

Kick Detection During Offshore Drilling using Artificial Intelligence

ANDREAS KVALBEIN FJETLAND

SUPERVISOR

Prof. Jing Zhou

University of Agder, 2019
Faculty of Engineering and Science
Department of Engineering

Abstract

An uncontrolled or unobserved influx or kick during drilling has the potential to induce a well blowout, one of the most harmful incidences during drilling both in regards to economic and environmental cost. Since kicks during drilling are serious risks, it is important both to improve kick detection performance and capabilities, and to develop automatic flux detection methodology. There are clear patterns during an influx incident. However, due to complex processes and sparse instrumentation, it is difficult to predict the behavior of kicks or losses based on sensor data combined with physical models alone. Emerging technologies within Deep Learning are however quite adept at picking up on and quantifying, subtle patterns in time series given enough data.

In this paper, new models for kick detection is developed by using Long Short-Term Memory (LSTM) and Bidirectional LSTM (BiLSTM), two types of Deep Recurrent Neural Network, for kick detection and influx size estimation during drilling operations. The proposed detection methodology is based on simulated drilling data and involves detecting and quantifying the influx of fluids between fractured formations and the wellbore in a large range of dynamic drilling simulations.

The results show that the proposed methods are effective both to detect and estimate the influx size during drilling operations so that corrective actions can be taken before any major problem occurs. The results further indicate that these methods can be used on readily available sensor data on the drill rig. Making it a suitable technology for both modern and older drilling rigs.

Preface

This thesis is a product of the master course MAS500, which concludes the Mechatronics Masters program at the University of Agder (UiA).

Working with drilling and applied artificial intelligence in this thesis has been a great opportunity to supplement the mechatronics master program with some deeper knowledge about drilling engineering and computer science. It proved to be a steep and interesting learning experience, made possible by help from experts within their fields who took the time and effort to share their knowledge and experience with me.

First off, thanks to my supervisor Prof. Jing Zhou (UiA), who has not only provided an endless list of good literature within kick detection during drilling but also good guidance and explanations to difficult concepts within kick detection and control. For AI and Deep Learning, a big thanks to Ph.D. researcher Darshana Abeyrathna (UiA, CAIR) for answering questions and providing help when I've been stuck.

This paper was made possible by the Open-Lab Drilling simulator created by NORCE. A big thanks to my primary contact at NORCE, Dr. Jan Einar Gravdal, for helping me out with both literature and answers for questions within drilling, and an even bigger thanks to both Jan Einar and OpenLab-Drilling team, who has helped me out whenever I experienced problems with my simulations.

For the IT infrastructure needed for this task, thanks to Sigurd Kristian Brinch (UiA) for help both in setting up the servers and as a discussion partner for how to properly use them. Last, but not least a thanks to the Mechatronics Innovation Lab for giving me access to their Nvidia DGX-1, a specialized server for advanced deep learning.

Andreas K. Fjetland

University of Agder
Grimstad, May 24, 2019

Contents

	Page
Abstract	III
Preface	V
Contents	VII
List of Figures	IX
List of Tables	X
Nomenclature	XI
1 Introduction	1
2 Theoretical Background	3
2.1 Drilling	3
2.2 Artificial intelligence	6
2.2.1 Machine learning	6
2.2.2 Neural Networks	6
2.2.3 Deep learning	7
2.2.4 Generalization and overfitting	8
2.2.5 Optimization algorithms	9
2.2.6 Mini-Batch	10
2.2.7 Augmented learning	10
2.2.8 Combined learning on real and synthetic data	10
3 State of the Art	11
3.1 Return Flow	11
3.2 Detection of Wellbore Anomalies through Pressures	11
3.3 Downhole Pressure Measurements	11
3.4 Connection Flow-backs	12
3.5 Gass kick detection	12
3.6 Automated Monitoring of Traditional Parameters	12
3.7 Detection Algorithms for MPD	12
3.8 AI in kick detection	13
4 Drilling Simulation	15
4.1 Simulator	15
4.1.1 Rig, drill string, and wellpath	15
4.1.2 Geology & Casing	16
4.1.3 Drilling fluid (mud)	17
4.1.4 Influx & mud loss	17
4.1.5 Limitations	19
4.2 Simulation setup	19

4.2.1	Influx simulation	19
4.2.2	Geology and mud density	20
4.2.3	Flow Rate	20
4.2.4	Choke opening	21
4.2.5	Tripping & Drilling	22
4.2.6	Top of string position, string velocity ROP and Surface RPM	22
4.2.7	Drill bit depth	23
5	Data handling and preprocessing	25
5.1	Data storage system	25
5.1.1	Selection criteria	25
5.1.2	System selection	26
5.2	Database Design	26
5.2.1	Log	26
5.2.2	Run - A batch of simulations	27
5.2.3	Sim - Simulation settings table	27
5.2.4	Data - Simulated drilling data	28
5.2.5	Case list and Fracture profiles	28
5.2.6	TrainingSet, SimUse and Use	29
5.2.7	Network and SensorSets	29
5.2.8	Results and ResultComment	30
5.3	Generating Training, Validation and Test sets	30
6	Prediction Models	31
6.1	Training features	31
6.1.1	Standard score	32
6.2	Conventional Comparison	32
6.3	Network design	33
6.3.1	Alternate Classification approach	34
6.3.2	Training Option	34
6.3.3	Testing	35
7	Results	37
7.1	Simulation data	37
7.2	Network total RMSE comparison	38
7.3	Comparing feature sets and classification methods on Batch 1 feature set 1-4	40
7.4	NN Influx prediction on Batch 1	44
7.5	Results from Batch 3, excluding drilling & tripping	46
7.6	Results with drilling & tripping	46
7.7	Results with drilling & tripping including extra sensors	46
7.8	Predictions during drilling	50
7.9	Predictions during tripping	52
7.10	Examining badly performing cases	54
7.11	Training progress BiLSTM B2 1a	56
7.12	Training progress LSTM B2 1b	56
8	Discussion	57

8.1	Simulation data	57
8.2	Data storage	58
8.3	Prediction Models	58
9	Conclusion	61
	References	62
A	Appendices	A - 1
A.1	Entry Relation Diagram	A - 1
A.2	Code for generating training sets	A - 2
	A.2.1 Main script	A - 2
	A.2.2 partitionTestValData	A - 3
A.3	Flow Functions	A - 4
A.4	LSTM Training function	A - 7

List of Figures

1	Illustration of well	3
2	Formation pressure limits and hydro-static well pressure	4
3	Neural network structure	6
4	LSTM module [1]	8
5	Generic drillpipe. Illustrations: NORCE	15
6	Well path of InclinedWell 2500m. Illustrations: NORCE	16
7	Default geological profiles used by OpenLab. Illustrations: NORCE	16
8	Drilling fluid interface. Illustration: NORCE	17
9	Influx injection example	18
10	Geopressure based influx example	18
11	Example of geological profile	21
12	Examples of randomly seeded flow profiles	21
13	Training progression examples on 100 node single LSTM layer	35
14	Simulation #6974 - Artificial influx	36
15	Simulation #6786 - Geopressure based influx	36
16	Simulation #6554 - Geopressure based influx and loss	36
17	RMSE Histogram of BiLSTM B2 1a	40
18	Prediction of influx mass rate. Where the blue line represents the simulated influx rate, and set 1, 3a & 4a represents the predicted responses	41
	c Low regression trigger	42
	d High regression trigger	42
	e Classification Network	42
	f Δ Flow trigger	42
19	Classification results the whole test set	42
20	Kick classification. Where the blue line represents the simulated influx rate on the left axis and the remaining lines is the binary classification of influx or no influx on the right axis	43
21	Non-recurrent network responses on Batch 3 with feature set 1a/b	45

