tag{2.6}$$

$$\mathbf{a}^{[l]} = g^{[l]}(\mathbf{z}^{[l]}) \quad l = 1, ..., L, \tag{2.7}$$

where Equation (2.6) describes the linear component of the forward propagation, starting at $\mathbf{a}^{[0]} = \mathbf{x}_b$, and Equation (2.7) describes the nonlinear component where the linear combinations from the previous layer are passed through each layer's nonlinear activation function $g^{[l]}$. $\mathbf{1} \in \mathbb{R}^{1 \text{ x } m_b}$ is a row vector of ones broadcasting the bias term to each observation of the mini-batch. For layers $l = 1, ..., L-1$, activation functions $g^{[l]}$ can be selected from a range of often used functions, such as sigmoid, tanh, rectified linear units (ReLU) and leaky ReLU. For the output layer L, the activation function is determined by the type of problem being solved. For regression problems, such as LWD estimation, the outputs $\mathbf{y}_b$ are given by:

$$\mathbf{y}_b = \begin{bmatrix} y_1 & y_2 & \dots & y_{m_b} \end{bmatrix} \quad \in \mathbb{R}^{1 \text{ x } m_b}, \tag{2.8}$$

and the output activation is linear, i.e.:

$$\hat{\mathbf{y}}_b = \mathbf{a}^{[L]} = \mathbf{z}^{[L]}, \tag{2.9}$$

whereas in the multiclass classification problem, which is the case for lithology classification, the outputs $\mathbf{y}_b$ are given by:

$$\mathbf{y}_b = \begin{bmatrix} \vdots & \vdots & & \vdots \\ y_1 & y_2 & \dots & y_{m_b} \\ \vdots & \vdots & & \vdots \end{bmatrix} \quad \in \mathbb{R}^{C \text{ x } m_b}, \tag{2.10}$$

where each $y_i \in 1, ..., m_b$ is a one hot vector encoding, and C is the number of classes. The output activation in this case is the softmax:

$$\hat{\mathbf{y}}_b = g^{[L]}(\mathbf{z}^{[L]}) = \frac{e^{\mathbf{z}^{[L]}}}{\sum_{j=1}^{C} e^{\mathbf{z}_j^{[L]}}}, \tag{2.11}$$

Now that the output of the DNN is calculated, it can be compared to the labels $\mathbf{y}_b$ to compute the loss. As for the output activations, the loss is also determined by the type of problem the DNN is aimed at. For the regression problem, the mean squared error is typically used, which is defined by:

$$\mathscr{L}_b = \frac{1}{2m_b} \sum_{i=1}^{m_b} (\hat{\mathbf{y}}_{b,i} - \mathbf{y}_{b,i})^2, \tag{2.12}$$

while for the multiclass classification problem, the categorical cross-entropy is used:

$$\mathscr{L}_b = -\frac{1}{m_b} \sum_{i=1}^{m_b} \sum_{j=1}^{C} \mathbf{y}_{b,j,i} \log \hat{\mathbf{y}}_{b,j,i}. \tag{2.13}$$

Lastly, the gradients of the loss $\mathscr{L}_b$ w.r.t. the trainable weights $\mathbf{w}$ and biases $\mathbf{b}$ can be calculated through backpropagation. This procedure starts by calculating the gradient of the loss w.r.t the output activations, $\frac{\partial \mathscr{L}_b}{\partial \mathbf{a}^{[L]}}$. Furthermore, the gradients backward in the network can be found by the chain rule, so that

$$\frac{\partial \mathscr{L}_b}{\partial \mathbf{z}^{[L]}} = \frac{\partial \mathscr{L}_b}{\partial \mathbf{a}^{[L]}} \frac{\partial \mathbf{a}^{[L]}}{\partial \mathbf{z}^{[L]}}, \tag{2.14}$$

$$\frac{\partial \mathscr{L}_b}{\partial \mathbf{w}^{[L]}} = \frac{\partial \mathscr{L}_b}{\partial \mathbf{a}^{[L]}} \frac{\partial \mathbf{a}^{[L]}}{\partial \mathbf{z}^{[L]}} \frac{\partial \mathbf{z}^{[L]}}{\partial \mathbf{w}^{[L]}}, \tag{2.15}$$

$$\frac{\partial \mathscr{L}_b}{\partial \mathbf{b}^{[L]}} = \frac{\partial \mathscr{L}_b}{\partial \mathbf{a}^{[L]}} \frac{\partial \mathbf{a}^{[L]}}{\partial \mathbf{z}^{[L]}} \frac{\partial \mathbf{z}^{[L]}}{\partial \mathbf{b}^{[L]}}, \tag{2.16}$$

where Equations (2.15) and (2.16) are calculations of the gradients for the weights and biases of the output layer L. This procedure can be further extended throughout layers $L-1, ..., 1$. Finally, the trainable parameters can be updated so that

$$\mathbf{w}^{[l]} = \mathbf{w}^{[l]} - \alpha \frac{\partial \mathscr{L}_b}{\partial \mathbf{w}^{[l]}} \tag{2.17}$$

$$\mathbf{b}^{[l]} = \mathbf{b}^{[l]} - \alpha \frac{\partial \mathscr{L}_b}{\partial \mathbf{b}^{[l]}}, \tag{2.18}$$

where α is the learning rate. When the updates described in Equations (2.17) and (2.18) are used, the procedure is called *gradient descent*, which is the standard (and simplest) parameter update rule. However, several extensions exist [37], including Momentum, RMSProp, Nesterov accelerated gradients, AdaMax, Adagrad, and Adam optimization. Adam optimization [38] is one of the most commonly used optimizers, which adaptively estimates appropriate momentum for the gradient updates. Using Adam optimization, biased first moment estimates $V_{\mathbf{dw}}$ and $V_{\mathbf{db}}$, and biased second raw moment estimates, $S_{\mathbf{dw}}$ and $S_{\mathbf{db}}$ are initialized as 0 before optimization begins. Exponential decay rates for the first and second moment estimates, $\beta_1 = 0.9$ and $\beta_2 = 0.999$, along with a small number to avoid division with zero, $\varepsilon = 10^{-8}$ are also initialized, where the presented values for these are default values suggested by the authors of the Adam paper. On every iteration i, after the gradients of the weights and biases have been calculated, the following equations describe the gradient update steps:

$$V_{\mathbf{dw}} = \beta_1 V_{\mathbf{dw}} + (1 - \beta_1)\frac{\partial \mathscr{L}_b}{\partial \mathbf{w}}, \tag{2.19}$$

$$V_{\mathbf{db}} = \beta_1 V_{\mathbf{db}} + (1 - \beta_1)\frac{\partial \mathscr{L}_b}{\partial \mathbf{b}}, \tag{2.20}$$

$$S_{\mathbf{dw}} = \beta_2 S_{\mathbf{dw}} + (1 - \beta_2)\frac{\partial \mathscr{L}_b}{\partial \mathbf{w}}^2, \tag{2.21}$$

$$S_{\mathbf{db}} = \beta_2 S_{\mathbf{db}} + (1 - \beta_2)\frac{\partial \mathscr{L}_b}{\partial \mathbf{b}}^2, \tag{2.22}$$

$$V^c_{\mathbf{dw}} = \frac{V_{\mathbf{dw}}}{1 - \beta_1^i}, \tag{2.23}$$

$$V^c_{\mathbf{db}} = \frac{V_{\mathbf{db}}}{1 - \beta_1^i}, \tag{2.24}$$

$$S^c_{\mathbf{dw}} = \frac{S_{\mathbf{dw}}}{1 - \beta_2^i}, \tag{2.25}$$

$$S^c_{\mathbf{db}} = \frac{S_{\mathbf{db}}}{1 - \beta_2^i}, \tag{2.26}$$

$$\mathbf{w} = \mathbf{w} - \alpha\frac{V^c_{\mathbf{dw}}}{\sqrt{S^c_{\mathbf{dw}}} + \varepsilon}, \tag{2.27}$$

$$\mathbf{b} = \mathbf{b} - \alpha\frac{V^c_{\mathbf{db}}}{\sqrt{S^c_{\mathbf{db}}} + \varepsilon}, \tag{2.28}$$

where superscript c denotes the bias-corrected counterparts of the moment estimates.

In a supervised learning setting, a model is typically trained and validated in order to converge towards a model that performs well on a test set. The labels of the training data are used to supervise the updates made to the model, while the labels of the validation set are used by the developer in the tuning process. Finally, the labels of the test set, previously unseen by the model, are used to give an unbiased estimate of model performance. Test set performance is the most important metric to evaluate a model, as it speaks to the *generalizability* of the model, i.e. if the representations learned during training are transferable to unseen observations. This property relies heavily on the assumption that the unseen observations that the model is tested on, come from the same distribution as the training data, which is an assumption that often fails for real-world problems. Generally,

when shifts in independent or dependent variables occur, or the underlying process that links the variables evolves, it is called *nonstationarity*. This phenomenon can occur for a variety of reasons, such as seasonality, aging effects in sensors, or thermal drifts, all of which are harmful to the predictive power of a data-driven model [39]. This fact, along with the growing availability of labeled data, has motivated the paradigm of streaming learning, where a model learns from a data stream in an online fashion. This allows the model to adapt to new available data. However, such a learning scheme introduces new problems. One example is that of retaining existing, still relevant knowledge, while at the same time being capable of learning new relationships. These two objectives could very well be conflicting, giving rise to the *stability-plasticity* dilemma [40]. If no measures are taken to retain old knowledge, adaption to new data can lead to *catastrophic forgetting* [41] of old knowledge, which is a common problem in streaming learning.

Multiple previous publications have aimed to handle catastrophic forgetting in different ways. One method is through *regularization*, which aims to protect weights important for retaining previous knowledge. This can be done in several ways, such as selective constraining of certain weights [42], or configurations with both shared and task-specific layers, in which shared layers are trained on several tasks [43], which has a regularizing effect. Ensemble methods [44, 45] are also used to overcome catastrophic forgetting, such as the Learn^{++}.NSE algorithm [40], which trains a new model for each batch of data it receives, and combines these using a dynamically weighted majority voting. Different *experience replays* also exist, which play a role in deciding which examples to retain or train on. The prioritized experience replay [46] selects observations for training based on model error, while the ExStream algorithm [47] is a stream clustering method to retain valuable information in the replay buffers. There are several ways to characterize algorithms designed to handle concept drift, one of which is to separate them into active and passive approaches. Active approaches include a drift detection mechanism, which leads to learning only when drift is detected. This mechanism aims to identify the occurrence and severity of the drift, which is exposed to the risk of imperfect detection, leading to false reports. Passive methods on the other hand, make no assumption of the drift, acknowledging that drift may occur at any time. This leads to a continuous learning as new data is presented to the model [40].

2.2 Managed pressure drilling

During drilling, the annular pressure in the wellbore should be controlled to be within pressure margins, i.e. between the pore pressure and fracture pressure of the surrounding formation. Pressures below pore pressure lead to influx of formation fluid, also called a kick, which if not handled correctly, could lead to a blowout. Blowouts are severe incidents which could lead to loss of lives, damage to equipment, and significant environmental damage. One well-known example is the Deepwater Horizon accident[1]. Pressures exceeding the fracture pressure, on the other hand, will lead to fracturing in the formation, and subsequently loss of drilling fluid. This in itself is expensive, due to necessary replacement of drilling fluid, but also because loss of circulation fluid, as mentioned previously, could lead to pipe sticking.

Several measures are available to avoid these scenarios. Casings can be cemented throughout the wellbore to isolate various zones which may have different pore and fracture pressures. However, the lower part of the wellbore is open to the surrounding formation, and in these areas, annular pressure must be accurately controlled. One way is by changing the density of the drilling fluid, which is a slow process, and not well suited to handle sudden events. Managed pressure drilling (MPD) is more suitable in this respect, and is a technology commonly employed for more accurate pressure control, which is important especially in areas with tight pressure margins. Different versions of MPD exist, although a common configuration is to close off the annulus topside, and restricting the returning drilling fluid through a topside choke valve, which controls the annular pressure. Also, a backpressure pump is used to ensure flow through the choke valve, and maintain pressure in the event of loss of fluids. The schematics for this configuration is illustrated in Figure 2.2.

Traditionally, managed pressure systems were operated manually by the crew, until this process became automated. Currently, PID control is widely accepted as the standard common practice for bottomhole pressure control, although research has been conducted on alternatives, such as nonlinear model predictive control (NMPC) [48]. Efforts have also been made to de-

[1]National Geographic article available online at https://www.nationalgeographic.com/science/article/bp-oil-spill-still-dont-know-effects-decade-later (accessed July 15, 2021)

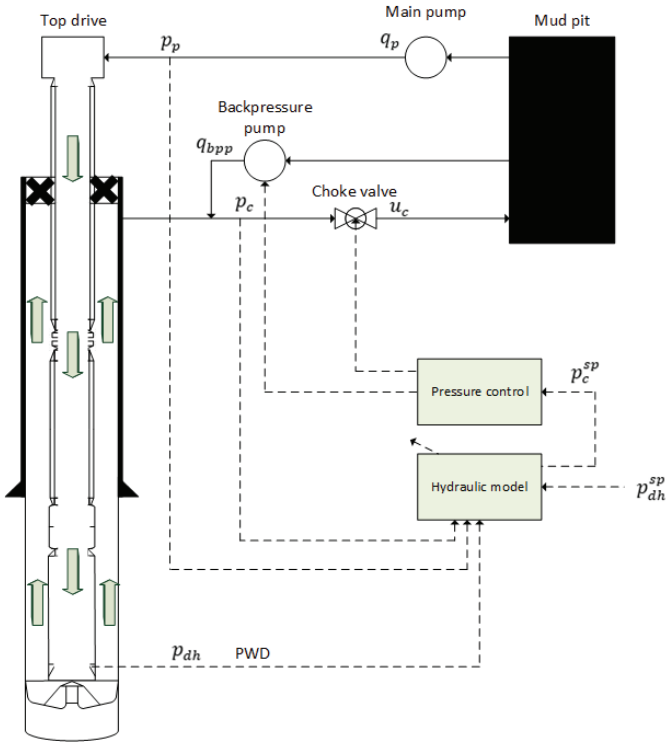

Figure 2.2: Schematics of a managed pressure system using a topside choke valve and backpressure pump.

velop hydraulics models to estimate the bottomhole pressure, on which the choke pressure controller relies. Both advanced [49] and simplified [50] models have been developed from these endeavours.

In this thesis, the utilization of deep Q learning is explored for bottomhole pressure control during pipe connection. Deep Q learning is a model-free, off-policy temporal difference learning algorithm in which an agent through interaction with an environment learns an optimal behavioral policy in order to maximize reward. At time step t, the agent starts from state s_t, interacts with the environment by performing an action a_t, receives a scalar reward r_t, and observes the subsequent state s_{t+1}. The reward is an essential parameter in reinforcement learning, as it is the metric the agent uses to evaluate its performance. This is provided by a reward function, which is

designed by the developer. Reward function design is a determining factor for the behavior of the agent, and rewards can be both positive to reinforce wanted behavior, or negative (penalty) to discourage unwanted behavior. In a process control setting, the reward function typically penalizes deviation from setpoint. To avoid excessive actuation, a penalty on controller output can also be added. This kind of formulation is common within the field of optimal control, and is standard for controllers such as the linear quadratic regulator (LQR) and MPC.

In reinforcement learning it is assumed that the environment is a Markov Decision Process (MDP), meaning that the process is defined by its state-action pair, and the one-step dynamics of the environment [51]. In this context, model-free refers to the fact that the agent does not rely on any prior knowledge of how the environment will change in response to an action. In other words, the agent starts with a clean slate, or *tabula rasa*. The goal is to estimate the optimal action value function $Q^*(s, a)$, which represents the expected return of starting in state s, taking action a, and then following an optimal behavior policy. The optimal action value function is defined as

$$Q^*(s, a) = \max_\pi \mathbb{E}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ...|s_t = s, a_t = a, \pi], \tag{2.29}$$

which is the maximum expected discounted cumulative reward. Here, γ is the discount factor which determines the present value of future rewards. This parameter can be tuned in order to change the "patience" of the agent, i.e. whether to prefer short-term or long-term reward. The optimal action value function obeys the Bellman equation, which allows a re-phrasing of Equation (2.29). Bellman's principle of optimality states that "an optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision" [52]. Here, the intuition is to consider the first decision first, and set aside all future decisions, so that

$$Q^*(s, a) = \mathbb{E}_{s'}[r + \gamma \max_{a'} Q^*(s', a')|s, a], \tag{2.30}$$

where it can be seen that the optimal action value is the sum of the reward obtained in the current step, r, and the discounted sum of all future rewards.

To find the optimal policy π^*, the action maximizing the Q function can be selected at every time step. This can be expressed as

$$\pi^*(s) = \underset{a}{\operatorname{argmax}}\, Q^*(s,a). \tag{2.31}$$

This policy is *greedy*, as it always *exploits* current knowledge to maximize reward. During training of the agent, such a policy is often undesirable, as it never *explores* alternative actions. As the goal of the agent is to maximize cumulative reward, exploration of alternatives is of importance, since they may in fact lead to higher reward in the long run. The typical way to balance exploration and exploitation is to employ an $\varepsilon-$greedy policy, which behaves greedily most of the time, but with a small probability ε selects an action at uniform random. This exploration rate can either be constant or exponentially decaying.