22	LSTM and BiLSTM response on Batch 3 with feature set 1a/b	47
23	LSTM and BiLSTM response on Batch 2 with feature set 1a/b	48
24	LSTM and BiLSTM response on Batch 2 with feature set 5a/b	49
25	Simulated response of drilling simulation #12450	50
26	BiLSTM B2 1a - prediction on drilling simulation #12450	51
27	BiLSTM B3 1a - prediction on drilling simulation #12450	51
28	Simulated response of tripping simulation #12450	52
29	BiLSTM B2 1a - prediction on tripping simulation #12450	53
30	BiLSTM B3 1a - prediction on tripping simulation #12450	53
31	BiLSTM B2 1a - prediction on simulation #3104	54
32	Simulation #3104	54
33	BiLSTM B2 1a - prediction on simulation #17458	55
34	Simulation #17458	55
35	Training progress BiLSTM B2 1a	56
36	Training progress LSTM B2 1b	56

List of Tables

1	Generic offshore drill rig setup. Illustration: NORCE	15
2	Influx type probability	20
3	Features sets explored in this paper	31
4	Simulation set 1 summary	37
5	Simulation data and data set 3 summary	37
6	Simulation set 2 summary	38
7	Layer size RMSE on small training batch with feature set 1a	38
8	Feature set RMSE with single 100 node LSTM layer on batch 1	38
9	Network type RMSE with feature set 1a, batch 3	39
10	mini-batch and sequence length RMSE on Batch 2, feature set 5a, max Epoch 500	39
11	Best performing model accuracy after 5000 epochs of training	39
12	ER Diagram SQL DB	A - 1

Nomenclature

Abbreviations

ADP	Annulus discharge pressure
AI	Artificial Intelligence
BHP	Bottom Hole Pressure
BiLSTM	Bidirectional Long Short - Term Memory
CPU	Central Processing Unit
CSV	Comma Separated Values
DNN	Deep neural networks
ER	Entry Relationship
GPU	Graphics Processing Unit
LSTM	Long Short-Term Memory
ML	Machine Learning
NN	Neural Network
RAM	Random Access Memory
RDBMS	Relational Database Management System
RMSE	Root Mean Squared Error

RNN Recurrent neural networks

ROP Rate of Penetration

RPM Revolutions Per Minute

SPP Stand Pipe Pressure

Definitions

Annulus: The void between any piping, tubing or casing and the piping, tubing or casing immediately surrounding it.

Epoch: A NN training cycle through all the available training data

Features: One feature is one input value to a NN

Iterations: Models tried out during training of a NN network, differs from Epochs as a training cycle can try out many iterations of a network within one epoch

Symbols

U Pseudo random number between 0 and 1

1 Introduction

Oil and gas drilling is a large and prosperous industry with a history stretching back as far as to 347 AD [2]. Despite the need for a reduction in our hydrocarbon footprint, there is no sign hydrocarbon extraction will be obsolete in the near future [3], furthermore, the global production keeps increasing each year [4]. The number of active wells today is in the millions, daily production of oil exceeding 90 million barrels and the daily production of natural gas exceeding 30 trillion cubic meters. With this unprecedented production scale, even minor risks become likely, and despite technological advancements in safe drilling the majority of wells still rely on older and simpler technology.

During any drilling adventure, many risks can happen quickly and have large consequences. One common risk for all wells is an uncontrolled blowout, which can be caused by an influx of formation fluids (water, gas, oil or a combination of the three) into the wellbore, often termed a "Kick". If it is not detected and counteracted in an early phase, the unstable effect can cause severe financial loss, environmental contamination, and loss of human lives. As such [5] concludes that; "Their prevention is undoubtedly the most important task in any drilling venture".

Perhaps one of the most renowned kick related incidents is the Deepwater Horizon (DWH) oil spill in the Gulf of Mexico in 2010. In this incident, a sequence of safety mechanisms failed, but this sequence of damaging events was initialized by an undetected gas influx in the well [6]. As concluded by the incident report on the DWH accident, the undetected influx in the well is one of six direct causes for the accident. The accident cost 11 human lives, 4.9 million barrels of oil spill into the ocean and an estimated total cost to BP at around \$65 billion (USD).

To achieve risk reduction, the industry should not only try to reduce kick occurrences but also improve detection as countermeasures applied in early stages can severely limit the risk of an uncontrolled blowout by regaining control the well. And in the worst case give the crew of the rig ample time to plan and prepare for the blowout. Towards this goal, this project aims to explore early prediction and detection systems for kick incidences while drilling by the use of emerging techniques within artificial intelligence. Such a system will significantly reduce non-productive time in a drilling process and is the natural first step towards achieving autonomous automatic well control. The methods will be developed and tested on a high fidelity drilling simulator, OpenLab Drilling, developed by NORCE.

2 Theoretical Background

2.1 Drilling

Relative to the modern technological era drilling is well-established engineering field. But while the history of extracting oil stretches back as far as to 347 AD, the history of offshore drilling finds its origin in 1897 [7]. While these first offshore oil rigs were land rigs placed on wooden poles in shallow water, the first on-water drilling rigs were produced in the 1930s. From the start of these shallow offshore wells, there has been an increasing demand for petroleum products, in combination with the high prices the production has had a nearly exponential increase. This has pushed production to both deeper wells and more difficult environments. Increasing the complexity and risk associated with the drilling adventure. While there have been substantial technological advancements, especially the last three decades [5], with some rigs already being designed with this cutting edge technology today, most of the established drilling rigs and prospected rigs still rely on older technology. There are several reasons behind this, one of which is the difficult and expensive process in getting new technology approved for use in the industry, hampering the competition in the market. Another worth mentioning is that the drilling operation is performed partly by the oil company, the drilling contractor and the service company. This means that the overall process overview is not fully understood by each of the companies involved in the drilling operation. This applies specifically to the service companies since they typically deliver equipment and personnel trained to operate their equipment. Additionally, applying automation to some of the emerging fields would mean less personnel involved, therefore this would lead to reduced revenue for the service companies. However, with the competitive market, fluctuations in oil prices and the potential advancements in improving Health, Safety, and Environment (HSE), there is currently a renewed interest for innovation in the drilling industry.

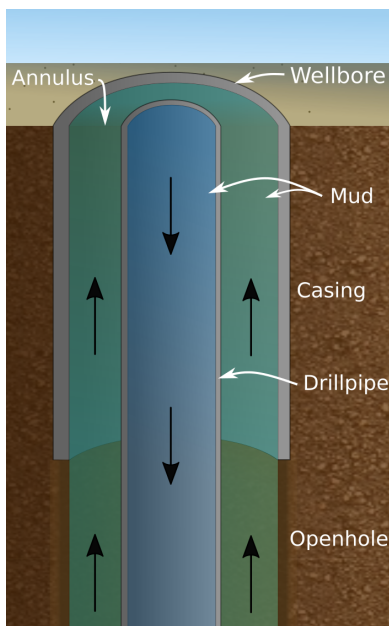


Figure 1: Illustration of well

Today's platforms, also called rigs, are large structures carrying both drilling equipment, living quarters and anything needed for personnel to stay for an extended period. Depending on the water depth the platform can either be mounted straight on the seabed or a floating structure, both can also have extensive subsea structures connected to the facility. So-called 'Rotary Drilling' is done by applying force on a rotating a drill string with a drill bit at the end. The drill string is fed through the top deck of the platform while drilling and extended with a new drill pipe when needed. A drill pipe is approximately 9m and is stored pre-assembled on the drill rig in sets of three, therefore an extension need to be added or removed for approximately every 27th meter when drilling or tripping. The cuttings released by the drill bit are carried up to the platform by mud cycled down through the inner drill string and up again in the annulus of the well. The mud serves several purposes as it also lubricates and cools the well.