In deep Q learning, the Q function is approximated by a DNN, often referred to as the Q network, which is parameterized by weights Ψ, denoted $Q(s,a;\Psi)$, and the goal is to converge to the optimal action value function through iterative updates, so that $Q(s,a;\Psi) \approx Q^*(s,a)$. The optimal target values $r + \gamma \max_{a'} Q^*(s',a')$ are approximated by a *target* network, $r + \gamma \max_{a'} Q^*(s',a';\Psi_{targ})$. The use of a separate target network is a measure to avoid unstable optimization due to correlations between action values Q and target values $r + \gamma \max_{a'} Q(s',a')$. The iterative update of the Q function when using these two networks becomes

$$Q(s,a;\Psi) \leftarrow Q(s,a;\Psi) + \alpha \overbrace{[\underbrace{r + \gamma \max_{a'} Q(s',a';\Psi_{targ})}_{\text{Target}} - \underbrace{Q(s,a;\Psi)}_{\text{Current estimate}}]}^{\text{Temporal difference}}, \tag{2.32}$$

where α is the learning rate. From Equation (2.32), it can be seen that the Q function is updated by use of the estimated Q value for the next state s' and the *greedy* action a', rather than an estimate following the behavior policy. This is what makes Q learning an off-policy method. To avoid instability due to correlations in sequences of observations, an experience replay is used. The experience replay is typically a fixed-size window, where the agent's experiences at each time step, $e_t = [s_t, a_t, r_t, s_{t+1}]$, are stored. Mini-batches

are sampled for training either at a uniform random distribution, or by prioritization [46]. One last measure is taken to avoid instability. To avoid "chasing moving targets", a slow-updating mechanism for the parameters of the target network, Ψ_{targ} is implemented. In the original DQN algorithm [9], these parameters are updated every C iterations to be equal to those of the Q network, Ψ, and are held constant in between. However, gradual updates, such as

$$\Psi_{targ} \leftarrow \phi\Psi + (1-\phi)\Psi_{targ}, \tag{2.33}$$

have been utilized [10] as well. In this update rule, ϕ is a very small number between 0 and 1.

2.3 Overview

In the previous sections, two focus areas within the drilling process have been presented, along with machine learning methodology to automate them. At-bit LWD estimation and lithology classification are helpful to the driller to allow more informed decision making at an earlier stage, which is a central focus of this thesis. While this is a pure data analysis task, such systems can be further built upon to facilitate higher autonomy modes in the future. Examples of added features could be alarms for changing downhole conditions, suggestions for suitable actions and adjustments to drilling parameters, or even fully autonomous setpoint changes. Through incremental increases in autonomy level along with field trials at each stage, confidence in such a system can be built over time. Bottomhole pressure control can be achieved by a reinforcement learning agent, which, if trained on enough scenarios, could be applicable to a wider range of cases, such as varying well depths, drilling fluids, flow rates and friction effects (such as during pipe connection), thus requiring less tuning. Schematics of a possible future drilling rig including these autonomous subsystems are given in Figure 2.3.

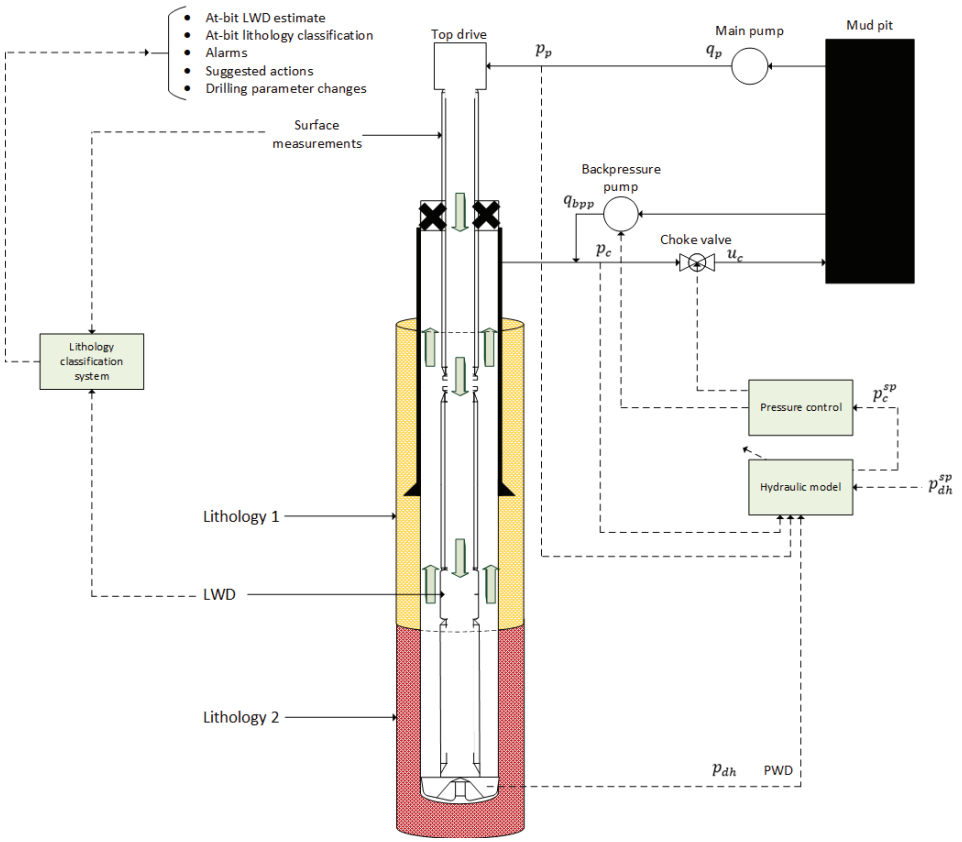

Figure 2.3: Schematics of possible future autonomous subsystems on a drilling rig enabled by machine learning.

Chapter 3

Paper A: At-bit estimation of rock density from real-time drilling data using deep learning with online calibration

Arnø ML, Godhavn J-M, Aamo OM. At-bit estimation of rock density from real-time drilling data using deep learning with online calibration. In *Journal of Petroleum Science and Engineering*, page 1009006, 2021 https://doi.org/10.1016/j.petrol.2021.109006

At-Bit Estimation of Rock Density from Real-Time Drilling Data Using Deep Learning with Online Calibration

Mikkel Leite Arnø*, Department of Engineering Cybernetics, Norwegian University of Science and Technology, 7491 Trondheim, Norway; **John-Morten Godhavn,** Equinor Research Center, 7053 Ranheim, Norway; and **Ole Morten Aamo,** Department of Engineering Cybernetics, Norwegian University of Science and Technology, 7491 Trondheim, Norway

*Corresponding author; email: mikkel.l.arno@ntnu.no

Abstract

We present a novel streaming learning approach, utilizing a deep neural network (DNN) to learn from data available during operation to estimate at-bit density using drilling parameters. Since every wellbore is different, the relationship between drilling parameters and at-bit density varies. Equipment used, well trajectory, friction and bit wear are examples of conditions that affect this relationship and makes a pre-trained model unable to represent an accurate input/output mapping applicable to all wells. However, using delayed density log measurements, continuously supervising updates to the model is possible during operation. The algorithm has been tested on drilling data from wells on a field operated by Equinor and compared to a standard deep learning approach, where results show that a streaming learning approach outperforms the traditional method. Statistical analyses have been performed to verify the statistical significance and effect size on the data sets. Data visualizations using a t-distributed Stochastic Neighbor Embedding (t-SNE) indicate that the relationship between drilling parameters and density log indeed vary between wellbores, making generalizability an issue for a traditional supervised learning approach to this problem, and motivating a streaming learning approach. Using the proposed method, more accurate at-bit estimates can be made, providing preliminary indications ahead of the tool placed 20-30 m behind the bit, which, dependent on rate of penetration

(ROP), will be available 20 - 120 minutes later.

3.1 Introduction

The drilling operation is complex. It is also subject to significant uncertainty, which needs to be managed in order to ensure a safe and efficient drilling operation. One such source of uncertainty is at-bit lithology, which is a central part of the environment the drilling system is in interaction with. Lithology evaluation is relevant for best-practice selection of suitable drilling parameters, evaluation of reservoir structure and evaluation of well placement, to mention a few. Logging while drilling (LWD) tools [22] are typically used by experts to classify lithology. However, they are placed some distance behind the bit, making at-bit lithology evaluation directly from LWD tools impossible. The density log is an LWD tool that can be used to separate harder lithologies such as stringers from softer lithologies. Although not a conclusive lithology indicator on its own, it provides valuable information on downhole conditions, and in combination with other LWD logs an accurate understanding of downhole lithology can be achieved. This tool is typically mounted 20-30 m behind the bit. Due to the placement of these tools, drilling parameters are the earliest indicators for changes in at-bit lithology, although they are difficult for humans to interpret manually. Deep learning specializes in finding patterns in data, and can be used to find a mapping from drilling parameters to density log, although every wellbore is different with case-specific conditions such as equipment used, well trajectory, friction and bit wear. Bit wear, for example, is very difficult to estimate due to vast uncertainty. During drilling, as the bit is gradually worn, the bit-rock interaction will change [53]. A static model might misinterpret the dulled bit as a harder lithology, and thus overestimate bulk density.

LWD tools and deep neural networks (DNNs) have been combined before in [23], where a DNN takes as input a set of LWD logs and outputs an estimate for another logging tool. They present the results for three different input/output combinations, in which the estimated logs are: resistivity, density and neutron. As LWD tools are expensive, companies do not always use all of them, resulting in a less complete image of lithology properties, and [23] was motivated by reducing these costs, and

to estimate logs when missing. In [24] a long short-term memory (LSTM) model is used for virtual log generation. They present an experiment where the neutron porosity, delta-time shear, and array induction two-foot resistivity are estimated based on gamma ray and delta-time compressional. Results include comparisons between the LSTM and DNN, where they found the LSTM to be superior for their study. Another study [32] simulates missing data in a wellbore, and presents results for several machine learning algorithms attempting to estimate gamma ray log from drilling parameters. In [25], a conditional variational autoencoder (CVAE) is used to estimate shear-slowness (DTS) from other logs such as gamma ray, neutron porosity, bulk density and compressional-slowness (DTC). Their results are compared to that of an LSTM and a bi-directional LSTM. [33] presents experiments using both a DNN and an adaptive network-based fuzzy inference system (ANFIS) for estimation of density log using drilling parameters as input. The models in this study are trained and tested on 2400 observations from the same well, where missing data is simulated for a section of the well, on which the trained models perform with high accuracy. If a well has available log data for only certain sections, these standard deep learning methods could be applied to fill in sections of missing log data. A novel Bayesian neural network approach, using neutron porosity, gamma ray, deep resistivity, photoelectric factor and density logs to estimate sonic log, is presented in [26]. Their results are comparable to that of traditional neural networks, and in addition offers quantification of uncertainty in the predictions. Lithology classification based on LWD logs using data-driven methods has been presented in several previous publications. Examples of methods include scalable gradient boosted decision trees [27], support vector machines (SVMs) [28] and bidirectional gated recurrent units (GRUs) [29], [30]. Evaluation studies [54] have also been presented. Other downhole characteristics have also been addressed using machine learning. In [55], a streaming learning system for inclination prediction in directional drilling using a GRU is presented. This work is motivated by avoiding delayed corrective actions, thus improving well placement. It is also argued that standard machine learning approaches without online retraining fail to generalize well to different wellbores. In [56], a Bayesian approach is applied to fault detection using a convolutional neural network (CNN), allowing risk evaluation through uncertainty quantification.

The contribution of this work is two-fold. First, we present a novel streaming learning approach, using a DNN to solve the problem of estimating at-bit density log from drilling parameters. A pre-trained model continuously learns from the data stream available during operation, allowing the model to adapt to case specific conditions. The method is applied to data from several wellbores on a field operated by Equinor to demonstrate performance, and compared to the performance of a baseline model, which represents the traditional approach to log estimation. Next, an unsupervised learning data analysis is performed, which visualizes how data from different wellbores are structured differently. This analysis indicates the weakness in using a pre-trained and static model to estimate at-bit density using drilling parameters, motivating the streaming learning approach.

The paper is structured as follows: Section 3.2 presents the data used, along with the different methods and algorithms used in this study, including our own n-bin experience replay buffer. Section 3.3 presents results for data visualizations and at-bit density estimates versus measurements. Lastly, section 3.4 offers conclusions and suggestions for further work.

3.2 Data and methodology

3.2.1 Data

Data from the reservoir sections of 9 different wells on a homogeneous field operated by Equinor were gathered for this work, for a total of 1.75 million observations. The training set contained 5 wells, for a total of approximately 740 000 observations, and the validation set consisted of 1 well with approximately 365 000 observations. The test set contained 3 wells, with a total of 645 000 observations. Data cleaning was performed by visual inspection. Obviously erroneous measurements were removed by logic specifying reasonable values. Next, data was centered and scaled to have zero mean and unit variance before training commenced.

3.2.2 Measurements description

The drilling system is composed of several subsystems, where data acquisition is different for different subsystems. For the purpose of this

work, we divide measurements into two groups: *surface measurements* and *downhole measurements*. Surface measurements are taken on the rig, and examples are drilling parameters like hook load (HKLD), surface torque (T), surface drillstring rotation (RPM), flow (Q) and hook height. Weight on bit (WOB) is closely related to, and derived from HKLD, while bit depth, hole depth and rate of penetration (ROP) are derived from hook height. In this work, downhole measurements refers to the density log. The density logging tool measures density of lithology perpendicular to the drill string by emitting gamma rays and detecting backscatter, giving a measure of average electron density in the lithology, which is strongly correlated with bulk density. In other words, it gives an indirect measurement of the bulk density. These measurements are communicated to the rig by mudpulse telemetry.

It is established [20] that lithologies with different properties yield different bit-rock interactions, meaning that ROP is dependent on rock properties and input actuation like WOB, T and RPM. This means that information regarding at-bit conditions like density should be latently available in real-time through these surface measurements. The density logging tool on the other hand is installed in the bottomhole assembly (BHA), 20 - 30 m behind the bit. Dependent on ROP, this amounts to significant time delays on logs in ranges of typically 20 - 120 minutes. To eliminate this delay, we propose to estimate the at-bit density from surface measurements.

3.2.3 Depth correction & resampling

Surface measurements are sampled with a sampling period of typically 2-3 s, while the density log is sampled at a lower rate, typically every 10-15 s. Also, surface measurements and downhole measurements recorded at a given time provide information at different depths. For these reasons, a method to correct for depth and differing sampling rates was required. This was done using a *holding buffer* that stored incomplete observations. As Figure 3.1 illustrates, an observation can be completed once the distance between bit and tool has been drilled, and the label corresponding to some set of surface measurements is obtained. We start by denoting the sampling periods for surface measurements x, and the density log y, as T_s and T_d, respectively. The i-th surface measurements are taken at time iT_s, so that $x_i = x(iT_s)$. Similarly, the i-th density log measurement is taken at

time iT_d, so that $y_i = y(iT_s)$. x_i provides information at the bit, at depth $d_{s,i} = d_s(iT_s)$, while y_i, measured at a known distance behind the bit, d_{tool}, provides information at depth $d_{d,i} = d_s(iT_d) - d_{tool}$. Once we obtain the first density log measurement y_k, where $d_{d,k} > d_{s,j}$ for any indices j in the holding buffer, we can pair observations so that each x_j in the holding buffer is paired with the previous density log measurement, y_{k-1}, where $d_{d,k-1} \leq d_{s,j}$. In addition to correcting for depth, this procedure is equivalent to resampling the y's using the forward fill method.

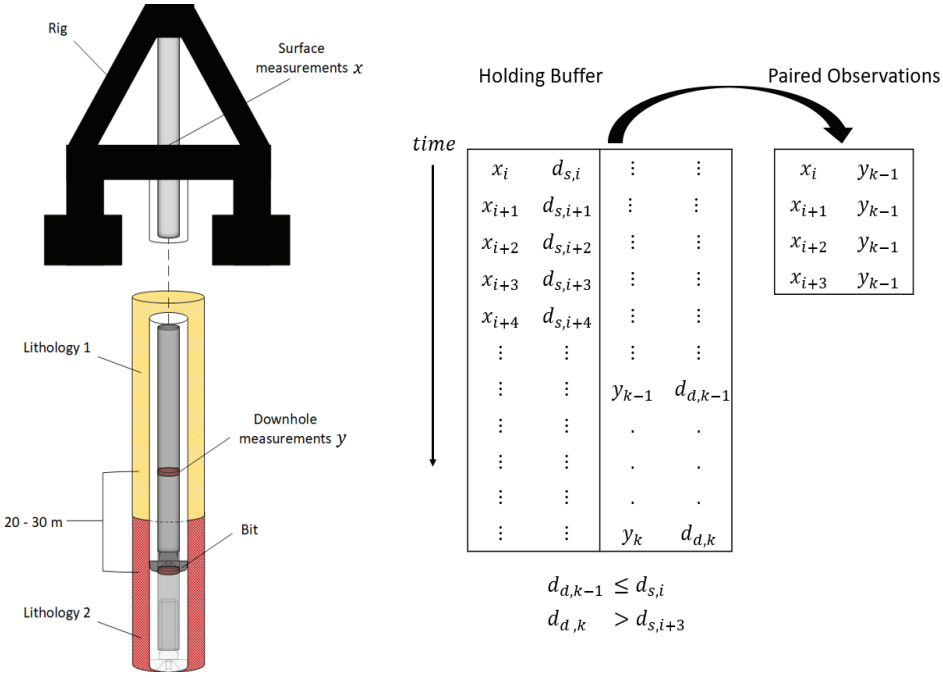

Figure 3.1: Schematics illustrating the drilling operation and the availability of measurements.