Perhaps the most important safety function of the drilling fluid (mud) is controlling the well pressure to stay within the limits of the formations pore- and fracture- pressure as not to induce a kick, see fig 2. When the well pressure is reduced below the pore pressure of the formation can occur [5], this influx can consist of water, gas, oil or a combination off all. This can happen either because it is drilled into a formation with unexpected high pressure, or from the well pressure dropping below the pore pressure limit. Another troublesome factor is if the well pressure exceeds the fracture pressure of the formation, in this case, the mud can permeate the formation, this is termed mud loss. In the case of a total mud loss, the formation can be fractured in a way that larger quantities of mud are lost to the formation, potentially decreasing the hydrostatic wellhead, I.e the mud level will be below the top of the well. This can, in turn, reduce the pressure in the well below the pore pressure and induce a kick.

As illustrated in figure 2, these pressure limits do not conform to any predefined patterns but rather depend on the formation, as such it can be impossible for the hydrostatic pressure gradient to fit between the upper and lower formation limit for an extended depth range. To extend drilling depth casings, fig 6, are used as a barrier between the well flow and the formation, greatly increasing the pressure range of the well.

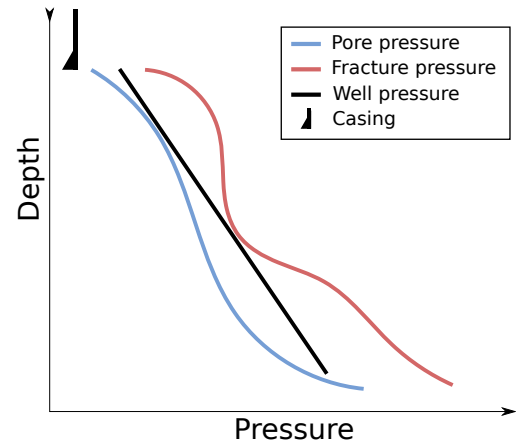


Figure 2: Formation pressure limits and hydro-static well pressure

Besides functioning as a formation barrier the casing is also used as an installation point for specialized equipment like the blowout preventer (BOP). The BOP is a multi-layer safety barrier mounted at the end the well casing on the seabed. As such it can cut off the well even if the pipeline between the seabed and the platform is damaged. Towards this purpose the BOP stack has a multi-tier redundancy system that allows for flow and pressure control as well as flow cutoff, this is to help regain control of a well where a kick has occurred. As a last resort, the BOP has pipe rams and shear rams designed to cut off both annular flow and cut a potential pipeline going through the BOP.

Keeping control of the well pressure is paramount to safe drilling. The well pressure gradient is controlled through the mud density, choke opening and the flow rate. While the mud density decides the wells hydrostatic pressure at rest, also affected by any cuttings or influx masses, it is slow to respond on new changes, as such the choke is used for more responsive adjustments. Controlling the pressure by the choke does however require there to be active flow through the well. Due to friction in the well, the flow-rate through the well also impacts the pressure. As such, there is always an increased chance of influx when adding or removing drill pipes, a process that requires the mud pump to be turned off. Due to the friction coefficients of the well, especially in the open hole, being unknown, making models of the well response has so far been beyond the reach of science. This is further complicated by the fact that there is often no live readout of the actual bottom hole pressure (BHP). Making fine control over the pressure in the well a difficult task.

The most common way to read bottom hole pressure today is a signal sent by pressure modu-

lation in the mud, typically giving a readout at 0.5 Hz when the mud pump is connected. The readouts are unavailable during tripping and other procedures where the hydrostatic head is not reaching the top deck of the rig. While newer technology, like drill by wire [8], offers much better bandwidth downhole and are emerging in the market. This is much more expensive and still not common among the rigs in operation.

While there are uncertainties in the well pressure throughout the wellbore, estimated lower and upper bounds do give a reasonable knowledge of the well pressure. To know what pressure to keep in the well formation, surveys are done. While there are no exact procedures for determining the formation pressure limits, it is generally done by a combination of seismic data, logged drilling data and/or live drilling data. All of which are prone to uncertainties in their results [5,9]. As such, the estimated formation pressure limits inherit these uncertainties, and one can not be sure that all abnormalities, like fractures or different pressure pockets, are detected in a planned well path.

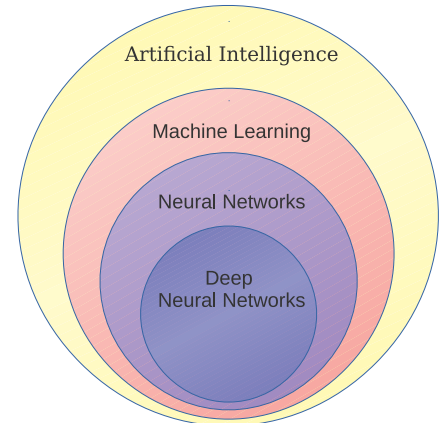
What is done when a kick occurs [10] The consequences of these uncertainties and others are that kicks do happen in the well. When an influx does occur it can, as previously mentioned, consist of multiple of substances like water, oil or gas, but also combination off them all. The result is different characteristics and risks involved in the incidents. While an oil or liquid influx will displace some volume during its influx, it will have a relatively stable ascent among the mud in the well. In general, the density of the influx fluid will, however, be lower than that of the mud, as such the casing pressure will increase. This risk is even higher when a gas kick occurs, this is due to the increased expansion in the gas volume when it rises towards the surface and decreasing pressure. A gas kick is especially dangerous when it reaches the riser, where it can do a rapid expansion and blow out through the top deck of the platform. The severity of these consequences is, therefore, dependant on both the influx type and volume.

While less severe kicks might be handled with uninterrupted operation, especially by modern MPD drilling rigs, larger kicks need to be bleed out of the well in a slow controlled manner to reduce risk. While there are several methods for controlling a kick, [5] breaks these down to two fundamental elements; firstly by displacing the mud with a heavier mud to stop the influx and secondly to safely circle out the kick fluid and/or gas from the well. On conventional drilling rigs this procedure, called well control, is a manual operation including sensor readings, calculations, and control performed by several members of the drilling crew. It involves control of the blow-out preventer (BOP), the rig pump and the well control choke, all located at different locations at the rig. The well control choke is adjusted manually to maintain a certain pressure in the well. This may be a difficult task due to large time-delays in the drilling process and the complex behavior of the multiphase flow. To ensure alertness in the crew and that proper measures are taken, early detection with few false alarms are important. Methods for kick detection will be presented in chapter 3.

In summary drilling for petroleum products offer great economic reward and valuable resources for society. The act of drilling is, however, a complex multidisciplinary task requiring considerable control of the well. As the demand pushes production to harder to reach reservoirs, and our awareness of the lasting environmental damages done by severe accidents increases, it is more important than ever to ensure safe drilling and increase our capacity to monitor and control downhole events.

2.2 Artificial intelligence

In the broadest terms, artificial intelligence is a family of tools, digital or not, that in some way solves complex tasks. While its a thriving field in research and development today, it is also a field with a history that can be traced back to ancient Greece. Due to its long history and popularity, there are many definitions of the word depending on which perspective it is interpreted from. For this paper, AI is defined as the superfamily of all tools that in some way solve complex tasks.



While much of the backbone of modern deep learning and neural networks were written in the 1900s [11], the required computational power and ability to both gather and handle the big data needed for training sufficiently complex models has not been readily available until fairly recently.

2.2.1 Machine learning

Machine learning is a field of computer science for pattern recognition and statistics. It is the scientific study of algorithms and statistical models that computer systems use to effectively perform a specific task without using explicit instructions, relying on models and inference instead. Machine learning algorithms build a mathematical model of sample data, known as “training data”, to make predictions or decisions without being explicitly programmed to perform the task [12]. Machine learning algorithms are used in the applications of email filtering, detection of network intruders, and computer vision, where it is unfeasible to develop an algorithm of specific instructions for performing the task. Machine learning is closely related to computational statistics, which focuses on making predictions using computers. As the complexity of the models have increased in recent times, it has become evident that the quality of the input data to machine learning models plays a significant impact on their performance [11]

2.2.2 Neural Networks

Neural networks are a type of machine learning that was devised to mimic the way a brain works. While there are devised many types of networks and nodes, it is in its simplest form a collection of simpler functions working together to achieve complex tasks as depicted in figure 3. The network is generally built with a set of input features, depicted in green, some hidden layers, in grey, and outputs. An input value is termed a feature and can contain a single value or a multidimensional array of data. The only limit to the number of features and the array size of each feature is

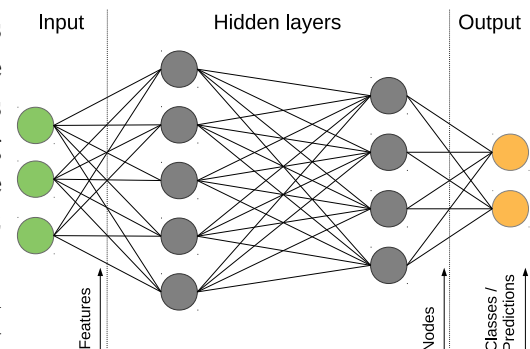


Figure 3: Neural network structure

the computational power needed. The hidden layers of a neural network are often referred to as the 'black box', and is where the neural network calculates the response of the features.

With zero hidden layers, the network can essentially just do linear regression. By adding hidden layers, the network can adapt to increasingly complex non-linear functions. The complexity of the function it can predict is further impacted by the number of nodes, also called neurons, in each layer. There seems to be no general agreement on the number of layers and neurons to use in a network as of yet, especially when there is more than one input feature. With only one input feature there are no apparent reasons for using more than one layer.

While there are many types of nodes that can be used in a neural network, each layer generally only consists of one type of node. One of the simplest types of nodes to be used is a fully connected node, eq 1. The fully connected node takes the value of each node or feature in the previous layer, X , and multiply it with an individual weight for each, W , and add a bias, b . It is also common to use some kind of activation function to keep the resulting value within a predefined range.

$$Z = W \cdot X + b \tag{1}$$

2.2.3 Deep learning

Deep learning is a subset of machine learning methods. Deep learning architectures such as deep neural networks, recurrent neural networks, and deep belief networks can have hundreds of millions of parameters [11, 13], allowing them to model complex functions such as nonlinear dynamics. Unlike many machine learning methods, they do not require a human expert to hand-engineer feature vectors from sensor data. Some deep learning models can, however, present particular challenges in physical robotic systems, where generating training data is generally expensive, and sub-optimal performance in training poses a danger in some applications. Yet, despite such challenges, researchers are finding creative alternatives, such as leveraging training data via digital manipulation, simulation and automating training to improve the performance of deep learning models and reduce the training time.

Compared with traditional neural networks, recurrent neural networks (RNNs) are known for making decisions by reasoning about previous events. The looping nature of RNNs allows information to persist so that not only the information from the previous time step and current time step model the prediction but also the information from more than one previous time steps. Some of the applications that can be successfully solved with RNN are language modeling, speech recognition, image captioning, and translation.

Depending on the application, it varies how much of the historical data is needed to be taken into account. Standard RNNs do not perform well when much context is needed. This is dubbed the long-term dependency problem.

LSTM: Considering the above issue with the standard RNNs, in this research, we utilize a special kind of RNN i.e. Long Short Term Memory (LSTM) networks. The selected approach is capable of learning long-term dependencies. Instead of the chain of repeating simple modules

having a single neural network layer in standard RNNs, modules in LSTM have a more advanced structure having four neural network layers.

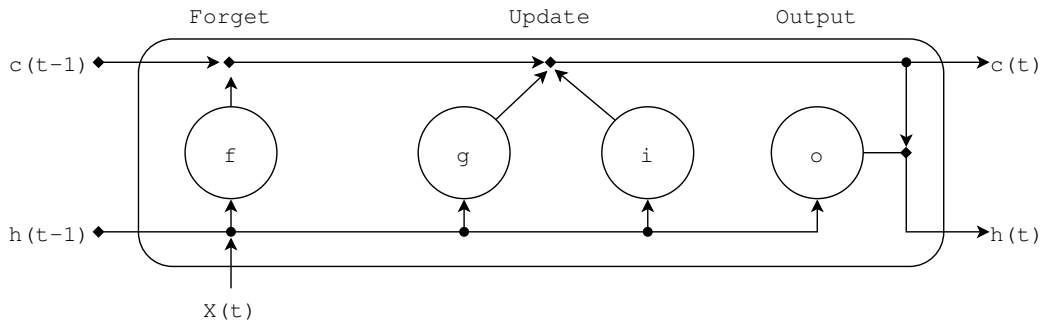


Figure 4: LSTM module [1]

These four layers in an LSTM module perform different tasks during the training phase. Three of them act as gates that optionally let information through and are made of a sigmoid neural net layer. Therefore the output of these gates is a value between 0 and 1 i.e. value 0 let nothing through and value 1 let everything through. First, the forget gate layer, f in Fig. 1, decides which information should be removed by looking at the current input, $x(t)$ to the module and the output from the previous module, $h(t-1)$. Then, the input gate layer (g) and the tanh layer (i) collectively decide which new information should be added to the existing knowledge, $c(t)$. Once we are done with the updating of information within the module, the output gate layer (o) decides what to output, basically a filtered version of the existing information.

BiLSTM: A Bidirectional Long Short-Term Memory is a version of LSTM module that is used to predict a full sequence of responses by looking both forward and backwards in the sequence for any given point. A BiLSTM layer is generally more accurate than a LSTM layer but does not perform as well at the end of the sequence as an LSTM. This generally makes it unsuitable for real-time predictions, however, the increased accuracy away from the edges of the series can make it a suitable tool if a short delay is not an issue.

2.2.4 Generalization and overfitting

When training a neural network of any kind the goal is often to create a model that generalizes to the value or event that is intended to be detected. As described in section 2.2.2, the number of layers and nodes in a network determines the capacity of the network, and thus its ability to find a good fit. If the network is too simple it will underfit to the training data, essentially this can be seen as the equivalent of trying to describe a polynomial function with a linear function, while you might align with a point or two on the line, most of the polynomial function will not be described by the linear function. In contrast, if the network is too complex the network can overfit to the training data. By overfitting, the network generates a function that describes the data points seen in the training set with an accuracy excluding similar values/events from previously unseen data [11]. This can again be seen as describing the points along a low-resolution sinus curve with a high-frequency sinus function, aligning perfectly to the points seen, but not with the function itself.

While it is possible to calculate the capacity of the network, this is a non-trivial task to match

up with the needed capacity for data-set. As such the conventional method is to use trial and error to determine a suitable capacity for the data seen in conjunction with suitable training time, as it is not uncommon to use a network with a higher than needed capacity and stop training when overfitting starts to occur. This is termed *early stopping*, and is a well-established method for generalizing the resulting model. Early stopping is done by using a validation set alongside the training set during training. For set intervals, during training, the prediction error is calculated by using the validation set. While both the training error and validation error are decreasing, the network is trained towards a generalized model. However, if the training error continues to be reduced while the validation error starts increasing this is a sign that the model is overfitting. As such the early stopping algorithm will stop training when this occurs.