3.2.4 Variable selection

The variables used as input to the DNN were selected based on domain knowledge and availability. To eliminate the density log delay, we limit ourselves to measurements providing at-bit information, which eliminates other LWD tools. In addition to using drilling parameters, we can perform some feature engineering. Mechanical specific energy (MSE) quantifies the

amount of energy required to remove a unit volume of rock. It is a function of ROP, WOB, T and RPM, and known to be different for different lithologies [31]. MSE is given by:

$$\text{MSE} = \frac{\text{WOB}}{A_b} + \frac{120\pi \cdot \text{RPM} \cdot \text{T}}{A_b \cdot \text{ROP}}, \tag{3.1}$$

where units are WOB (lb), T (lb-ft), ROP (ft/hr), and A_b is bit area (in^2). Another parameter of interest is the hydro-mechanical specific energy (HMSE), which also accounts for the weakening of the rock ahead of the bit due to flow. HMSE is given by:

$$\text{HMSE} = \frac{\text{WOB}}{A_b} + \frac{120\pi \cdot \text{RPM} \cdot \text{T}}{A_b \cdot \text{ROP}} + \frac{1154\eta \cdot \Delta P_b \cdot \text{Q}}{A_b \cdot \text{ROP}}, \tag{3.2}$$

where η is the hydraulic energy reduction factor, ΔP_b is the bit pressure drop at the nozzle (psi), and Q is the flow rate (gpm). Some parameters related to the bit and mud were not available to compute the hydraulic contribution of HMSE [32]. Default parameter values were set for these, as described in Table 3.1.

Table 3.1: Default values set for unknown parameters.

Parameter	**Default value**
Bit type	Polycrystalline Diamond Compact (PDC)
Junk slot area	14 (in^2)
Flow area	0.12 (in^2)
Mud weight	20 (ppg)

From the default parameters, we can compute HMSE, and further define the input vector of predictors for the DNN, $\mathbf{x}$, as:

$$\mathbf{x} = [\text{ROP}, \quad \text{RPM}, \quad \text{WOB}, \quad \text{T}, \quad \text{MSE}, \quad \text{HMSE}]^\top. \tag{3.3}$$

The output of the DNN, $\hat{\mathbf{y}}$, is simply at-bit density. As a sanity check, we wish to investigate the correlations between the density log and the inputs to the model, which are illustrated in Figure 3.2. The correlations are presented separately for each wellbore, and it can be seen that the strength of the linear relationships varies between wellbores. Correlations between density and ROP are the most consistent. MSE and HMSE are also quite

strongly correlated with density for most wellbores, although the strength varies more. Especially for training wells 3 and 5, these relationships are weaker. RPM, WOB and T correlations vary more, although for some wells, these can be seen to be strongly correlated with density. Several of the drilling parameters are controlled by the driller. Autodrillers can be set to maintain constant WOB or ROP. When the driller suspects that a stringer is being drilled, the WOB is routinely increased while RPM is reduced. This will typically lead to a reduction in T as well. For training wells 2 - 5, the correlations for RPM, WOB and T in Figure 3.2 support this. For training well 1, however, the signs of the correlations for these parameters are inverted. For the validation well, the sign of the correlation for T is inverted. Since these parameters are controlled by the driller, correlations with density will be highly affected by the driller's choices. As an example, if the drilling strategy is to increase both WOB and RPM for stringers, the resulting ROP might increase as well. Isolated, increased ROP for stringers might seem counterintuitive, but would be explained by the overall drilling strategy and the actions performed by the driller. MSE and HMSE on the other hand, eliminate the driller's actions, and define the input energy required to drill through the rock. The correlations between density and these parameters are consistently positive, indicating that more energy is required to drill denser lithology.

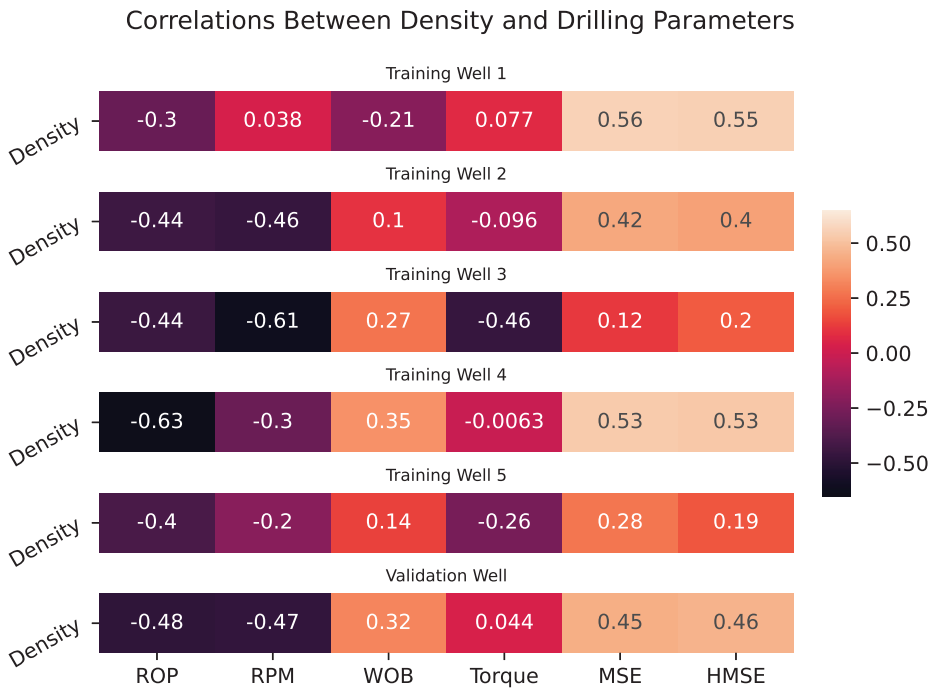

Figure 3.2: Correlations between density log and drilling parameters for training wells and validation well.

3.2.5 Deep neural networks

The specifics of the DNN used in this work are outlined in this section. First, we provide notation for the DNN:

$$
\begin{aligned}
L &: \text{number of layers in the DNN} \\
m &: \text{number of observations} \\
f &: \text{number of features} \\
x &: \text{drilling parameters / features} \\
y &: \text{density log / target variable} \\
s_l &: \text{number of neurons in layer } l \in 1, ...L \\
(x_i, y_i) &: i\text{-th training example, } i \in 1, ...m \\
\mathbf{w}^{[l]} &: \text{trainable weight matrix for layer } l \\
\mathbf{b}^{[l]} &: \text{trainable bias vector for layer } l
\end{aligned}
$$

and

$$
\mathbf{x} = \begin{bmatrix} \vdots & \vdots & & \vdots \\ x_1 & x_2 & \dots & x_m \\ \vdots & \vdots & & \vdots \end{bmatrix} \quad \in \mathbb{R}^{f \text{ x } m}, \tag{3.4}
$$

$$
\mathbf{y} = \begin{bmatrix} y_1 & y_2 & \dots & y_m \end{bmatrix} \quad \in \mathbb{R}^{1 \text{ x } m}, \tag{3.5}
$$

$$
\mathbf{w}^{[l]} = \begin{bmatrix} \dots & w_1^{[l]\top} & \dots \\ \dots & w_2^{[l]\top} & \dots \\ & \vdots & \\ \dots & w_{s_{l-1}}^{[l]\top} & \dots \end{bmatrix} \quad \in \mathbb{R}^{s_{l-1} \text{ x } s_l}, \tag{3.6}
$$

$$
\mathbf{b}^{[l]} = \begin{bmatrix} b_1^{[l]} \\ b_2^{[l]} \\ \vdots \\ b_{s_l}^{[l]} \end{bmatrix} \quad \in \mathbb{R}^{s_l \text{ x } 1}. \tag{3.7}
$$

For a given layer l, $\mathbf{z}^{[l]}$ denotes the linear combination of activations from the previous layer, $\mathbf{a}^{[l-1]}$, determined by the trainable weights $\mathbf{w}^{[l]}$ and biases $\mathbf{b}^{[l]}$:

$$\mathbf{z}^{[l]} = \mathbf{w}^{[l]\top}\mathbf{a}^{[l-1]} + \mathbf{b}^{[l]}\mathbf{1}, \tag{3.8}$$

where $\mathbf{1}$ is a row vector of ones. Equation (3.8) describes the linear component of the forward propagation. Next, $\mathbf{z}^{[l]}$ is passed through the layer activation function $g^{[l]}$, which is the nonlinear component of the forward propagation:

$$\mathbf{a}^{[l]} = g^{[l]}(\mathbf{z}^{[l]}). \tag{3.9}$$

The leaky ReLU activation function is utilized for every hidden layer, so that

$$\mathbf{a}^{[l]} = \max\{\gamma\mathbf{z}^{[l]}, \mathbf{z}^{[l]}\}, \quad l = 1, ..., L-1. \tag{3.10}$$

γ is the slope in the left half plane. Note that $\mathbf{a}^{[0]} = \mathbf{x}$. Finally, the DNN outputs the estimated density at-bit, which is a quantitative output, resulting in a regression layer:

$$\hat{\mathbf{y}} = \mathbf{a}^{[L]} = \mathbf{z}^{[L]}. \tag{3.11}$$

Equations (3.8) - (3.11) describe the forward propagation of the DNN. The model can be visualized by a layered model with connected nodes, as shown in Figure 3.3. The weight initialization is He normal [36], which mitigates exploding and vanishing gradients by managing the variance of the activations throughout the layers of the network. This is done by pulling the weights in each layer from a truncated normal distribution with mean $\mu = 0$ and standard deviation $\sigma = \sqrt{\frac{1}{s_{[l-1]}}}$. The weight initialization for layer l is then given by:

$$\mathbf{w}^{[l]} \in \mathbb{R}^{s_{l-1} \text{ x } s_l} \sim \mathcal{N}\left(\left[0, (s_{[l-1]})^{-1}\right]\right). \tag{3.12}$$

The biases, $\mathbf{b}^{[l]}$, are initialized as zeros. The optimizer used was Adam optimization [38], which adaptively estimates appropriate momentum for the gradient updates. The Adam optimization algorithm is given by **Algorithm 1**.

Algorithm 1 Adam Optimization

$V_{\mathbf{dw}} = 0, S_{\mathbf{dw}} = 0, V_{\mathbf{db}} = 0, S_{\mathbf{db}} = 0,$
for each sampled mini-batch **do**
  Forward prop on $\mathbf{x}$:
    $\mathbf{z}^{[1]} = \mathbf{w}^{[1]\top}\mathbf{x} + \mathbf{b}^{[1]}\mathbf{1}$
    $\mathbf{a}^{[1]} = g^{[1]}(\mathbf{z}^{[1]})$
    $\mathbf{z}^{[2]} = \mathbf{w}^{[2]\top}\mathbf{a}^{[1]} + \mathbf{b}^{[2]}\mathbf{1}$
    $\mathbf{a}^{[2]} = g^{[2]}(\mathbf{z}^{[2]})$
    $\vdots$
    $\mathbf{a}^{[L]} = \mathbf{z}^{[L]}$
    $\hat{\mathbf{y}} = \mathbf{a}^{[L]}$
  Compute cost J
  Backpropagate using the chain rule to compute gradients $\mathbf{dw}$ and $\mathbf{db}$
  $V_{\mathbf{dw}} = \beta_1 V_{\mathbf{dw}} + (1-\beta_1)\mathbf{dw}$
  $V_{\mathbf{db}} = \beta_1 V_{\mathbf{db}} + (1-\beta_1)\mathbf{db}$
  $S_{\mathbf{dw}} = \beta_2 S_{\mathbf{dw}} + (1-\beta_2)\mathbf{dw}^2$
  $S_{\mathbf{db}} = \beta_2 S_{\mathbf{db}} + (1-\beta_2)\mathbf{db}^2$
  $V^c_{\mathbf{dw}} = \frac{V^c_{\mathbf{dw}}}{1-\beta_1^r}$
  $V^c_{\mathbf{db}} = \frac{V^c_{\mathbf{db}}}{1-\beta_1^r}$
  $S^c_{\mathbf{dw}} = \frac{S^c_{\mathbf{dw}}}{1-\beta_2^r}$
  $S^c_{\mathbf{db}} = \frac{S^c_{\mathbf{db}}}{1-\beta_2^r}$
  $\mathbf{w} = \mathbf{w} - \alpha\frac{V^c_{\mathbf{dw}}}{\sqrt{S^c_{\mathbf{dw}}}+\varepsilon}$
  $\mathbf{b} = \mathbf{b} - \alpha\frac{V^c_{\mathbf{db}}}{\sqrt{S^c_{\mathbf{db}}}+\varepsilon}$
end for

β_1, β_2 and ε are tunable hyperparameters for Adam optimization, and r is the current iteration number. $V_{\mathbf{dw}}$ and $V_{\mathbf{db}}$ are biased first moment estimates. $S_{\mathbf{dw}}$ and $S_{\mathbf{db}}$ are biased second raw moment estimates. Superscript c denotes their bias-corrected counterparts. $\mathbf{dw}$ and $\mathbf{db}$ are $\frac{\partial J}{\partial \mathbf{w}}$ and $\frac{\partial J}{\partial \mathbf{b}}$, respectively. α is the learning rate. Cost J is defined as

$$J = \frac{1}{2m_b}\left|\hat{\mathbf{y}} - \mathbf{y}\right|^2, \tag{3.13}$$

where m_b denotes the number of observations in a mini-batch.

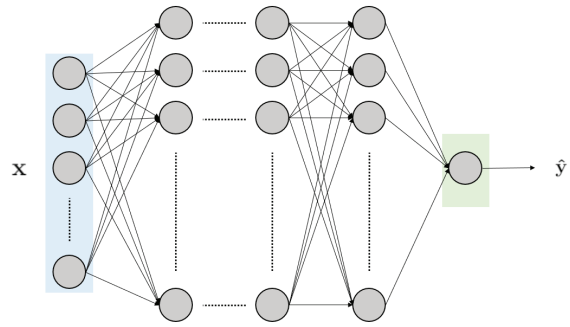

Figure 3.3: Deep neural network. **Input: x**. **Output: ŷ**.

3.2.6 n-bin prioritized experience replay

In conventional supervised learning, a model is typically trained on a fixed dataset for multiple passes, aiming to converge towards a well-performing model for unseen data assumed to come from the same distribution. However, this simple assumption breaks down in many cases due to a variety of factors, such as shift in the independent or dependent variables, or due to an evolving underlying process. This phenomenon is commonly known as nonstationarity or concept drift, and is harmful to the predictive power of such models [39], [40]. The aim of streaming learning is to continuously update the model to correct for these effects. However, a common problem in streaming learning is catastrophic forgetting [41], where old representations are forgotten due to adaptation to the non-stationary environment. In our attempt to adapt to drifting concepts while mitigating catastrophic forgetting, an n-bin prioritized experience replay $\mathscr{D} \in \mathbb{R}^{n \times N}$ buffer was developed. The prediction space $y \in [y_{min}, y_{max}]$ is divided into n bins, and each bin in the buffer contains N observations. This configuration ensures that the replay buffer always contains observations covering the span of the prediction space, and thus that the model is better equipped to give accurate estimates overall, and not be biased by the distribution of the latest available observations. A similar configuration has been used for multi-class classification, with one buffer for each class [47]. When learning from a data stream, an observation (x_i, y_i) is allocated to a bin in the experience replay buffer based on the value of y_i, and consequently, the oldest observation of that bin is discarded. The mini-batch used for backpropagation is sampled

from the experience replay by prioritization using the softmax function. At every update of the model, the observations are given a probability of being sampled for the next update by:

$$p_i = \frac{e^{(y_i - \hat{y}_i)^2}}{\sum_{c=1}^{C} e^{(y_c - \hat{y}_c)^2}}, \tag{3.14}$$

where $C = nN$ is the total number of observations in the buffer. It can be seen that a higher model error on an observation results in a higher probability of being sampled for the next training step. This method is well known in the reinforcement learning field, and has shown to be an improvement over sampling observations from a uniform distribution [46], due to added focus on areas where the model performs poorly. The n-bin prioritized experience replay algorithm is given formally as:

Algorithm 2 n-Bin Prioritized Experience Replay

```
Load n-bin replay buffer 𝒟 filled with historical data
while learning do
    if new observations (x, y) available then
        Enqueue observations (x, y) in appropriate bin in 𝒟
        Dequeue appropriate bin
        Calculate p_i, i = 1, ..., C
        Randomly sample K observations by probabilities p_i from 𝒟
        Backpropagate on sampled observations
    end if
end while
```

Retention of observations from the entire range of the prediction space in this fashion allows smaller replay buffers. Rather than retaining all previous observations for further training, a small subset is kept, making this method memory efficient. In addition, dividing the prediction space into bins ensures that historical observations in one bin are available as long as no new observations stream into that bin. Thus, the algorithm can take into account older representations while making updates during operation.

3.2.7 Streaming learning system

The streaming learning system is based on a pre-trained and validated model, which is loaded as the *baseline model*, on which to iteratively perform updates during operation. As surface measurements become available during operation, they are first fed to the DNN to estimate at-bit density. Subsequently, they are stored in the holding buffer until its corresponding label, the density log measurement, is available. At this point, the completed observation is moved from the holding buffer to the experience replay buffer, and then used for further training. Note that the learning phase begins at time τ when $d_{d,\tau} \geq d_{s,0}$, meaning that we are not interested in the density log measurements above the first available drilling parameter measurements. **Algorithm 3** summarizes the method.

Algorithm 3 System Overview

Pre-train and validate model on historical data
Load model as *baseline model*
Load experience replay buffer filled with historical data
while learning **do**
    Receive x_i, estimate $\hat{y}_i$
    Store x_i along with $d_{s,i}$ in holding buffer
    if y_k available **then**
        for all observations in holding buffer **do**
            Find indices j for all observations satisfying $d_{d,k} > d_{s,j}$
            Pair x_j with y_{k-1}, $\forall j$
        end for
        Remove these observations from holding buffer, and update experience replay buffer
        Backpropagate on observations sampled from experience replay buffer
    end if
end while

3.3 Results

3.3.1 Pre-training and validation

Pre-training and validation of the baseline model was an iterative approach. Hyperparameters such as DNN architecture (neurons and layers), learning rate, mini-batch size and number of epochs were tuned in an informal search based on validation set performance. Upon arrival on a satisfactory model, the streaming learning hyperparameters were tuned on the same data set. Table 3.2 shows the hyperparameter settings. The resulting baseline model from pre-training and validation had 3 hidden layers, each with 12 neurons. Streaming learning hyperparameters refers to the settings for streaming learning during operation. It can be seen that here, the learning rate and mini-batches sampled from the experience replay are smaller than during pre-training. Experience replay bin limits refers to the limits on the density log used in **Algorithm 2** to select the appropriate bin. Density log for the data used in this study was in the range 2.0 - 2.7 (g/cm^3), so that observations below the first bin limit would belong to the *low* bin, observations between the two limits belonged to the *mid* bin, and lastly, observations above the second bin limit belonged to the *high* bin. The hyperparameter values presented in Table 3.2 should make decent initial values for similar problems. However, note that the experience replay bin limits in particular will be very case-dependent. The bin limits ensure retention of observations in different parts of the prediction space, making an understanding of the dependent variable key. We suggest consideration of important areas in the prediction space when tuning these parameters. If several areas are important, the number of bins could also be increased.

Table 3.2: Summary of hyperparameters for the density log model.

Pre-Training Hyperparameter	**Value**
Learning rate	$7.5 \cdot 10^{-5}$
Hidden layers	3
Neurons in hidden layers	12
Mini-batch size	128
Epochs	25
Optimizer	Adam
Streaming Learning Hyperparameter	**Value**
Learning rate	$7.5 \cdot 10^{-6}$
Experience replay bins	3
Experience replay bin sizes	256
Experience replay bin limits	[2.115, 2.535]
Mini-batch size	16
Epochs	1
Optimizer	Adam

3.3.2 Streaming learning results

This subsection is dedicated to the presentation and evaluation of the performance of the streaming learning approach compared to the baseline model. We provide results for the validation well, along with the 3 test wells. Along with each plot, the mean absolute error (MAE) is presented. MAE is given by:

$$\text{MAE} = \frac{1}{m} \sum_{i=1}^{m} |y_i - \hat{y}_i|. \tag{3.15}$$

Although the raw data used for the algorithm was time data, the results are converted into depth data with equidistant points at a resolution of 1 m by downsampling. At every integer depth, data points within 0.5 m are averaged. Figure 3.4 shows the comparison between the baseline and streaming learning approach on the validation well. For the baseline model, one can see that the model suffers from bias on low-density observations throughout the wellbore, and that the model gradually overestimates density from 6000 m and towards the end. An inspection of the raw data revealed that for this wellbore, the torque gradually

increased which indicates concept drift. The streaming learning approach can be seen to mitigate both the bias and the drift, resulting in an MAE decrease from 0.1087 g/cm^3 to 0.0615 g/cm^3. This is a relative decrease of approximately 43%.

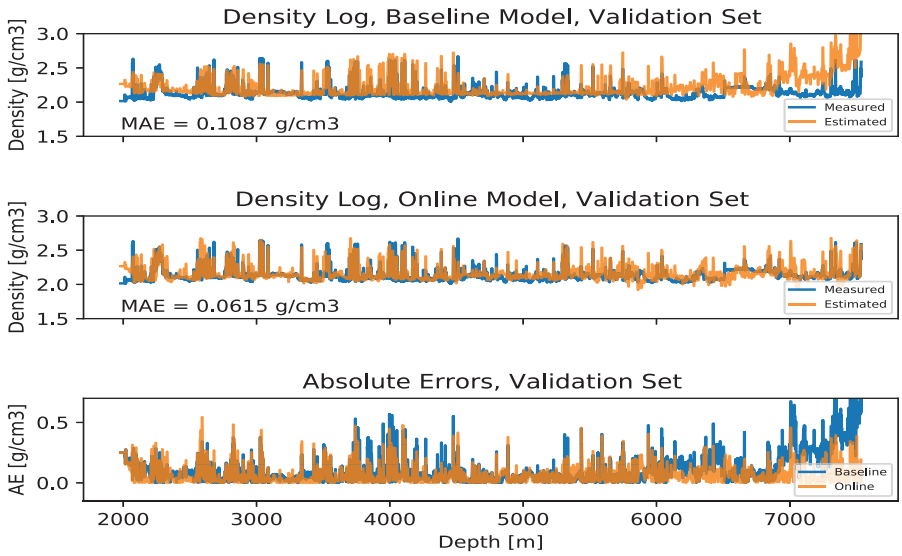

Figure 3.4: Measured and estimates on the validation set. **Top:** Baseline model performance. **Middle:** Streaming learning performance. **Bottom:** Absolute errors (AE).

Figures 3.5 and 3.6 show the same comparisons for test wells 1 and 2. On these wellbores, the baseline model performs quite well. Still, on test set 1 at approximately 5000 - 5500 m depth, the baseline model is visibly off measurements. On test set 2, the baseline model overestimates low density observations. For test wells 1 and 2, the streaming learning approach reduces these errors, resulting in 21% and 7% decreases in MAE, respectively. For test well 2, the resulting improvement from the streaming learning algorithm is incremental, and the added complexity of method implementation compared to the baseline model counterpart may not be worthwhile. For wellbores with similar input/output relationships to those of the training wells, the baseline model will perform well without the need for continuous updates. However, in many cases it is not known in advance whether or not this will be the case. Such considerations should

be a part of the validation process. If it is found that a static model performs well during validation, a traditional supervised learning approach might be sufficient. If, however, heterogeneity between wellbores is found during this process, a streaming learning approach might be warranted to correct for drifts and shifts in concepts.

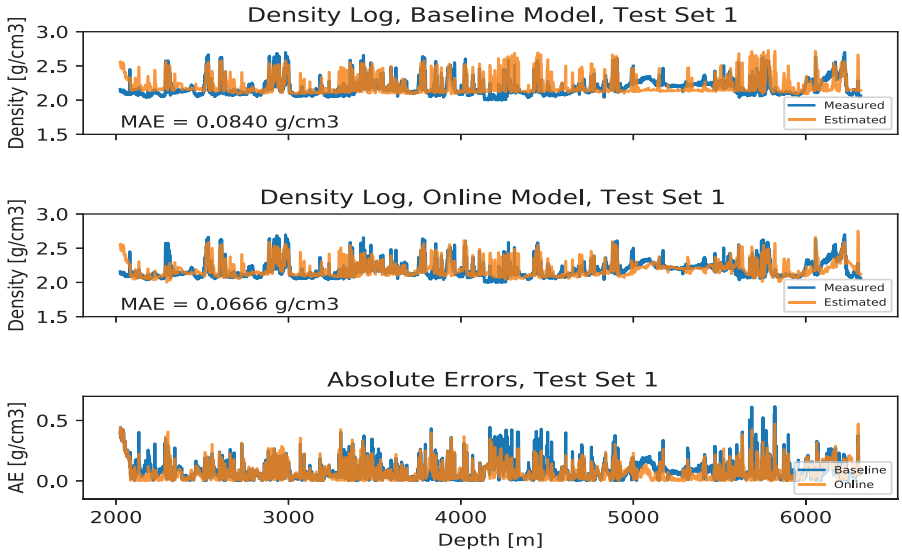

Figure 3.5: Measured and estimates on test set 1. **Top:** Baseline model performance. **Middle:** Streaming learning performance. **Bottom:** Absolute errors (AE).

Lastly, Figure 3.7 shows the results for test well 3. It can be seen that the baseline model is very flat, and unable to capture any trends in the at-bit density. This indicates that the mapping from drilling parameters to at-bit density is significantly different from that of the wellbores in the training set. We can call this a shift in concepts. Although the streaming learning approach is not as good as for the previous wells, it is to a much larger degree able to capture the trends by adapting to the concept shift, resulting in an MAE decrease of 36%. Even though the maximum recorded density for these wellbores is 2.7 g/cm^3, it can be seen that the online models estimates densities above this value at depths 5030 m and 5110 m. At depth 5030 m, an unusually low value for RPM was measured, along with a high WOB. At 5110 m, an unusually high MSE was recorded, indicating an unusual combination of drilling parameters. As neural

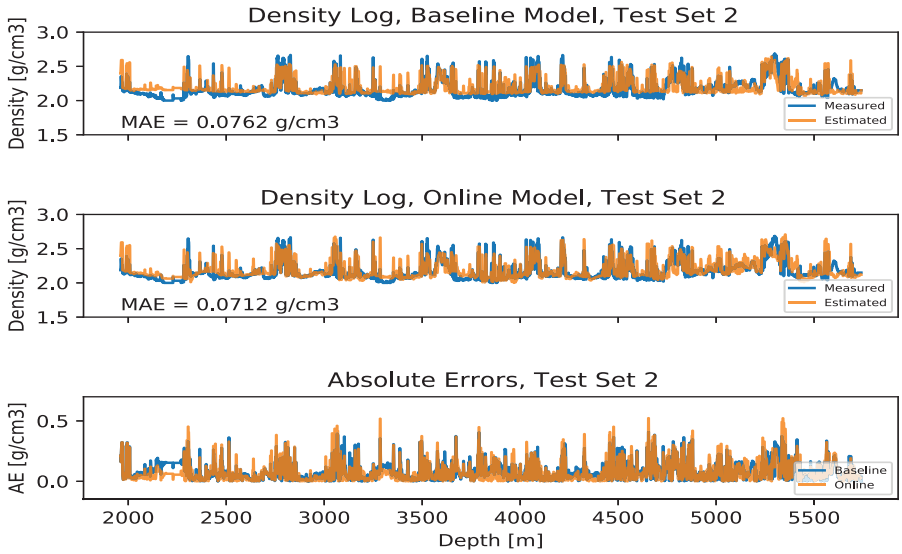

Figure 3.6: Measured and estimates on test set 2. **Top:** Baseline model performance. **Middle:** Streaming learning performance. **Bottom:** Absolute errors (AE).

networks do not extrapolate well, these events result in erroneously high density estimates. The flattening effect apparent for this wellbore can also to some extent be observed in the other wellbores. In the other wellbores this takes the form of a cut-off effect, so that low-density observations are not estimated well by the baseline model. This is likely due to heterogeneity between the training wellbores. Since the input/output relationships in the wells differ, the baseline model is trained to fit an "average" of these, resulting in a compressive effect on the predictions.

The rate of adaption to newly available data can be seen to differ for the different data sets. For the validation, test 1 and test 2 sets, we can see that the baseline model overestimates the low-density observations in the beginning of the drilling operation. For the streaming learning approach, we can see that this is quickly mitigated by the online retraining. Also for test set 3, the performance is improved, although the corrections are not as fast. From the poor baseline model performance, we could argue that this wellbore is the most different from the training sets, and that the speed of the online retraining is dependent on how much the model must be corrected to fit well to the newly available data. The learning rate of

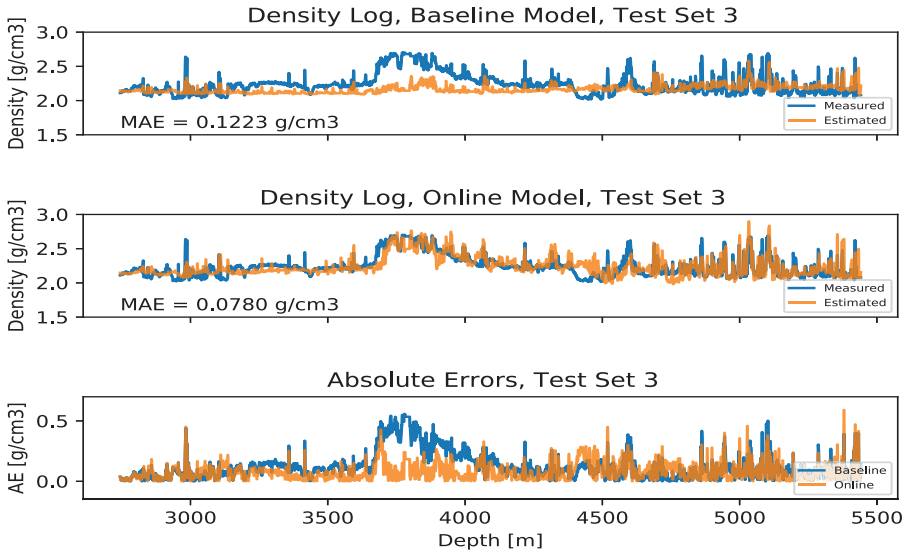

Figure 3.7: Measured and estimates on test set 3. **Top:** Baseline model performance. **Middle:** Streaming learning performance. **Bottom:** Absolute errors (AE).

the system determines the size of the gradient descent updates, however, setting this hyperparameter too high can lead to unstable optimization. Thus, the online retraining rate must be a trade-off between speed and stability.

From the absolute errors on the depth converted datasets, we can investigate the statistical significance of the difference in performance between the two approaches, along with effect size and statistical power. We can perform paired, two-tailed t-tests for each wellbore to determine statistical significance. The null hypothesis is $H_0 : \mu_1 = \mu_2$, where μ_1 is the population mean absolute error for the baseline model, and μ_2 is the population mean absolute error for the streaming learning approach. We also determine Cohen's d, which is a measure of standardized effect size. The proposed interpretation of d is along a continuum, with conventional *small*, *medium* and *large* effect sizes at approximately 0.2, 0.5 and 0.8, respectively. Combined with a statistical significance level $\alpha_0 = 0.05$, which is the accepted probability that we are failing to reject a false null hypothesis (type I error), we can find the statistical power, $1 - \beta$, where β

quantifies the probability of rejecting the null hypothesis given that it is actually correct (type II error) [57]. In Table 3.3 we present m, the number of observations in each depth converted dataset, MAE_b, the mean absolute error using the baseline model, MAE_s the mean absolute error using the streaming learning approach, $\Delta\text{MAE} = \text{MAE}_b - \text{MAE}_s$, Cohen's d, p-values and statistical power, $1 - \beta$. We can see from the p-values that the difference in performance are statistically significant for all wellbores. Thus, we reject H_0. From Cohen's d, we see that the effect size on the validation and test 3 wellbores are medium to large. The effect size for the test 1 wellbore is medium, and small for the test 2 wellbore. For all wellbores, the probability of committing a type II error, is ≈ 0.

Table 3.3: Results of power analysis.

Data set	m	MAE_b [g/cm^3]	MAE_s [g/cm^3]	ΔMAE [g/cm^3]	d	p	$1-\beta$
Val	5556	0.1087	0.0615	0.0472	0.61	$1.2 \cdot 10^{-206}$	≈ 1
Test 1	4296	0.0840	0.0666	0.0174	0.42	$9.1 \cdot 10^{-82}$	≈ 1
Test 2	3780	0.0762	0.0712	0.005	0.14	$9.3 \cdot 10^{-10}$	≈ 1
Test 3	2698	0.1223	0.0780	0.0443	0.58	$3.4 \cdot 10^{-92}$	≈ 1

3.3.3 Data visualization with t-SNE

Data visualizations with t-SNE in two dimensions are provided to complement the results in Figures 3.4 - 3.7 (see Appendix A for a brief summary of the t-SNE method). From these plots, we have observed several different scenarios: drift, minor offsets and significant concept shift. We wish to visualize the data to better understand these effects. In the t-SNE analyses, a set of data points in 7 dimensions, taken as:

$$\theta_i = [\text{ROP}_i, \text{RPM}_i, \text{WOB}_i, \text{T}_i, \text{MSE}_i, \text{HMSE}_i, \text{Density log}_i], \tag{3.16}$$

is reduced to the set of data points $[\phi_1, \phi_2]$ in two dimensions. Using this setup, we can identify the structures of the data in the wellbores, for example if similar drilling parameters result in similar or different density log measurements for different wellbores. Figure 3.8 illustrates the two-dimensional mapping for 3 random subsets from the same training well, each containing one third of the observations. As expected, these subsets occupy similar spaces in the plot. This indicates that a model

trained on one of these subsets could perform well on the other two. Figure 3.9 illustrates the same analysis on training well 1, and test sets 1 and 2. The baseline model was found to perform quite well for these test sets, and from this t-SNE analysis, we can see that the observations from these wellbores indeed overlap reasonably well with the training well in the two-dimensional mapping. Lastly, we inspect Figure 3.10, which shows the two-dimensional mapping for training well 1, along with the validation well and test well 3. The validation well and test well 3 exhibited concept drift and shift respectively, and the baseline model performed poorly on them. From the plot, we can observe natural clustering within each wellbore, which might be attributed to the fact that low-density and high-density observations should be different. However, it can be seen that several clusters from the validation well and test well are isolated from each other and from the training well. As they form separate clusters, this indicates that all 3 wellbores belong to their own natural grouping. Because of this, any baseline model regardless of model type and architecture, cannot be trained on this training well and be expected to generalize well to the others. In other words, streaming learning is essential for obtaining acceptable performance in this case.