While *early stopping* can counteract overfitting on a network with a given data-set, it cannot improve on the achievable generalization with a given model and data set. To further increase the network capacity and/or the generalization of the network a larger training set can be used. When using DNN's which inherently has a large capacity even more simple setups need large data sets to not overfit. Increasing the size of the training data is often referred to as the most efficient way to achieve a more generalized model [11].

2.2.5 Optimization algorithms

With the number of parameters to tune in a NN range from just a few to hundreds of millions, sophisticated methods of optimizing these parameters are essential for achieving the desired result. The applied optimization algorithm and its initial parameters can greatly affect both the end result and the training time. As such it is a field of active research with several promising methods. For this paper two well-established methods [11] have been evaluated.

SGD: Stochastic Gradient Descent (SGD) was introduced in the early ages of deep learning (cybernetics) [14, 15], and is probably one of the most used optimization algorithms. While gradient descent is a method of following the gradient of an entire training set downhill, SGD makes a significant time improvement by calculating the gradient by only a random selection of the training set, termed a mini-batch. Critical initial values for an SGD algorithm includes it's learning rate, learning rate reduction over time and in some variants the momentum [16]. [17] concludes on the effectiveness of SGD in training on large data sets. While [18] discusses the issues with setting the correct optimization parameters and the negative effect this can have on the resulting model.

Adam: Adam was introduced in 2015 as an adaptive learning rate optimization algorithm [19]. It is presented as a method that is robust to noise, computationally efficient and requires little tuning. By adaptive optimization of the parameters, it greatly reduces the need for trial and error in determining training parameters. While it comes with a default learning rate, this can be changed where needed. As with SGD, Adam uses mini-batches to increase training efficiency but does not offer early stopping.

2.2.6 Mini-Batch

Computing the error on a single model over the whole data-set is a computationally heavy task. In practice, the training set is therefore divided up in mini-batches with a predetermined size. There are no generally agreed upon rules for the mini-batch size most efficient for training. Several papers [11,20,21] do however argue for a relatively small mini-batch size. The definition of a small mini-batch size differs depending on the authors but ranges between 32-512 in most suggestions.

Both [20] and [21] attribute this improved performance to a larger mini-batch size optimizing to a sharper minima, which leads to a poorer generalization. Moreover, the finding in [21] supports a commonly held belief that the inherent noise in the results of a smaller mini-batch size is beneficial in the gradient estimation of the optimization algorithm.

2.2.7 Augmented learning

While increasing the training set often is the best way to generalize your model, acquiring enough data to do so is not always possible. This is especially true for drilling data. A well-known method for increasing the size of the data set already available is using data augmentation. This is a method where the already available data is in some way modified and augmented in one way or another while making sure the augmentation itself does not interfere with a realistic model, i.e rotating the number 6 180 degrees in a handwriting data sample set. A further example of this can be seen in image augmentation where the pictures are modified with random color adjustments, rotations, scaling and more. With a physics-based simulation, one needs to be especially careful that the augmentation does not interfere with the realistic results. However realistic drilling data is prone to noise and augmenting a training set with different noise and disturbance filters have been known to give better results [22]. With this method, the noiseless simulated data set could easily be doubled or tripled in size, while also being trained to filter realistic noise on the sensor data.

2.2.8 Combined learning on real and synthetic data

ML methods are often used to detect and evaluate uncommon occurrences in the real world. This does, however, pose a problem as the available training data might be sparse, as is the case when evaluating kick. While real data exists, it is unlikely that the available recorded data can be combined in such a way as to present a full generalized description of the incidents. Towards this end, [23] makes a convincing argument for combined learning with both real and synthetic data to improve the accuracy and robustness of a machine learning model, especially in the cases where ML methods are used as a tool with high-risk scenarios, like a kick. During the development of automatic power line inspection drones, [24] explained that using combined learning was an essential tool for achieving good results correctly identifying power lines in difficult conditions. He further explains that they, in general, expected a 25 % performance loss when moving from only synthetic data to real.

3 State of the Art

The conventional kick and loss indications are summarized in [5] as follows: abnormal variations of active pit volume, the difference between flow in and flow out, variations of standpipe pressure and annular discharge pressure, etc. It is widely accepted in the literature that flow measurements give the most rapid indication of a kick [25]. The flow-rate measurements are often quite noisy and subject to calibration problems. Several filtering methods have been used to extract more reliable parameters including low-pass filter [25] and a method based on Bayesian probability calculations [26].

3.1 Return Flow

Monitoring the return flow out of the well may also provide indications of both reservoir influx and lost circulation. In a stable well, the flow in and out of the well should be approximately the same over shorter time ranges when flow rates are unchanged and a change from this will indicate unstable conditions.

3.2 Detection of Wellbore Anomalies through Pressures

Another proposed method of detection of kick and loss, as well as other wellbore anomalies, is the use of standpipe pressure (SPP) and annulus discharge pressure (ADP). [27] The behavior of these pressures by themselves and in comparison to each other can help identify downhole problems. Pressure sensors are smaller and easier to install than Coriolis flow meters. For kicks and losses, the alarms are based on pressure change equivalents for total flow or a continuous total change in volume. Washout and plugging are detected based on changes in pressure. To reduce noise and make interpretation easier, the variance is normalized.

The method seems to compare well with the use of a Coriolis flow meter, with comparable results for the time used for detection, as well as the flow and volumes. The method also allows for the detection of anomalies with a shut-in well, which is not possible with a flow meter. Also, the method is not prone to problems due to plugging or proximity to vibration sources in the same way as the flow meters.

3.3 Downhole Pressure Measurements

Measurements of downhole pressures may also be used for kick detection. These measurements can be transmitted to surface by traditional mud pulse telemetry, but real-time measurements would then be limited to whenever the pumps are running. Data rate capabilities are limited, due to low bandwidth by mud pulse telemetry itself, and because other downhole data measurements are transmitted in the same way. A faster alternative is the wired drill pipe [8], which would also give measurements when not circulating. It is however also a lot more expensive.

3.4 Connection Flow-backs

Connected to the mud pit volumes are the flow-backs experienced during connections. During circulation, a certain amount of mud will be occupying the surface circulation system. When the pumps are shut off during a connection, this mud will flow back into the pits, increasing the pit level. Depending on the flow rate, the amount of flow-back should be more or less the same at each connection, and any changes may indicate changes downhole.

3.5 Gass kick detection

A gas kick alarm system is presented in [28] where the principle is to measure the propagation time of a pressure pulse through the well by using a sonic technique. A new drilling method was developed in [29,30] by using the concept of micro-flux control, which is based on detecting a loss or influx of fluids, and instantly adjusting the return flow and the bottom hole pressure to regain control of the well.

3.6 Automated Monitoring of Traditional Parameters

The simplest approach to automated kick detection is to monitor the pit level or mud flow rate in and out of the well, and raise an alarm when threshold values are exceeded. Automated systems for monitoring variables such as pit levels and flow out would be able to spot reservoir influx in the same way as humans do today. However, one of the challenges with computer-assisted decision making in drilling is that the active circulation system is a highly dynamic and complex system, and having alarms on simple rules would raise false alarms. The system needs to be able to understand what is going on and adapt to this information.

Recent experience indicates that to optimize the drilling operation the entire drilling system, not just the mechanics or software, needs to be designed from a control system point of view [31–36]. A difficult and expensive task for drilling rigs already in operation. Furthermore, model-based detection in a well can be challenging, both due to the very complex dynamics of the multiphase flow consisting of drilling mud, cuttings, reservoir fluids and modeling of subsurface conditions e.g pressure limits and formation friction.