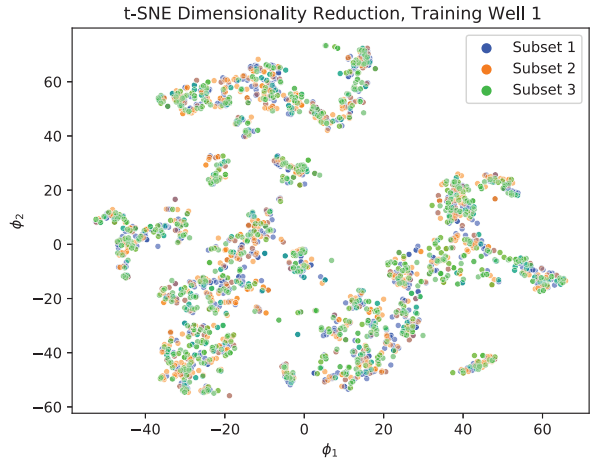

Figure 3.8: t-SNE plot for 3 random subsets from training set 1.

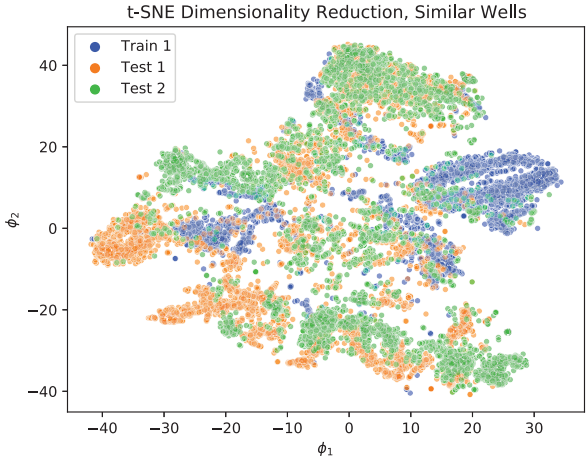

Figure 3.9: t-SNE plot for 3 random subsets from training set 1, test set 1 and test set 2.

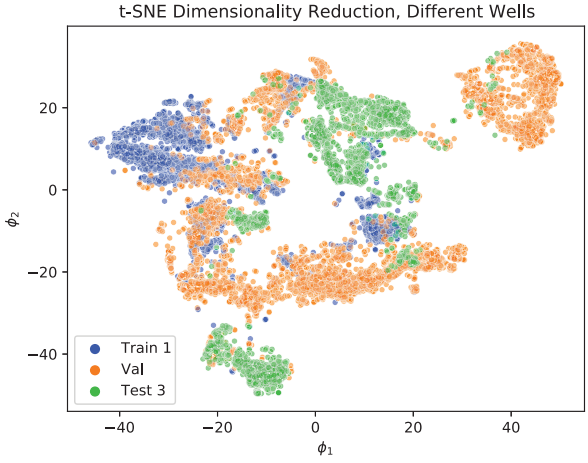

Figure 3.10: t-SNE plot for 3 random subsets from training set 1, validation set and test set 3.

3.4 Conclusions & further work

Since the density log is typically mounted 20-30 m behind the bit, the driller is rendered blind to at-bit conditions. In the current work, a DNN is used to estimate at-bit density from drilling parameters to eliminate this delay. The DNN is pre-trained on historical data from wells on a field operated by Equinor, serving as a baseline model. Using delayed density log measurements, the model is continuously updated during operation. This streaming learning approach allows adjustments to changing conditions that are not explicitly included in the model as variables. Comparisons of the results for the baseline model and the streaming learning approach indeed show that performance in terms of mean absolute error can be greatly improved using a streaming learning approach. This is especially true for wellbores where the relationships between drilling parameters and density log are significantly different from what the model has seen before during training. The method gives preliminary at-bit density log estimates that are available in real-time, while adapting to change, thus increasing generalizability so that the model is applicable to a wider range of cases. t-SNE is used to visualize the data from different wellbores and shows that the datasets are structurally different. This indicates that a pre-trained model, regardless of model architecture, will be unable to generalize to all the wellbores used in the analysis. It also serves as motivation for a streaming learning approach.

For further work, attempts should be made at a streaming learning approach for other LWD tools, as having several at-bit logs would further nuance the bottomhole information.

Acknowledgments

The authors wish to thank Equinor and NTNU for funding this project, and the Troll licence for sharing their drilling data. We also want to thank colleagues at Equinor who have provided valuable domain expertise, among them the REAL team.

Appendix A. *t-distributed stochastic neighbor embedding*

t-distributed Stochastic Neighbor Embedding (t-SNE) [58] falls within the category of unsupervised learning algorithms. It is typically used for visualization of high-dimensional data by dimensionality reduction. It is a nonlinear method capable of preserving the local structure of high-dimensional data while revealing global structures such as clusters. When converting the high-dimensional data $\vartheta = \{\theta_1, \theta_2, ...\theta_m\}$ to a low-dimensional mapping $\varphi = \{\phi_1, \phi_2, ..., \phi_m\}$, t-SNE starts by converting the high-dimensional Euclidean distances between datapoints to similarities $p_{i|j}$, quantifying the conditional probability that θ_i would pick θ_j as its neighbor if neighbors were picked in proportion to their probability density under a Gaussian centered at θ_i. This similarity is given as:

$$p_{i|j} = \frac{\exp\left(-||\theta_i - \theta_j||^2/2\sigma_i^2\right)}{\sum_{k\neq i}\exp\left(-||\theta_i - \theta_k||^2/2\sigma_i^2\right)}. \tag{3.17}$$

From these, the joint probabilities are defined to be symmetrized conditional probabilities, that is:

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}. \tag{3.18}$$

σ_i is the variance of the Gaussian centered at θ_i. This parameter can be indirectly tuned by the user through the *perplexity* hyperparameter. For a user-specified perplexity, t-SNE performs a binary search for the value of σ_i that produces a probability distribution P_i over all the other data points with the same perplexity $Perp(P_i)$. This is defined as:

$$Perp(P_i) = 2^{H(P_i)}, \tag{3.19}$$

where $H(P_i)$ is the Shannon entropy measured in bits:

$$H(P_i) = -\sum_j p_{j|i}\log_2 p_{i|j}. \tag{3.20}$$

Perplexity can be viewed as a smoothing measure for the number of effective neighbors, and t-SNE is robust to changes in this parameter. For

the low-dimensional mappings, the similarities q_{ij} are computed using a Student t-distribution with one degree of freedom, resulting in:

$$q_{ij} = \frac{(1 + ||\phi_i - \phi_j||^2)^{-1}}{\sum_{k \neq l}(1 + ||\phi_k - \phi_l||^2)^{-1}}. \tag{3.21}$$

Note that p_{ii} and q_{ii} are set to 0 since t-SNE is only interested in modeling pairwise similarities. Next, t-SNE minimizes the Kullback-Leibler divergence between the two probability distributions P and Q through gradient descent. The Kullback-Leibler divergence is given by:

$$C = KL(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}, \tag{3.22}$$

from which the gradient w.r.t the low-dimensional map can found to be:

$$\frac{\partial C}{\partial \phi_i} = 4 \sum_j (p_{ij} - q_{ij})(1 + ||\phi_i - \phi_j||^2)^{-1}(\phi_i - \phi_j), \tag{3.23}$$

which can be used to update the low-dimensional mapping φ from an initial value.

Chapter 4

Paper B: A divided and prioritized experience replay approach for streaming regression

Arnø ML, Godhavn J-M, Aamo OM. A divided and prioritized experience replay approach for streaming regression. *MethodsX (Accepted)*

A Divided and Prioritized Experience Replay Approach for Streaming Regression

Mikkel Leite Arnø*, Department of Engineering Cybernetics, Norwegian University of Science and Technology, 7491 Trondheim, Norway; **John-Morten Godhavn,** Equinor Research Center, 7053 Ranheim, Norway; and **Ole Morten Aamo,** Department of Engineering Cybernetics, Norwegian University of Science and Technology, 7491 Trondheim, Norway

*Corresponding author; email: mikkel.l.arno@ntnu.no

Abstract

In the streaming learning setting, an agent is presented with a data stream on which to learn from in an online fashion. A common problem is catastrophic forgetting of old knowledge due to updates to the model. Mitigating catastrophic forgetting has received a lot of attention, and a variety of methods exist to solve this problem. In this paper, we present a divided and prioritized experience replay approach for streaming regression, in which relevant observations are retained in the replay, and extra focus is added to poorly estimated observations through prioritization. Using a real-world dataset, the method is compared to the standard sliding window approach. A statistical power analysis is performed, showing how our approach improves performance on rare, important events at a trade-off in performance for more common observations. Close inspections of the dataset are provided, with emphasis on areas where the standard approach fails. A rephrasing of the problem to a binary classification problem is performed to separate common and rare, important events. These results provide an added perspective regarding the improvement made on rare events.

4.1 Introduction

In supervised learning, generalizability is a primary focus during the development of a model. During training, the labels are available, and are

used to supervise the learning of a function that maps from inputs to an output. The goal is for the model to learn a function that gives accurate predictions for unseen observations, for which labels are not available. This is the essence of generalization, and relies on the assumption that the unseen observations come from the same distribution as the training observations. However, this assumption fails for many real-word problems. Shifts in independent or dependent variables, an evolving underlying process, and dependence on variables not included in the model are all examples of nonstationarity, which is harmful to the predictive power of such models [39], [40]. Nonstationarity occurs to some extent in most real world data sets, and has been a motivating factor for the paradigm of streaming learning, where a model is presented with a stream of data on which to continuously learn from in an online fashion. This allows adaptation to changing data distribution, although the approach is prone to catastrophic forgetting [41]. In the setting of streaming learning, the goal is to leverage newly available data to adapt to changing environments while still performing well on previous observations [59]. These two objectives might be conflicting, giving rise to the *stability-plasticity* dilemma [60], asking how one can stay stable to irrelevant events, while plastic to new information. This problem has been addressed in several ways, perhaps most commonly by use of experience replay, where new observations from the data stream are mixed with older observations as they become available [9].

Streaming data can occur in many situations, and several problems can be solved by learning from these streams. Classification with dynamic selection of appropriate window [61], and classification using a streaming random forest [62] have been investigated. Clustering of data streams [63], multi-task learning with a global loss function [64], and multiple output linear regression [65] are other examples. The challenges of streaming learning have been discussed in many earlier publications, with emphasis on managing catastrophic forgetting. Reducing the risk of catastrophic forgetting has been approached in several ways, among them regularization, [42], [43], where weight updates are constrained so previously learned relationships are not erased, and ensembling methods, where multiple models are trained, and their outputs are combined with some form of majority voting [40], [44], [45]. Various experience replay configurations have also been developed for this purpose, among them

stream clustering methods to retain valuable information [47], and prioritization of samples to train on [46], where the former has with benefit been applied to streaming classification, and the latter to reinforcement learning. These differ from the aforementioned methods, as they focus on which observations to retain and train on rather than on the model itself.

In the current work, a divided and prioritized experience replay approach for streaming regression is presented to mitigate the effects of catastrophic forgetting while allowing adaptation to nonstationarity. We adopt both the philosophy of retaining relevant knowledge in the replay, along with that of prioritization. A deep neural network (DNN) is trained and validated on historical data. This model serves as a baseline model which is deployed for streaming learning during operation, using this approach. To demonstrate its effect, the method is applied to a real-world dataset, and benchmarked against a standard sliding window approach. The method was first presented in [1] as a case study. In the current work, however, a thorough comparison to the standard sliding window approach for streaming regression is given, with focus on rare, important events, and discussions of areas where the standard sliding window fails.

The paper is structured as follows: section 4.2 presents the methods used in this work, and section 4.3 presents our results in a case study, comparing our method with a standard sliding window. Lastly, section 4.4 offers conclusions.

4.2 Method

The selected model architecture for this work was a DNN. DNNs consist of multiple layers of interconnected neurons with nonlinear activation functions. This allows extraction of latent, possibly nonlinear features within the data. First, we provide notation for the DNN:

$$
\begin{aligned}
L &: \text{number of layers in the DNN} \\
m &: \text{number of observations} \\
f &: \text{number of features} \\
x &: \text{features} \\
y &: \text{target variable} \\
s_l &: \text{number of neurons in layer } l \in 1, ..., L \\
(x_i, y_i) &: i\text{-th observation, } i \in 1, ..., m \\
\mathbf{w}^{[l]} &: \text{trainable weight matrix for layer } l \\
\mathbf{b}^{[l]} &: \text{trainable bias vector for layer } l
\end{aligned}
$$

and

$$
\mathbf{x} = \begin{bmatrix} \vdots & \vdots & & \vdots \\ x_1 & x_2 & \dots & x_m \\ \vdots & \vdots & & \vdots \end{bmatrix} \quad \in \mathbb{R}^{f \text{ x } m}, \tag{4.1}
$$

$$
\mathbf{y} = \begin{bmatrix} y_1 & y_2 & \dots & y_m \end{bmatrix} \quad \in \mathbb{R}^{1 \text{ x } m}, \tag{4.2}
$$

$$
\mathbf{w}^{[l]} = \begin{bmatrix} \dots & w_1^{[l]\top} & \dots \\ \dots & w_2^{[l]\top} & \dots \\ & \vdots & \\ \dots & w_{s_{l-1}}^{[l]\top} & \dots \end{bmatrix} \quad \in \mathbb{R}^{s_{l-1} \text{ x } s_l}, \tag{4.3}
$$

$$
\mathbf{b}^{[l]} = \begin{bmatrix} b_1^{[l]} \\ b_2^{[l]} \\ \vdots \\ b_{s_l}^{[l]} \end{bmatrix} \quad \in \mathbb{R}^{s_l \text{ x } 1}. \tag{4.4}
$$

To make a prediction on a single observation of inputs x_i, the input is mapped to $\hat{y}_i$ by forward propagation through the DNN:

$$
z_i^{[l]} = \mathbf{w}^{[l]\top} a_i^{[l-1]} + \mathbf{b}^{[l]} \quad l = 1, ..., L, \tag{4.5}
$$

$$a_i^{[l]} = g^{[l]}(z_i^{[l]}) \quad l = 1, ..., L-1, \tag{4.6}$$

$$a_i^{[l]} = \max\{\gamma z_i^{[l]}, z_i^{[l]}\} \quad l = 1, ..., L-1, \tag{4.7}$$

$$\hat{y}_i = a_i^{[L]} = z_i^{[L]}. \tag{4.8}$$

Equation (4.5) describes the linear part of the forward propagation, mapping from one layer to the next throughout the DNN. This starts at $a_i^{[0]} = x_i$. At each hidden layer, these linear combinations are passed through nonlinear activation functions $g^{[l]}$, as described in Equation (4.6). In this work, the leaky ReLU activation functions given in Equation (4.7) are used, where γ is the slope in the left half plane. Lastly, the output layer outputs the prediction, which for this work is a real number, resulting in a regression layer. This is described in Equation (4.8).

For learning on a mini-batch of size m_b, inputs $\mathbf{x}_b$ and outputs $\mathbf{y}_b$ are used. The forward propagation for the mini-batch is given by:

$$\mathbf{z}^{[l]} = \mathbf{w}^{[l]\top}\mathbf{a}^{[l-1]} + \mathbf{b}^{[l]}\mathbf{1} \quad l = 1, ..., L, \tag{4.9}$$

$$\mathbf{a}^{[l]} = g^{[l]}(\mathbf{z}^{[l]}) \quad l = 1, ..., L-1, \tag{4.10}$$

$$\mathbf{a}^{[l]} = \max\{\gamma \mathbf{z}^{[l]}, \mathbf{z}^{[l]}\} \quad l = 1, ..., L-1. \tag{4.11}$$

$$\hat{\mathbf{y}}_b = \mathbf{a}^{[L]} = \mathbf{z}^{[L]}, \tag{4.12}$$

where $\mathbf{1} \in \mathbb{R}^{1 \text{ x } m_b}$ is a row vector of ones broadcasting the bias term to each observation of the mini-batch.

The weight initialization is He normal [36], which pulls the weights in each layer from a truncated normal distribution with mean $\mu = 0$ and standard deviation $\sigma = \sqrt{\frac{1}{s_{[l-1]}}}$. The weight initialization for layer l is then given by:

$$\mathbf{w}^{[l]} \in \mathbb{R}^{s_{l-1} \text{ x } s_l} \sim \mathcal{N}\left(\left[0, (s_{[l-1]})^{-1}\right]\right). \tag{4.13}$$

The biases, $\mathbf{b}^{[l]}$, are initialized as zeros. The optimizer used was Adam optimization [38] , which adaptively estimates appropriate momentum for the gradient updates.

The parameters $\mathbf{w}^{[l]}$ and $\mathbf{b}^{[l]}$ are iteratively updated by means of a gradient-based optimizer to minimize some cost function describing a distance between true labels and predictions. This is done by finding the gradients of the cost function J, w.r.t the trainable parameters, $\frac{\partial J}{\partial \mathbf{w}}$ and $\frac{\partial J}{\partial \mathbf{b}}$. For this work the cost is the mean squared error between the true labels and the predictions, defined by:

$$J = \frac{1}{2m_b}\left|\hat{\mathbf{y}} - \mathbf{y}\right|^2. \tag{4.14}$$

For supervised learning, the procedure of iteratively updating the trainable parameters, along with other hyperparameters such as model architecture, is typically done repeatedly in a validation process to assess the model's ability to generalize to unseen observations. However, the ability to generalize relies on the assumption that the unseen observations come from the same distribution as the training data, which for many real-world datasets is an invalid assumption. The target variable may depend on features not included in the model, shifts may occur in the independent or dependent variables, or the underlying process may evolve. These can all be contributors to poor generalization, and are called nonstationarity. In streaming learning, the goal is to bridge the gap between data distributions by adapting to new available observations. For this to work, labels must be available so that supervision of the updates can be achieved. In addition to adaption to changing distributions, it is of interest to "remember" older, relevant observations.

To stay stable to older observations while being plastic to new ones, the prioritized $n-$bin experience replay was developed. This experience replay configuration allows retention of observations spanning the prediction space $y \in [y_{min}, y_{max}]$ by splitting it into bins. We denote the buffer as $\mathscr{D} \in \mathbb{R}^{n \text{ x } N}$, where n is the number of bins, and N is the capacity of each bin. When observation (x_i, y_i) becomes available from the data stream, it will be placed in a bin after assessment of which bin in the prediction space y_i belongs to. Subsequently, the oldest observation in the same bin is discarded. Using this configuration, observations spanning the prediction space are retained, eliminating bias towards the distribution of the latest available observations, which occurs in the standard sliding window. Additionally, the mini-batch sampled from the experience replay

for further learning is sampled by prioritization using the softmax function, so that each observation is assigned a probability of being sampled for training by:

$$p_j = \frac{e^{(y_j - \hat{y}_j)^2}}{\sum_{c=1}^{C} e^{(y_c - \hat{y}_c)^2}}, \quad j = 1, ..., C, \tag{4.15}$$

where $C = nN$ is the total number of observations in the replay. Assigning the probabilities based on the softmax of the model's mean squared error on the observations results in added focus on observations for which the model performs poorly, as they are more likely to be sampled. By combining the prioritized $n-$bin with the DNN using the Adam optimizer, we obtain our streaming learning algorithm, given in **Algorithm 4**.

Algorithm 4 Streaming Learning Using Prioritized $n-$bin Experience Replay & Adam Optimization

Load baseline model
Load prioritized $n-$bin replay (pre-filled with historical data)
$V_{\mathbf{dw}} = 0, S_{\mathbf{dw}} = 0, V_{\mathbf{db}} = 0, S_{\mathbf{db}} = 0,$
while learning (on iteration i) **do**
    Receive x_i
    Forward prop on x_i to estimate $\hat{y}_i$:
    $z_i^{[1]} = \mathbf{w}^{[1]\top} x_i + \mathbf{b}^{[1]}$
    $a_i^{[1]} = g^{[1]}(z_i^{[1]})$
    $z_i^{[2]} = \mathbf{w}^{[2]\top} a_i^{[1]} + \mathbf{b}^{[2]}$
    $a_i^{[2]} = g^{[2]}(z_i^{[2]})$
    $\vdots$
    $a_i^{[L]} = z_i^{[L]}$
    $\hat{y}_i = a_i^{[L]}$
    } Prediction when input x_i is available.
    if new (x_i, y_i) pair available **then**
        Enqueue (x_i, y_i) in appropriate bin
        Dequeue appropriate bin
        Calculate p_j, $j = 1, ..., C$
        Sample observations by probabilities
        p_j to obtain mini-batch $\mathbf{x}_b$
    } Prioritized $n-$bin experience replay.
        Forward prop on $\mathbf{x}_b$:
        $\mathbf{z}^{[1]} = \mathbf{w}^{[1]\top} \mathbf{x}_b + \mathbf{b}^{[1]} \mathbf{1}$
        $\mathbf{a}^{[1]} = g^{[1]}(\mathbf{z}^{[1]})$
        $\mathbf{z}^{[2]} = \mathbf{w}^{[2]\top} \mathbf{a}^{[1]} + \mathbf{b}^{[2]} \mathbf{1}$
        $\mathbf{a}^{[2]} = g^{[2]}(\mathbf{z}^{[2]})$
        $\vdots$
        $\mathbf{a}^{[L]} = \mathbf{z}^{[L]}$
        $\hat{\mathbf{y}}_b = \mathbf{a}^{[L]}$
        Compute cost J
    } Adam optimization.

Backpropagate using the chain rule
to compute gradients $\frac{\partial J}{\partial \mathbf{w}}$ and $\frac{\partial J}{\partial \mathbf{b}}$
$V_{\mathbf{dw}} = \beta_1 V_{\mathbf{dw}} + (1 - \beta_1)\frac{\partial J}{\partial \mathbf{w}}$
$V_{\mathbf{db}} = \beta_1 V_{\mathbf{db}} + (1 - \beta_1)\frac{\partial J}{\partial \mathbf{b}}$
$S_{\mathbf{dw}} = \beta_2 S_{\mathbf{dw}} + (1 - \beta_2)\frac{\partial J}{\partial \mathbf{w}}^2$
$S_{\mathbf{db}} = \beta_2 S_{\mathbf{db}} + (1 - \beta_2)\frac{\partial J}{\partial \mathbf{b}}^2$
$V^c_{\mathbf{dw}} = \frac{V_{\mathbf{dw}}}{1-\beta_1^i}$
$V^c_{\mathbf{db}} = \frac{V_{\mathbf{db}}}{1-\beta_1^i}$
$S^c_{\mathbf{dw}} = \frac{S_{\mathbf{dw}}}{1-\beta_2^i}$
$S^c_{\mathbf{db}} = \frac{S_{\mathbf{db}}}{1-\beta_2^i}$
$\mathbf{w} = \mathbf{w} - \alpha\frac{V^c_{\mathbf{dw}}}{\sqrt{S^c_{\mathbf{dw}}}+\varepsilon}$
$\mathbf{b} = \mathbf{b} - \alpha\frac{V^c_{\mathbf{db}}}{\sqrt{S^c_{\mathbf{db}}}+\varepsilon}$

} Adam optimization.

end if
end while

β_1, β_2 and ε are tunable hyperparameters for Adam optimization, $V_{\mathbf{dw}}$ and $V_{\mathbf{db}}$ are biased first moment estimates, and $S_{\mathbf{dw}}$ and $S_{\mathbf{db}}$ are biased second raw moment estimates. Superscript c denotes their bias-corrected counterparts. α is the learning rate.

4.3 Case study

4.3.1 Problem description

The method presented in this work was developed in order to estimate the density of drilled lithology. This is traditionally measured using the density logging tool, which is a specialized logging while drilling (LWD) tool. Determining the density of the drilled lithology is of interest for several reasons, among them best-practice selection of drilling parameters to ensure a safe and efficient operation. Especially, accurate separation of high-density and low-density lithology can reduce the risk of dysfunctions like buckling, severe doglegs, washout and vibrations, resulting in lost time. However, the density log is delayed due to its placement behind the bit, making mechanical drilling parameters the earliest indicators of

change in drilled lithology, although it is very difficult for humans to directly interpret density from these.

To eliminate the density log delay, we propose to estimate a virtual density log using parameters available at the bit, i.e. mechanical drilling parameters. Since the true label of the density log is available after the distance from the log to the bit is drilled, we can pair the delayed density log with drilling parameters measured earlier at the same depth to obtain complete input/output observations that can be used to further supervise updates to our model during operation. This turns the problem into a streaming regression problem with delayed labels. Drilling data from wells on a field operated by Equinor was used for this work. After data cleaning and removal of irrelevant observations, the training set contained approximately 740 000 observations from 5 different wellbores, while the validation set contained 365 000 observations from one wellbore. Lastly, the test set contained 227 000 observations from one wellbore. As per standard convention for deep learning, the data was normalized and scaled so that each feature had zero mean and unit variance.

Several mechanical drilling parameters are available during drilling. v is the drilling velocity (ft/hr), w is the weight on bit (lb), T is the torque (lb-ft), ϕ is the drillstring rotation (rpm). The driller may directly control v, w, and ϕ, while T is dependent on a variety of factors, among them rock properties, w, and ϕ. A metric commonly monitored during drilling is the mechanical specific energy, U_{ms}, which quantifies the energy required to remove a unit volume of rock. It is independent of the driller's actions, is different for different lithologies [31], and is given by:

$$U_{ms} = \frac{w}{a_b} + \frac{120\pi \cdot \phi \cdot T}{a_b \cdot v}, \tag{4.16}$$

where a_b is the bit area (in^2). To account for weakening of the rock ahead of the bit due to flow through the nozzles, the hydraulic mechanical specific energy [32], U_{hms}, can be defined by:

$$U_{hms} = \frac{w}{a_b} + \frac{120\pi \cdot \phi \cdot T}{a_b \cdot v} + \frac{1154\eta \cdot \Delta p_b \cdot q}{a_b \cdot v}, \tag{4.17}$$

where η is the hydraulig energy reduction factor, Δp_b is the bit pressure drop at the nozzle (psi), and q is the flow rate of drilling fluid (gpm). From the available mechanical parameters, we define the input vector of predictors for the DNN, $\mathbf{x}$, as:

$$\mathbf{x} = [v, \quad \phi, \quad w, \quad T, \quad U_{ms}, \quad U_{hms}]^\top. \tag{4.18}$$

4.3.2 Pre-training and validation

Pre-training and validation of the baseline model was performed in an informal search. Training was performed using Adam optimization, where the training set was randomly shuffled, and divided into mini-batches of size m_b. The calculated gradients for each mini-batch are noisy estimates of the true gradients of the entire data set, and this additive noise is useful for avoiding getting stuck in poor local minima or saddle points early in training, and for improving generalization [66]. Neural network architecture and training configuration hyperparameters were iteratively tuned on the training and validation set split. Upon completion of this process, the streaming hyperparameters were tuned iteratively on the validation set. These hyperparameters are related to the streaming learning during operation. The density log limits on this field lies in the range 2.0 - 2.7 (g/cm^3). Tuning of the experience replay parameters resulted in 3 bins with limits at 2.115 (g/cm^3) and 2.535 (g/cm^3), effectively dividing the prediction space so that one bin retains observations below 2.115 (g/cm^3), the second between 2.115 (g/cm^3) and 2.535 (g/cm^3), and the last retains observations above 2.535 (g/cm^3). These parameters can be seen in Table 4.1, which summarizes the hyperparameters. We can also see that the learning rate is decreased by one order of magnitude for the streaming learning, which was necessary in order to facilitate stability.

Table 4.1: Summary of hyperparameters for the model.

Pre-Training Hyperparameter	**Value**
Learning rate	$7.5 \cdot 10^{-5}$
Hidden layers	3
Neurons in hidden layers	12
Mini-batch size	128
Epochs	25
Optimizer	Adam
Streaming Learning Hyperparameter	**Value**
Learning rate	$7.5 \cdot 10^{-6}$
Experience replay bins	3
Experience replay bin sizes	256
Experience replay bin limits	[2.115, 2.535]
Mini-batch size	16
Epochs	1
Optimizer	Adam

4.3.3 Test set results

After pre-training and validation of the baseline model, it was deployed for streaming regression during operation on the test set, where the density logging tool is placed 20 meters behind the bit. This was done using both the prioritized n-bin experience replay, and a standard sliding window for comparison. As summarized in Table 4.1, the prioritized n-bin replay consisted of 3 bins, each containing 256 observations at all times. The standard sliding window used for comparison contained the same amount of observations in total, 768. Although the algorithm is run on data in time domain, the density log (along with other LWD tools) are most interesting in depth domain. For this reason, the presented results are converted to depth domain with equidistant points at 1 meter resolution by downsampling. At every integer depth, observations within 0.5 meters are averaged. Since detection of the hard stringers is important, we evaluate both approaches in terms of mean absolute error (MAE) separately for low density (< 2.35 (g/cm^3)) and high density (≥ 2.35 (g/cm^3)) observations.

Figure 4.1 illustrates measured and estimated at-bit density vs depth on the entire test set for both methods. We define a false high density

estimate a *low risk* error. If the driller takes action, lowering T and increasing w based on such an estimate, the drilling will simply be sub-optimal. Conversely, we define a false low density estimate as a *high risk* error. If hard stringers are not detected, and action is not taken, risk of dysfunctions such as buckling, severe doglegs, washout and vibrations are increased. On the test set as a whole, the prioritized n-bin experience replay leads to a 22% increase in MAE for true low density observations and a 22% decrease in MAE for true high density observations, compared to the standard sliding window.

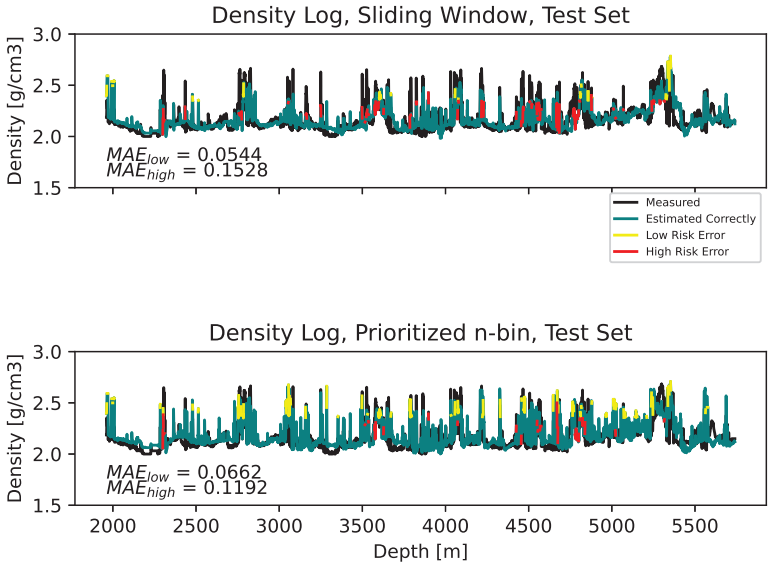

Figure 4.1: Top: Measured and estimated density log using the standard sliding window. **Bottom:** Measured and estimated density log using the prioritized n-bin sliding window.

We wish to investigate these results in more detail to provide some insight into the effect of applying the prioritized $n-$ replay, both on low density and high density observations. First, we perform paired, two-tailed t-tests to investigate the statistical significance of the changes in MAE, through obtaining $p-$values. The null hypothesis becomes $H_0 : \mu_1 = \mu_2$, where μ_1

is the population mean absolute error for the standard sliding window approach, and μ_2 is the population mean absolute error for our method. For this test, we select a significance level of $\alpha_0 = 0.05$. To quantify a standardized *effect size*, we also calculate Cohen's d. This value, in combination with α_0, can in turn be used to calculate the statistical power, 1 - β, where β is the probability of a type II error. Table 4.2 summarizes the results of our statistical analysis, where m is number of observations, MAE_s is the mean absolute error using a sliding window, MAE_n is the mean absolute error using our method, and $\Delta\text{MAE} = \text{MAE}_s - \text{MAE}_n$. Through the t-tests, we confirm the statistical significance at our chosen significance level. Observing Cohen's d show that for true low density observations, our method leads to a standardized effect size of $d = -0.3021$ (where the negative sign signifies worsening), while for the true high density observations, $d = 0.4578$. Interpreting these along a continuum, as proposed in literature [57], where 0.2, 0.5, and 0.8 are low, medium and high effect sizes, we see that our approach leads to a small to medium worsening on low density observations, and a medium to large improvement on high density observations.

Table 4.2: Results of power analysis.

True value	m	MAE_s [g/cm^3]	MAE_n [g/cm^3]	ΔMAE [g/cm^3]	d	p	$1-\beta$
< 2.35	3422	0.0544	0.0662	-0.01175	-0.3021	$4.63 \cdot 10^{-35}$	≈ 1
≥ 2.35	358	0.1528	0.1192	0.03359	0.4578	$2.51 \cdot 10^{-9}$	≈ 1

Figures 4.2 - 4.5 are zoomed plots on the test set. In Figure 4.2, at 2755 m, a hard stringer occurs. Using the standard sliding window, this event is completely undetected. We can see at 2779 m that the model accurately detects the next stringer. Here, 24 m further down, high density observations from the missed stringer has been passed by the density logging tool (20 m behind), and these observations are available for training in the sliding window. We can assume that the model misses the first stringer since the sliding window only contains observations from the earlier low density observations, resulting in *catastrophic forgetting*. Using our approach, we can see that also the stringer missed by the standard sliding window is detected.