3.7 Detection Algorithms for MPD

While MDP detection will not be evaluated directly in this study, several drilling operations have been performed successfully using Managed Pressure Drilling (MPD) techniques. As such it is of interest to evaluate the detection methods for this paper. MPD is relatively a new drilling process that allows greater, more precise control of the bottom hole pressure in a wellbore in [37]. The detection of kick and loss in MPD has received a lot of attention in [27, 38, 39], where the method of monitoring the variations of the standpipe and annular discharge pressures was developed to identify influx and loss during MPD.

In [40], the detection of gains and losses was based on the deviations of measured and the expected flow-rate out which depends on accurate hydraulic models. Model-based gain and loss

detection method were developed in [41] where a transient hydraulic model is used for downhole and surface equipment effects on the pit volume variations. A low-order model was developed for MPD in [34, 42] based on the conservation of mass and momentum balance. Research on kick detection and control based on the low-order model has been recently in the articles [33, 35, 43–45] during MPD.

3.8 AI in kick detection

Machine learning methods have been investigated for kick detection recently [46, 47]. They also provide a good overview of the role of machine learning and a summary of state-of-the-art, i.e. artificial intelligence, for drilling applications. In [48], machine learning algorithms were applied for the detection of well control events for a case study. This study explored the use of using machine learning to create an adaptive alarm threshold on flow out and pit volume with the flow in, bit depth and well depth as features with promising results in reducing false alarms. In [49], a case study in Iranian oil fields was conducted for early kick detection using real-time data analysis with a dynamic neural network trained with a range of different sensors as features. This study also explored different data frequencies, concluding that 15Hz gave the best result for their NN. Inconsistencies and missing tables in the publication make it hard to conclude on the network model and feature sets used to achieve these results.

Companies like Shell and Equinor are also currently working with artificial intelligence for early kick detection [50]. In this method, ML is used to train a model to predict the expected Flow Out, SPP, and mud pit gain/loss. A kick or loss alarm is then raised for the human controller if there is too large of a deviation from the expected value and the measured value. A benefit of this solution is that it is human-centered, meaning it tells the human control which values it expected a different result. This allows the controller to recheck and evaluate the model's prediction.

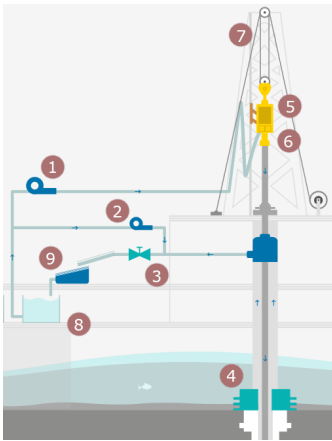
4 Drilling Simulation

4.1 Simulator

The drilling data for this paper was generated by the OpenLab Drilling [51] simulator delivered by NORCE. OpenLab Drilling is a high fidelity simulator that simulates the response of a well with great accuracy through differential equations. The simulator is based on the following published methods [52–57] and offers some generic well and ring templates to use for simulations. For this paper, all work is based on the supplied "Generic offshore" rig and the "InclinedWell 2500m" template, with modification applied as specified through this chapter. As this simulator is built as a versatile tool for training and research within drilling it offers support for modifying a large variety of parameters. Many of which have direct or indirect effects on the influx or loss in the well. This chapter will give an overview of the simulators key parameters and basic functionality in regards to drilling and influx simulation.

4.1.1 Rig, drill string, and wellpath

Rig parameters: The rig parameters in the simulation define the operational limits and characteristics of the simulations. For this study, the predefined "Generic offshore" rig was used for all simulations. The seen in table 1 are the rig parameters setup with this rig. For this rig, a few simplifications are done on the system. These simplifications include a main mud tank with infinite capacity and a shaker with zero loss, neither of which is of interest when simulating for the proposed methods in detecting kick.



Main Pump¹ - Flow rate acceleration	200 $\frac{l}{mins}$	MPD pump² - Flow rate acceleration	200 $\frac{l}{mins}$
MPD Choke³ - Change rate	20 %/s	BOP Choke⁴ - Change rate	20 %/s
Travelling block⁵ - Weight	20 ton	Top drive⁶ - Rotation acceleration	6 rpm/s
Drawworks⁷ - Top string acceleration	0.05 m/s^2	Main Tank⁸ - Tank Volume	∞m^3
Shaker⁹ - Mud loss proportion	0 %	Reserve Tank - Tank Volume	0 m^3

Table 1: Generic offshore drill rig setup. Illustration: NORCE

Drillstring: The simulator includes a library of different drill pipes and bottom hole assembly components. Of interest for this study is the capability to variate the drill pipe inner and outer parameters to change the volumetric displacement when tripping.



Figure 5: Generic drillpipe. Illustrations: NORCE

Wellpath: The well path can be completely customized on a meter by meter basis. Changes in the well path will affect the difference between the measured depth (MD) and true vertical depth (TVD). Furthermore, the difference in the well inclination will create a nonlinear relationship between the MD and the well pressure. As the models in this paper have been trained on MD, the single predefined well path seen in figure 6 has been used.

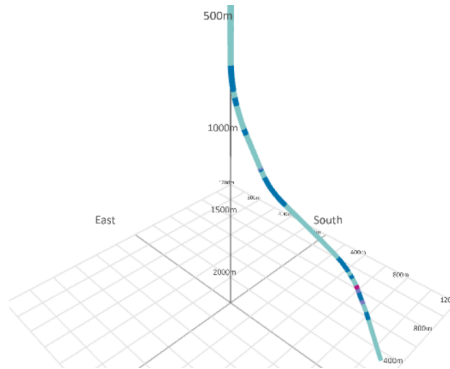


Figure 6: Well path of InclinedWell 2500m. Illustrations: NORCE

4.1.2 Geology & Casing

As described in chapter 2.1, the occurrence of kicks is highly reliant on the pressure and strength characteristics of the formation. To simulate the full geological profile Openlab uses a combination of formation pressure profiles, thermal profiles, and formation strength profiles. The pressure profile describes the pore pressure and the fracture pressure of the formation as seen in figure 7a. As the temperature is important for several aspects of the well dynamic including the well pressure a full thermal profile of the well is defined as seen in figure 7b. In addition, the formation strength, I.e, the pressure difference needed for a fracture, can also be defined, figure 7c. To create different influxes characteristics in the geological profile anyone can be modified. For this paper only the pressure profile has been modified, with this alone one can create a great variety of cases and limit the number of cases to select from. Towards that end the default thermal- and formation strength- profile where used.

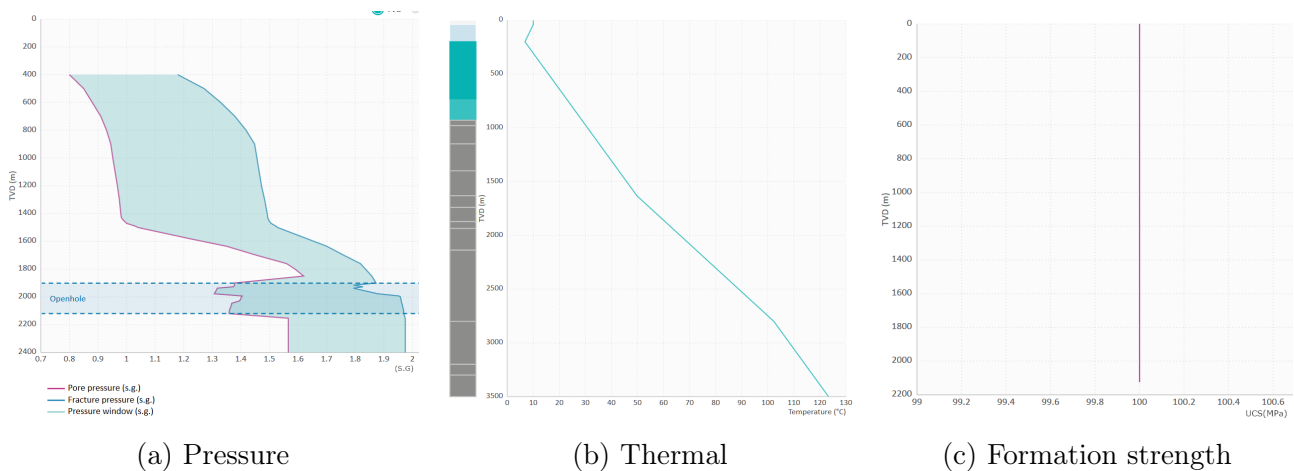


Figure 7: Default geological profiles used by OpenLab. Illustrations: NORCE

Just as with a real drilling operation casings are used to shield the well from the formation pressure. Each casing segments depth, diameter, and thickness can be specified. In these simulations, no attempts at influx due to casing defects have been done, as such the open hole has been the deciding factor for where the influx can occur. Simulations have been done with a 300m MD open-hole, ranging from 2200m MD to 2500m MD. This segment is outlined in figure 7a measured in TVD.

4.1.3 Drilling fluid (mud)

The drilling fluids in OpenLab can be changed with a high degree of freedom, both in terms of fluid mix and density as seen in figure 8. Editing the fluid allows for even more specific changes as gel strength over time, the oil density at different pressures and temperature zones and the fluid rheology. A reserve fluid can also be designed for the simulations, this can among others be used as a heavy mud when simulating and controlling influx scenarios. As control has not been studied the reserve fluid has not been used in this study. All fluid densities used are based on OpenLab’s predefined ‘Generic obm 1’ fluid, where mass and volume fractions are automatically adjusted to conform to a desired drilling fluid density.

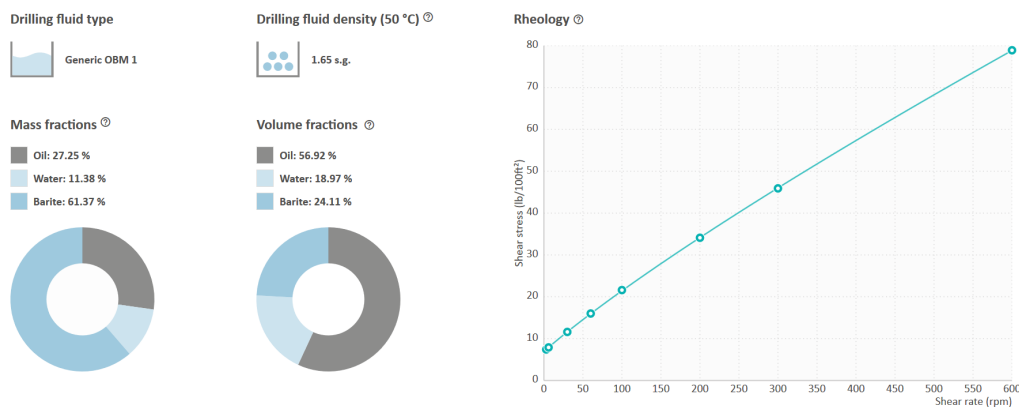


Figure 8: Drilling fluid interface. Illustration: NORCE

4.1.4 Influx & mud loss

OpenLab can simulate well operations with both artificial and pressure based influx. It is however limited to simulate the influx only with methane gas. While this reduces some of the complexity one would see in a real well, it is a worst case scenario of particular interest. During simulations, the amount of influx is measured by the total mass (kg) which has been injected or penetrated the well at every time step.

During the artificial influx simulations, a pre-defined influx is injected into the well at a pre-determined depth, rate, and total influx mass. Being independent of the drilling operations these cases should only be detectable from the well response and not as a consequence of how the well is operated.

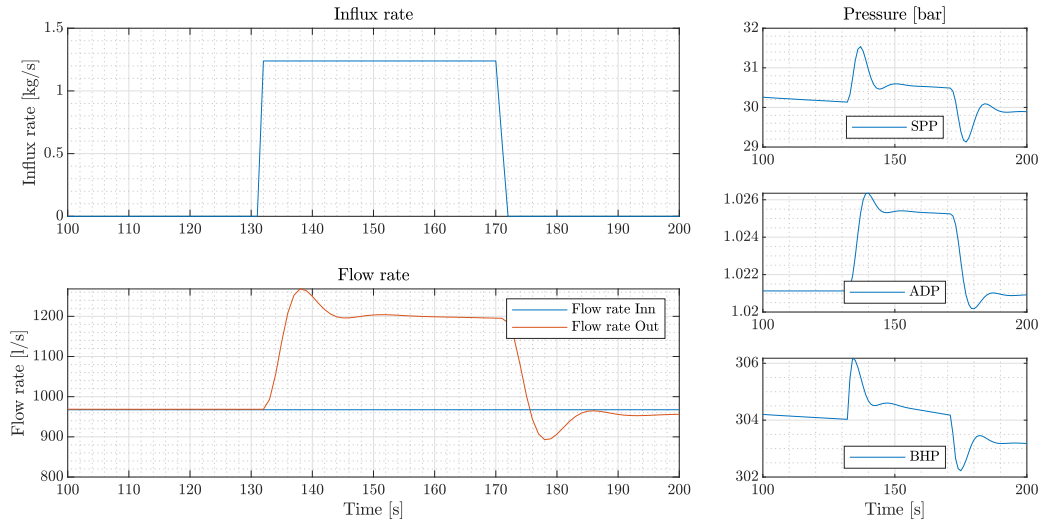


Figure 9: Influx injection example

Figure 9 shows an example of a artificial influx, and the flow and pressure responses of the well. In this case both flow and pressure where at a steady state when the influx occurred, the resulting effect on all measurements are easily observable.

Geopressure based influx or loss is based on a near-well formation flow model that calculates the flux between the well and the formation. The influx or mud loss is determined by the pressure difference between well and formation (see 4.2.2), the permeability, the porosity, and the density of the drilling fluid. This makes it difficult to reliably simulate a given realistic influx without introducing clear engineered operation patterns for a network to pick up on. To counteract this several different formation models, initial mud density and flow patterns were used and run dynamically using randomized patterns. With the large sample size, geopressure based influx inevitably occurred. Figure 10 shows an example of a geopressure based influx

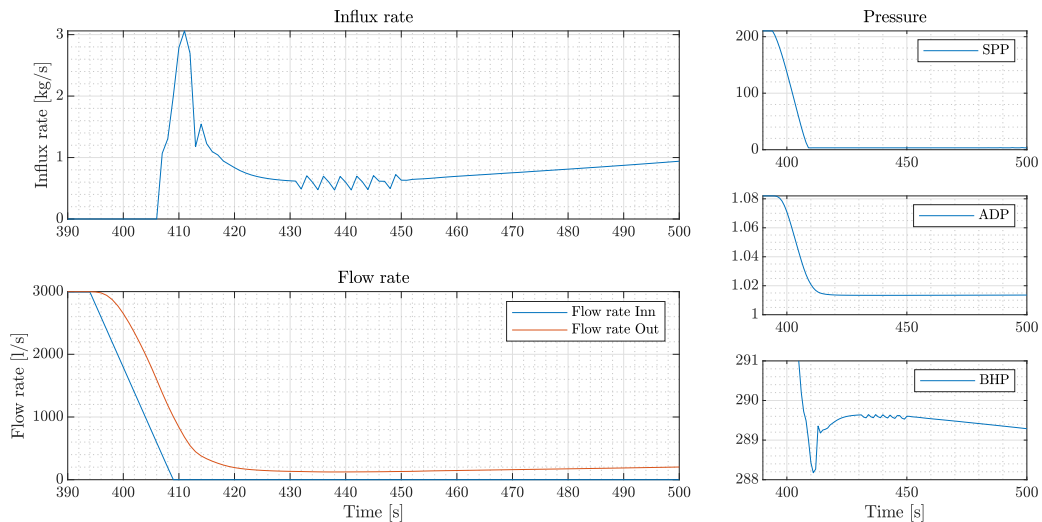


Figure 10: Geopressure based influx example

that occurred during a flow shutdown in the well. As with the injected influx in figure 9 the difference in flow rate is easily detectable. However, due to compressible in the drilling fluid, it

is difficult to conclude on an influx or its size based on the difference in flow in and out during the mud pump shutdown. While the SPP and ADP pressures both are too stable to make any visual observations of an influx the BHP can be perceived with an inverted correlation to the influx rate chart when the y-axis is scaled down.