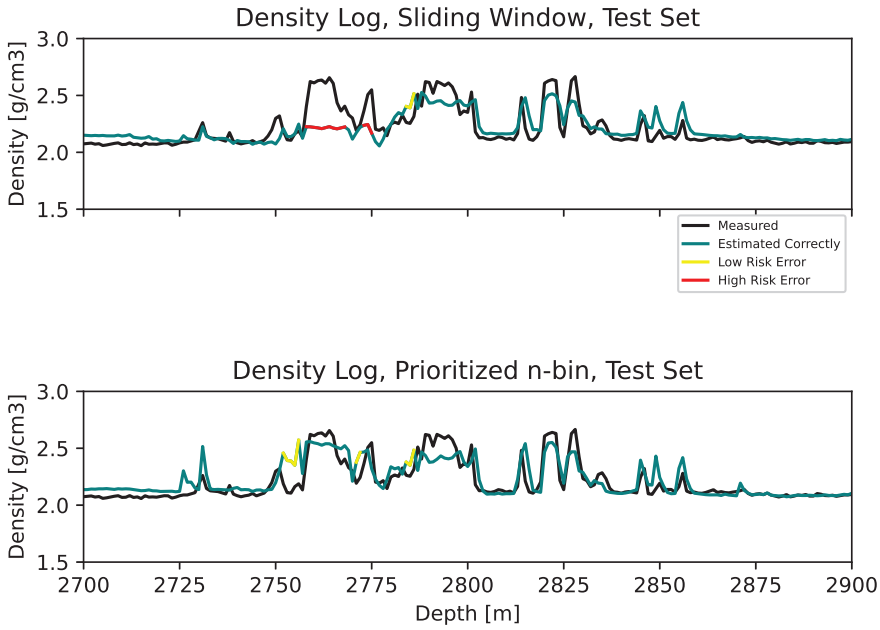

Figure 4.2: Zoom 1. **Top:** Measured and estimated density log using the standard sliding window. **Bottom:** Measured and estimated density log using the prioritized n-bin sliding window.

In Figure 4.3, we can see that the sliding window misses the stringer at 3157 - 3170 m, along with the one at 3250 m. Our approach accurately detects these events. At the same time, we observe that some observations at approximately 3050 m and 3285 are overestimated. In Figure 4.4, stringers at 3500 m and 3523 - 3536 m are missed by the sliding window, but detected by our approach. Lastly, Figure 4.5 shows missed stringers by the sliding window at 4450 - 4530 m, along with a poor transition to low density observations at approximately 4850 m. These are all improved using our approach. However, we observe some false high estimates at 4650 - 4700 m.

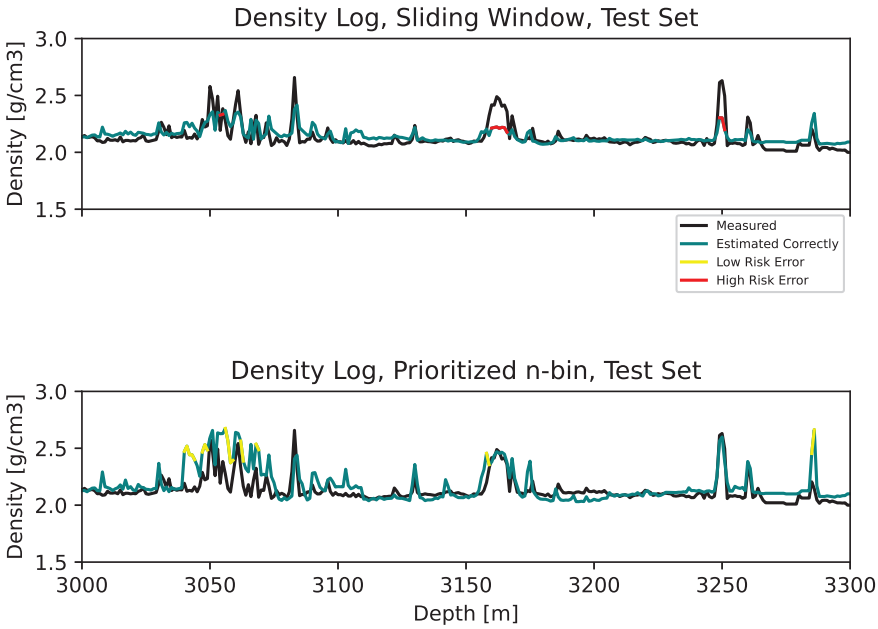

Figure 4.3: Zoom 2. **Top:** Measured and estimated density log using the standard sliding window. **Bottom:** Measured and estimated density log using the prioritized n-bin sliding window.

Figure 4.4: Zoom 3 **Top:** Measured and estimated density log using the standard sliding window. **Bottom:** Measured and estimated density log using the prioritized n-bin sliding window.

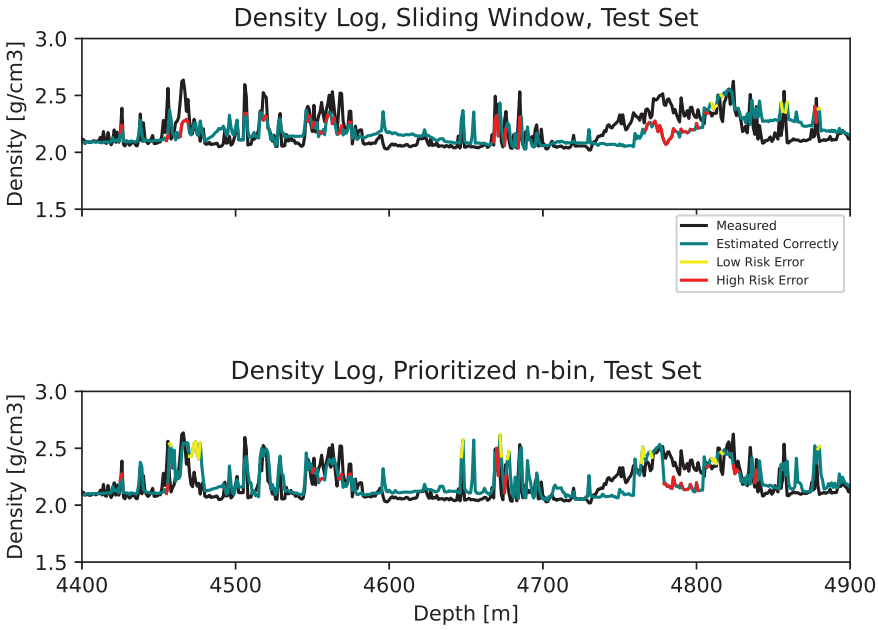

Figure 4.5: Zoom 4 **Top:** Measured and estimated density log using the standard sliding window. **Bottom:** Measured and estimated density log using the prioritized n-bin sliding window.

Since the density log is naturally divided into low- and high density observations, we rephrase the problem into a binary classification problem. Although this is an oversimplification, it adds to the previous analysis in terms of detection of hard stringers, and high risk/low risk errors, as defined previously. As can be seen from m in Table 4.2, less than 10% of the observations in the test set are stringers, which makes this an unbalanced data set. Figure 4.6 shows the resulting confusion matrices for both approaches, dividing the observations into low density and high density observations as previously. We see that the sliding window is very accurate in prediction of low density observations, with an accuracy rate of 0.97. However, only an accuracy of 0.55 is achieved for high density observations. Using the prioritized $n-$bin, the accuracy on low density observations is slightly decreased, to 0.92, while detection of high density

observations is greatly improved, scoring an accuracy of 0.8. The balanced accuracies for the sliding window and the prioritized $n-$ bin are 0.76 and 0.86, respectively. Due to drillstring compression and elongation, measurement error on depth can occur. To account for this, a 1 m acceptance was implemented, meaning that if a prediction is within 1 m of the correct label, it is accepted as correct. The confusion matrices using the 1 m acceptance are shown in Figure 4.7. We observe that the results for both approaches are improved. Here, the balanced accuracies are 0.89 for the sliding window and 0.95 for our approach.

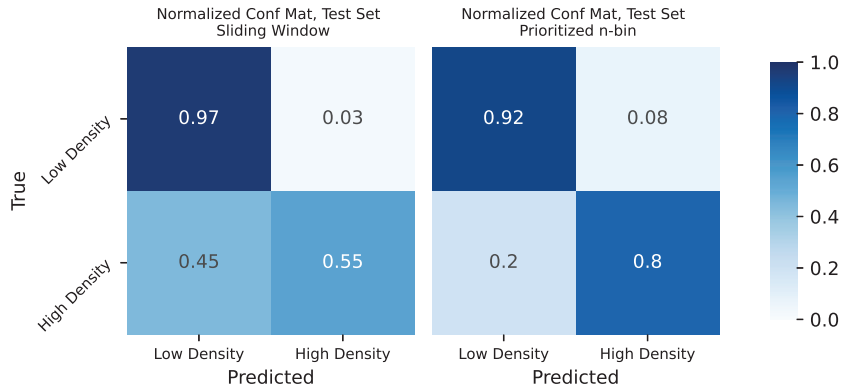

Figure 4.6: Confusion matrices. **Left:** Standard sliding window. **Right:** Prioritized n-bin method.

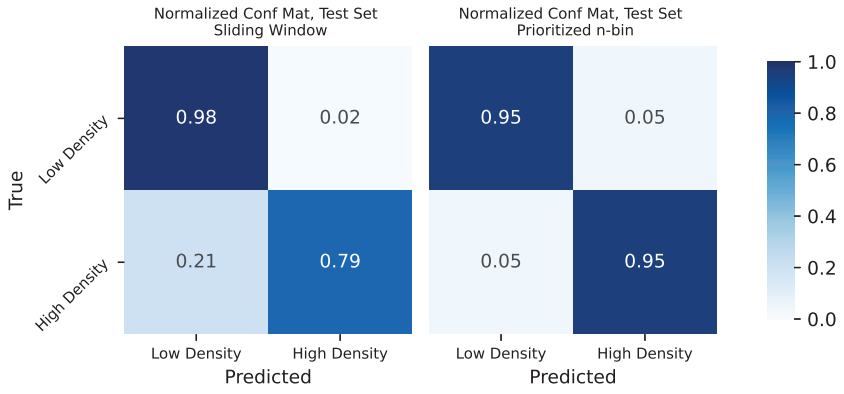

Figure 4.7: Confusion matrices applying a 1 m acceptance. **Left:** Standard sliding window. **Right:** Prioritized n-bin method.

4.4 Conclusions

A divided and prioritized experience replay suited for streaming regression has been presented, making use of known ideas such as retention of relevant observations along with prioritization. This makes the model less biased to the distribution of the latest available observations, and results in more frequent sampling of observations where the model performs poorly.

Comparison to a standard sliding window has been made on real-world data. From our results, we can deduce that the standard sliding window results in forgetting of old, rare events, leading to failure to detect them. Especially in cases where exclusively common events have been observed, the model becomes biased towards these observations, failing to detect new, important events. The presented $n-$bin method, on the other hand, retains observations from the entire range of the prediction space, resulting in more accurate estimates for these observations at a small cost

in accuracy on the common events. Also, some false detections are observed. In addition to analyses on the regression formulation, a simplified rephrasing to a binary classification problem has been performed. These results divide the observations into two classes to provide added insight into the improved performance on the rare events.

Acknowledgments

The authors wish to thank Equinor and NTNU for funding this project. We would also like to thank the Troll licence for sharing their data.

Chapter 5

Paper C: At-bit virtual neutron log estimated from real-time drilling data

Arnø ML, Godhavn J-M, Aamo OM. At-bit virtual neutron log estimated from real-time drilling data. *SPE/IADC International Drilling Conference and Exhibition, Galveston, Texas (Accepted).* March 8-10, 2022.

Chapter 6

Paper D: Classification of drilled lithology in real-time using deep learning with online calibration

Arnø ML, Godhavn J-M, Aamo OM. Classification of drilled lithology in real-time using deep learning with online calibration. *SPE Drilling & Completion (Accepted)*

Chapter 7

Paper E: Real-time classification of drilled lithology from drilling data using deep learning with online calibration

Arnø ML, Godhavn J-M, Aamo OM. Real-time classification of drilled lithology from drilling data using deep learning with online calibration. In *proceedings of the SPE/IADC International Drilling Conference and Exhibition, Virtual*. March 8-12, 2021. SPE-204093-MS https://doi.org/10.2118/204093-MS

This conference paper is an early stage contribution to at-bit lithology classification. The paper was invited for peer-review in the SPE Drilling & Completion journal, which is presented in Paper D, where updated results are included. Due to significant overlap, the conference paper is omitted.

Chapter 8

Paper F: Deep reinforcement learning applied to managed pressure drilling

Arnø ML, Godhavn J-M, Aamo OM. Deep reinforcement learning applied to managed pressure drilling. In *proceedings of the SPE Norway Subsurface Conference, Virtual*. November 2-3, 2020. SPE-200757-MS https://doi.org/10.2118/200757-MS

Chapter 9

Conclusions & further work

In this thesis, one instance of deep reinforcement learning to control bottomhole pressure during pipe connection has been presented in a simulation study. It is demonstrated that the agent capably controls bottomhole pressure for different well depths under the challenging conditions present during pipe connection. However, this work is a simplified case, and several areas for future work and improvements can be identified. One example is the simple hydraulics model used to simulate flow and pressure in the wellbore. A natural extension would be the use of more advanced models, such as partial differential equations (PDEs). Also, no noise is added to the measurements, and no theoretical stability guarantees are provided. As pressure control during drilling is critical, future endeavours to add to this work are necessary for its viability in the real world. Still, the presented work exemplifies how reinforcement learning can be applied to tackle this problem, and could serve as a starting point to build on.

A series of contributions towards LWD estimation and lithology classification using deep neural networks in a streaming learning configuration have also been presented. These focus on at-bit estimation and classification using drilling parameters as inputs, which has not been devoted much attention in existing literature. For this purpose, the prioritized $n-$bin experience replay was developed to learn from delayed LWD measurements to continuously adapt to changing relationships between drilling parameters and LWDs, which has not been done before. Real drilling data provided by Equinor has been used in the training, validation and testing of these

models, which show promise in making useful predictions that the driller can use to make informed decisions in real-time, rather than waiting for LWD measurements, which due to their placement could be 20-120 minutes delayed, depending on ROP. As these systems neither control any aspect of the drilling, nor make any suggestions in this regard, decision-making is left to the driller, meaning that these systems pose low risk implementation-wise. To facilitate further validation of these contributions, a real-time implementation on data streams available from drilling rigs would be an obvious next step. Another suggestion for future work relates to the data used. On real-time data streams, data cannot be cleaned in the same manner as in fixed datasets, and since a streaming learning system learns continuously from these streams, the models are susceptible to bad updates due to bad data. Further efforts to identify bad data could have a sizeable effect on the performance of these models. One possibility could be to use other logs, such as the delta rho log, to identify these observations.

Bibliography

[1] Arnø ML, Godhavn J-M, and Aamo OM. At-bit estimation of rock density from real-time drilling data using deep learning with online calibration. *Journal of Petroleum Science and Engineering*, 206:109006, 2021. https://doi.org/10.1016/j.petrol.2021.109006.

[2] Groover MP. Automation. In Dorf RC and Kusiac A, editors, *Handbook of design, manufacturing and automation*, chapter 1, pages 3–21. John Wiley & Sons, 1994.

[3] Thorogood J, Aldred W, Florence F, and Iversen F. Drilling automation: technologies, terminology, and parallels with other industries. *SPE drilling & completion*, 25(04):419–425, 2010. https://doi.org/10.2118/119884-PA.

[4] Parasuraman R, Sheridan TB, and Wickens CD. A model for types and levels of human interaction with automation. *IEEE Transactions on systems, man, and cybernetics-Part A: Systems and Humans*, 30(3):286–297, 2000. https://doi.org/10.1109/3468.844354.

[5] Mitchell TM. *Machine learning*. McGraw-Hill, New York, 1997.

[6] Lee JH, Shin J, and Realff MJ. Machine learning: Overview of the recent progresses and implications for the process systems engineering field. *Computers & Chemical Engineering*, 114:111–121, 2018. https://doi.org/10.1016/j.compchemeng.2017.10.008.

[7] Krizhevsky A, Sutskever I, and Hinton GE. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[8] Bojarski M, Del Testa D, Dworakowski D, Firner B, Flepp B, Goyal P, Jackel LD, Monfort M, Muller U, and Zhang J. End to end learning for self-driving cars. *arXiv preprint*, 2016. https://arxiv.org/abs/1604.07316v1.

[9] Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, and Ostrovski G. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. https://doi.org/10.1038/nature14236.

[10] Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, Tassa Y, Silver D, and Wierstra D. Continuous control with deep reinforcement learning. *arXiv preprint*, 2015. https://arxiv.org/abs/1509.02971v6.

[11] Diez-Olivan A, Del Ser J, Galar D, and Sierra B. Data fusion and machine learning for industrial prognosis: Trends and perspectives towards industry 4.0. *Information Fusion*, 50:92–111, 2019. https://doi.org/10.1016/j.inffus.2018.10.005.

[12] Crow DJG, Anderson K, Hawkes AD, and Brandon N. Impact of drilling costs on the US gas industry: Prospects for automation. *Energies*, 11(9):2241, 2018. https://doi.org/10.3390/en11092241.

[13] Ambrus A, Pournazari P, Ashok P, Shor R, and van Oort E. Overcoming barriers to adoption of drilling automation: Moving towards automated well manufacturing. In *proceedings of the SPE/IADC Drilling Conference and Exhibition, London, England, UK*. OnePetro, March 17-19, 2015. SPE-173164-MS https://doi.org/10.2118/173164-MS.

[14] Iversen F, Gressgård LJ, Thorogood JL, Balov MK, and Hepsø V. Drilling automation: potential for human error. *SPE Drilling & Completion*, 28(01):45–59, 2013. https://doi.org/10.2118/151474-PA.

[15] Arnø ML, Godhavn J-M, and Aamo OM. A divided and prioritized experience replay approach for streaming regression. *MethodsX (Accepted)*, 2021.

[16] Arnø ML, Godhavn J-M, and Aamo OM. At-bit virtual neutron log estimated from real-time drilling data. In *SPE/IADC International Drilling Conference and Exhibition, Galveston, Texas (Accepted)*. OnePetro, March 8-10, 2022.

[17] Arnø ML, Godhavn J-M, and Aamo OM. Classification of drilled lithology in real-time using deep learning with online calibration. *SPE Drilling & Completion (Accepted)*, 2021.

[18] Arnø ML, Godhavn J-M, and Aamo OM. Real-time classification of drilled lithology from drilling data using deep learning with online calibration. In *proceedings of SPE/IADC International Drilling Conference and Exhibition, Virtual*. OnePetro, March 8-12, 2021. SPE-204093-MS https://doi.org/10.2118/204093-MS.

[19] Arnø ML, Godhavn J-M, and Aamo OM. Deep reinforcement learning applied to managed pressure drilling. In *proceedings of the SPE Norway Subsurface Conference, Virtual*. OnePetro, November 2-3, 2020. SPE-200757-MS https://doi.org/10.2118/200757-MS.

[20] Bourgoyne Jr AT and Young Jr FS. A multiple regression approach to optimal drilling and abnormal pressure detection. *Society of Petroleum Engineers Journal*, 14(04):371–384, 1974. https://doi.org/10.2118/4238-PA.

[21] Moazzeni A, Nabaei M, and Jegarluei SG. Decision making for reduction of nonproductive time through an integrated lost circulation prediction. *Petroleum science and technology*, 30(20):2097–2107, 2012. https://doi.org/10.1080/10916466.2010.495961.

[22] Arps JJ and Arps JL. The subsurface telemetry problem-a practical solution. *Journal of Petroleum Technology*, 16(05):487–493, 1964. https://doi.org/10.2118/710-PA.

[23] Rolon L, Mohaghegh SD, Ameri S, Gaskari R, and McDaniel B. Using artificial neural networks to generate synthetic well logs. *Journal of Natural Gas Science and Engineering*, 1(4-5):118–133, 2009. https://doi.org/10.1016/j.jngse.2009.08.003.

[24] Zhang D, Yuntian C, and Jin M. Synthetic well logs generation via recurrent neural networks. *Petroleum Exploration and Development*,

45(4):629–639, 2018. https://doi.org/10.1016/S1876-3804(18)30068-5.

[25] Jeong J, Park E, Emelyanova I, Pervukhina M, Esteban L, and Yun S-T. Application of conditional generative model for sonic log estimation considering measurement uncertainty. *Journal of Petroleum Science and Engineering*, 196:108028, 2021. https://doi.org/10.1016/j.petrol.2020.108028.

[26] Feng R, Grana D, and Balling N. Variational inference in bayesian neural network for well log prediction. *Geophysics*, 86(3):1–45, 2021. https://doi.org/10.1190/geo2020-0609.1.

[27] Dev VA and Eden MR. Formation lithology classification using scalable gradient boosted decision trees. *Computers & Chemical Engineering*, 128:392–404, 2019. https://doi.org/10.1016/j.compchemeng.2019.06.001.

[28] Al-Anazi A and Gates ID. On the capability of support vector machines to classify lithology from well logs. *Natural Resources Research*, 19(2):125–139, 2010. https://doi.org/10.1007/s11053-010-9118-9.

[29] Zeng L, Ren W, and Shan L. Attention-based bidirectional gated recurrent unit neural networks for well logs prediction and lithology identification. *Neurocomputing*, 414:153–171, 2020. https://doi.org/10.1016/j.neucom.2020.07.026.

[30] Tian M, Omre H, and Xu H. Inversion of well logs into lithology classes accounting for spatial dependencies by using hidden markov models and recurrent neural networks. *Journal of Petroleum Science and Engineering*, 196:107598, 2021. https://doi.org/10.1016/j.petrol.2020.107598.

[31] Dupriest FE and Koederitz WL. Maximizing drill rates with real-time surveillance of mechanical specific energy. In *proceedings of the SPE/IADC Drilling Conference, Amsterdam, Netherlands*. Society of Petroleum Engineers, February 23–25, 2005. SPE-92194-MS https://doi.org/10.2118/92194-MS.

[32] Osarogiagbon AU, Oloruntobi O, Khan F, Venkatesan R, and Butt S. Gamma ray log generation from drilling parameters using deep

learning. *Journal of Petroleum Science and Engineering*, 195:107906, 2020. https://doi.org/10.1016/j.petrol.2020.107906.

[33] Gowida A, Elkatatny S, Abdulraheem A, and Shehri DA. Synthetic well-log generation: New approach to predict formation bulk density while drilling using neural networks and fuzzy logic. In *proceedings of the International Petroleum Technology Conference, Dhahran, Kingdom of Saudi Arabia*. OnePetro, January 13–15, 2020. IPTC-19787-MS https://doi.org/10.2523/IPTC-19787-MS.

[34] Moazzeni A and Haffar MA. Artificial intelligence for lithology identification through real-time drilling data. *Journal of Earth Science & Climatic Change*, 6(3):1–4, 2015. http://dx.doi.org/10.4172/2157-7617.1000265.

[35] Leshno M, Lin VY, Pinkus A, and Schocken S. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6):861–867, 1993. https://doi.org/10.1016/S0893-6080(05)80131-5.

[36] He K, Zhang X, Ren S, and Sun J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile*, December 7-13, 2015. https://doi.ieeecomputersociety.org/10.1109/ICCV.2015.123.

[37] Ruder S. An overview of gradient descent optimization algorithms. *arXiv preprint*, 2016. https://arxiv.org/abs/1609.04747v2.

[38] Kingma DP and Ba J. Adam: A method for stochastic optimization. In *proceedings of the International Conference on Learning Representations (ICLR), San Diego, California, USA*, May 7-9, 2015. https://arxiv.org/abs/1412.6980.

[39] Ditzler G, Roveri M, Alippi C, and Polikar R. Learning in nonstationary environments: A survey. *IEEE Computational Intelligence Magazine*, 10(4):12–25, 2015. https://doi.org/10.1109/MCI.2015.2471196.

[40] Elwell R and Polikar R. Incremental learning of concept drift in nonstationary environments. *IEEE Transactions on Neural Networks*, 22(10):1517–1531, 2011. https://doi.org/10.1109/TNN.2011.2160459.

[41] McCloskey M and Cohen NJ. Catastrophic interference in connectionist networks: The sequential learning problem. In Bower GH, editor, *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989. https://doi.org/10.1016/S0079-7421(08)60536-8.

[42] Kirkpatrick J, Pascanu R, Rabinowitz N, Veness J, Desjardins G, Rusu AA, Milan K, Quan J, Ramalho T, and Grabska-Barwinska A. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017. https://doi.org/10.1073/pnas.1611835114.

[43] Li Z and Hoiem D. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017. https://doi.org/10.1109/TPAMI.2017.2773081.

[44] Wang H, Fan W, Yu PS, and Han J. Mining concept-drifting data streams using ensemble classifiers. In *proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, Washington DC, USA*, pages 226–235, August 24-27, 2003. https://doi.org/10.1145/956750.956778.

[45] Nishida K and Yamauchi K. Adaptive classifiers-ensemble system for tracking concept drift. In *proceedings of the International Conference on Machine Learning and Cybernetics, Hong Kong, China*, volume 6, pages 3607–3612. IEEE, August 19-22, 2007. https://doi.org/10.1109/ICMLC.2007.4370772.

[46] Schaul T, Quan J, Antonoglou I, and Silver D. Prioritized experience replay. In *proceedings of the International Conference on Learning Representations, ICLR, San Juan, Puerto Rico*, May 2-4, 2016. https://arxiv.org/abs/1511.05952v4.

[47] Hayes TL, Cahill ND, and Kanan C. Memory efficient experience replay for streaming learning. In *proceedings of the International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada*, pages 9769–9776. IEEE, May 20-24, 2019. https://doi.org/10.1109/ICRA.2019.8793982.

[48] Breyholtz Ø, Nygaard G, Godhavn J-M, and Vefring EH. Evaluating control designs for co-ordinating pump rates and choke valve during managed pressure drilling operations. In *proceedings of the 2009 IEEE*

Control Applications,(CCA) & Intelligent Control,(ISIC), pages 731–738. IEEE, 2009. https://doi.org/10.1109/CCA.2009.5281013.

[49] Petersen J, Rommetveit R, Bjorkevoll KS, and Froyen J. A general dynamic model for single and multi-phase flow operations during drilling, completion, well control and intervention. In *proceedings of the IADC/SPE Asia Pacific Drilling Technology Conference and Exhibition, Jakarta, Indonesia*. OnePetro, August 25-27, 2008. SPE-114688-MS https://doi.org/10.2118/114688-MS.

[50] Kaasa G-O, Stamnes ØN, Imsland L, and Aamo OM. Simplified hydraulics model used for intelligent estimation of downhole pressure for a managed-pressure-drilling control system. *SPE Drilling & Completion*, 27(01):127–138, 2012. https://doi.org/10.2118/143097-PA.

[51] Sutton RS and Barto AG. *Reinforcement learning: An introduction, 2nd edition*. MIT press, 2018.

[52] Bellman RE. *Dynamic Programming*. Courier Dover Publications, 1957.

[53] Waughman RJ, Kenner JV, and Moore RA. Real-time specific energy monitoring reveals drilling inefficiency and enhances the understanding of when to pull worn pdc bits. In *proceedings of the IADC/SPE Drilling Conference, Dallas, Texas*. Society of Petroleum Engineers, February 26–28, 2002. SPE-74520-MS https://doi.org/10.2118/74520-MS.

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

[55] Tunkiel AT, Sui D, and Wiktorski T. Training-while-drilling approach to inclination prediction in directional drilling utilizing recurrent neural networks. *Journal of Petroleum Science and Engineering*, 196:108128, 2021. https://doi.org/10.1016/j.petrol.2020.108128.

[56] Feng R, Grana D, and Balling N. Uncertainty quantification in fault detection using convolutional neural networks. *Geophysics*, 86(3):M41–M48, 2021. https://doi.org/10.1190/geo2020-0424.1.

[57] Portney LG. *Foundations of Clinical Research: Applications to Evidence-Based Practice, 4th Edition*. FA Davis Company. Pennsylvania, United States, 2020.

[58] Van der Maaten L and Hinton G. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(11):2579–2605, 2008.

[59] Hoens TR, Polikar R, and Chawla NV. Learning from streaming data with concept drift and imbalance: an overview. *Progress in Artificial Intelligence*, 1(1):89–101, 2012. http://dx.doi.org/10.1007%2Fs13748-011-0008-0.

[60] Grossberg S. Nonlinear neural networks: Principles, mechanisms, and architectures. *Neural networks*, 1(1):17–61, 1988. https://doi.org/10.1016/0893-6080(88)90021-4.

[61] Aggarwal CC, Han J, Wang J, and Yu PS. A framework for on-demand classification of evolving data streams. *IEEE Transactions on Knowledge and Data Engineering*, 18(5):577–589, 2006. https://doi.org/10.1109/TKDE.2006.69.

[62] Abdulsalam H, Skillicorn DB, and Martin P. Classification using streaming random forests. *IEEE Transactions on knowledge and Data Engineering*, 23(1):22–36, 2010. https://doi.org/10.1109/TKDE.2010.36.

[63] Wang C-D, Lai J-H, Huang D, and Zheng W-S. Svstream: A support vector-based algorithm for clustering data streams. *IEEE Transactions on Knowledge and Data Engineering*, 25(6):1410–1424, 2011. https://doi.org/10.1109/TKDE.2011.263.

[64] Dekel O, Long PM, and Singer Y. Online learning of multiple tasks with a shared loss. *Journal of Machine Learning Research*, 8(10), 2007.

[65] Li C, Wei F, Dong W, Wang X, Liu Q, and Zhang X. Dynamic structure embedded online multiple-output regression for streaming data. *IEEE transactions on pattern analysis and machine intelligence*, 41(2):323–336, 2018. https://doi.org/10.1109/TPAMI.2018.2794446.

[66] Keskar NS, Mudigere D, Nocedal J, Smelyanskiy M, and Tang PTP. On large-batch training for deep learning: Generalization gap and sharp

minima. In *proceedings of the International Conference of Learning representations, ICLR, Toulon, France*, April 24-26, 2017. https://arxiv.org/abs/1609.04836v2.

[67] Mao Z-Q. The physical dependence and the correlation characteristics of density and neutron logs. *Petrophysics*, 42(05), 2001.

[68] Khandelwal M and Singh TN. Artificial neural networks as a valuable tool for well log interpretation. *Petroleum science and technology*, 28(14):1381–1393, 2010. https://doi.org/10.1080/10916460903030482.

[69] Rolon LF, Mohaghegh SD, Ameri S, Gaskari R, and McDaniel B. Developing synthetic well logs for the upper devonian units in pennsylvania. In *proceedings of the SPE Eastern Regional Meeting, Morgantown, West Virginia*. OnePetro, September 14-16, 2005. SPE-98013-MS https://doi.org/10.2118/98013-MS.

[70] Kanfar R, Shaikh O, Yousefzadeh M, and Mukerji T. Real-time well log prediction from drilling data using deep learning. In *proceedings of the International Petroleum Technology Conference, Dhahran, Kingdom of Saudi Arabia*. OnePetro, January 13-15, 2020. IPTC-19693-MS https://doi.org/10.2523/IPTC-19693-MS.

[71] Salaheldin Elkatatny, Zeeshan Tariq, Mohamed Mahmoud, and Abdulazeez Abdulraheem. New insights into porosity determination using artificial intelligence techniques for carbonate reservoirs. *Petroleum*, 4(4):408–418, 2018. https://doi.org/10.1016/j.petlm.2018.04.002.

[72] Bukar I, Adamu MB, and Hassan U. A machine learning approach to shear sonic log prediction. In *proceedings of the SPE Nigeria Annual International Conference and Exhibition, Lagos, Nigeria*. OnePetro, August 5-7, 2019. SPE-198764-MS https://doi.org/10.2118/198764-MS.

[73] Deng C, Pan H, Fang S, Konaté AA, and Qin R. Support vector machine as an alternative method for lithology classification of crystalline rocks. *Journal of Geophysics and Engineering*, 14(2):341–349, 2017. https://doi.org/10.1088/1742-2140/aa5b5b.

[74] Imamverdiyev Y and Sukhostat L. Lithological facies classification using deep convolutional neural network. *Journal of Petroleum Science*

and Engineering, 174:216–228, 2019. https://doi.org/10.1016/j.petrol.2018.11.023.

[75] Elkattan M, Al Alfy IM, and Elawadi E. Intelligent integration of neutron, density and gamma ray data for subsurface characterization. *Sensing and Imaging*, 21(1):1–14, 2020. https://doi.org/10.1007/s11220-020-0277-4.

[76] Schmidhuber J. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015. https://doi.org/10.1016/j.neunet.2014.09.003.

[77] Kirkpatrick J, Pascanu R, Rabinowitz N, Veness J, Desjardins G, Rusu AA, Milan K, Quan J, Ramalho T, and Grabska-Barwinska A. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017. https://doi.org/10.1073/pnas.1611835114.

[78] Nishida K and Yamauchi K. Adaptive classifiers-ensemble system for tracking concept drift. In *proceedings of the 2007 International Conference on Machine Learning and Cybernetics, Hong Kong, China*, volume 6, pages 3607–3612. IEEE, August 19-22, 2007. https://doi.org/10.1109/ICMLC.2007.4370772.

[79] Israel R, Farthing J, Walker H, Covarrubias RG, Bryant J, and Vahle C. Development to delivery-a collaborative approach to implementing drilling automation. In *proceedings of the SPE/IADC Drilling Conference and Exhibition, The Hague, The Netherlands*. OnePetro, March 14-16, 2017. SPE-184695-MS https://doi.org/10.2118/184695-MS.

[80] Dunlop J, Isangulov R, Aldred WD, Sanchez HA, Flores JL, Herdoiza JA, Belaskie J, and Luppens JC. Increased rate of penetration through automation. In *proceedings of the SPE/IADC Drilling Conference and Exhibition, Amsterdam, The Netherlands*. OnePetro, March 1-3, 2011. SPE-139897-MS https://doi.org/10.2118/139897-MS.

[81] Breyholtz Ø and Nikolao M. Drilling automation: Presenting a framework for automated operations. *SPE Drilling & Completion*, 27(01):118–126, 2012. https://doi.org/10.2118/158109-PA.

ISBN 978-82-326-5550-2 (printed ver.)
ISBN 978-82-326-5884-8 (electronic ver.)
ISSN 1503-8181 (printed ver.)
ISSN 2703-8084 (online ver.)

NTNU
Norwegian University of
Science and Technology