4.1.5 Limitations

As the simulated responses are calculated there is no noise or other data artifacts that can be expected in a real drilling environment. Noise filtering is, however, an advanced field of study, most of which comes at the cost of lag or data frequency.

The simulations were controlled and the resulting data gathered by an algorithm written in MATLAB described later in this chapter. On average the simulations could run successfully at $\sim 5\times$ real time at 1Hz, with up to 10 simulations running in parallel. Even with the up to $\sim 50\times$ time improvement, simulating enough data for a DNN to be trained was a time-consuming process, especially if the data step frequency were increased. To allow for enough training data to be generated, and to mitigate the simplification of noiseless data a frequency of 1Hz was decided upon. While this is much lower than the data rate of top deck sensors it is still faster than the data rate from most downhole sensors. The conventional data rate here is 0.5Hz when there is an active flow.

As discussed in chapter 2 NN can be sensitive to overfitting. Two key factors in generalizing the network are ensuring a good distribution of the data, and to use data sets for the training of sufficient size, while still allowing for a representative size to be reserved for testing. Using a versatile simulator allows for making a variety of cases. When designing simulation setups one should, however, be wary of introducing engineered patterns, as these can be easier for the ML model to pick up on than the real pattern. This was a key design question when designing the simulation algorithms described below.

With the influx consisting of methane gas, the loss being composed of drilling fluid and both being measures in the mass gained or lost. There is a nonlinear relationship between influx and loss in the volumetric measurement of the event due to their different densities. At the current build of OpenLab, it is a non-trivial task to acquire the volumetric change or parameters needed to calculate this value. As such this data has been unavailable for this paper and rather the change in mass has been calculated. With the use of different mud densities when simulating loss this will greatly increase the mass rate estimation complexity.

4.2 Simulation setup

4.2.1 Influx simulation

When generating artificial influx simulations the parameters were set by use of non-uniform distributions of random numbers to generate a variety of cases favoring characteristics leading to a lower influx rate and mass at a deeper depth as seen in eq 2-4. The increased chance of smaller and slower artificial influxes was done to supplement the data sets as geothermal

influxes tended to be both larger and faster.

$$I_{Rate} = U^2 * I_{max} \quad (2)$$

$$M_{TotalMass} = U^2 * M_{max} \quad (3)$$

$$D_{Influx} = D_{WellDepth} - (D_{WellDepth} - D_{OpenHole}) * U^2 \quad (4)$$

Where:

I_{Rate} : Influx rate in kg/s
 $M_{TotalMass}$: Total influx mass in kg
 D : Measured depth of in meters

To generalize the model, both artificial and geopressure based influx were included in the data sets. Furthermore, each simulation had a probability of being unable to produce influx or loss regardless of the operation. This was done to increase the probability of the ML models to register an actual influx and not just the patterns potentially leading to an influx in the simulated cases. Each simulation was initialized with a probability of either manual influx, blocked or geopressure based influx, with the probabilities seen in table 2. Note, even though geopressure based influx is allowed, the simulated response decides if an influx or loss occurs.

Table 2: Influx type probability

Influx	Probability
Geopressure	64 %
Artificial	16 %
Blocked	20 %

4.2.2 Geology and mud density

The geological profile determines the location of the kick and the influx- / loss- rate in the simulation. To help generalize the model five different pressure profiles were designed. The geopressure properties of the profiles were not changed. Fig. 11 represents an example of the profiles used. Variations here included peaks in the fracture area to decrease the exposed areas and shifts in the Specific gravity (SG) range.

To initialize the well in different pressure zones of the geological profile, each profile was used in several cases with a different initial mud density in the well. With these variations, a total of 26 different initial wells were used, and the simulation algorithm randomly selected one at the start of each simulation. To increase the chance of influx based on lower annulus pressure than the geopressure profile, profiles with an increased chance of influx were represented more often. 14 of the 26 profiles produced geopressure based influx in the final data set. All were represented with artificial influx.

4.2.3 Flow Rate

The mud pump (flow in) rate was varied through the simulations both to teach the network the response of a well during operations and to induce geopressure based influx/mud loss from

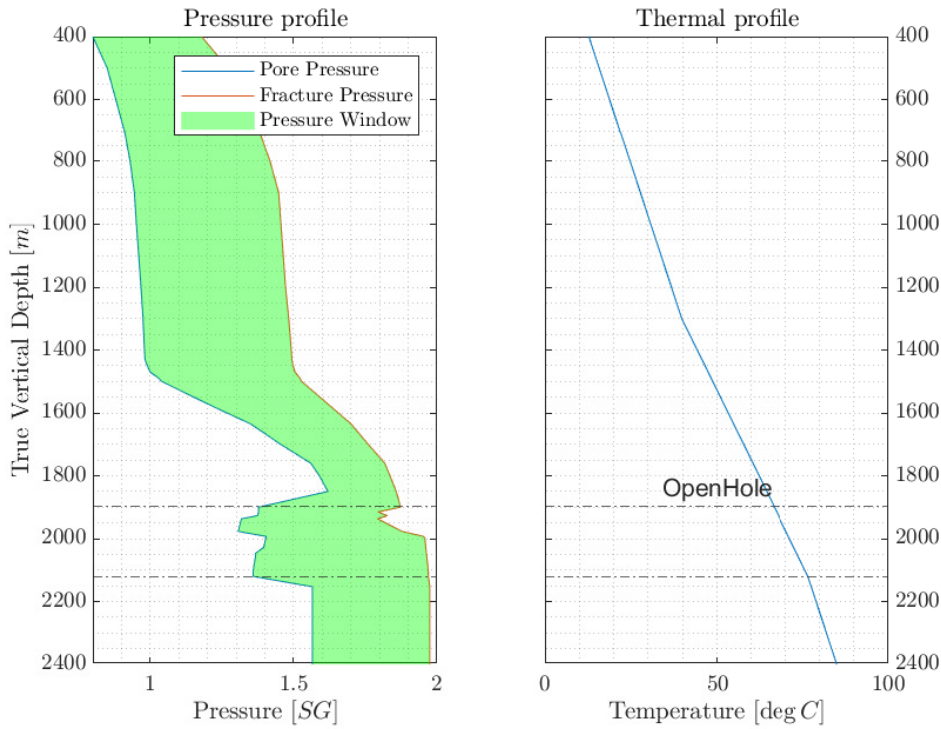


Figure 11: Example of geological profile

the resulting pressure changes in the well. The initial maximum flow rate of each simulation was randomly selected for each simulation, favoring a higher flow rate, by eq 5. This was done to increase the pool of actively driven wells in the data set. Furthermore, a variety of flow patterns were designed as a scalar on Q_{flow} to operate the flow during a simulation, Fig. 12. The design parameters of each pattern were seeded by a random value, ex number of periods and amplitude in the sin curve.

$$Q_{flow} = (1 - U)^2 \cdot Q_{max} \tag{5}$$

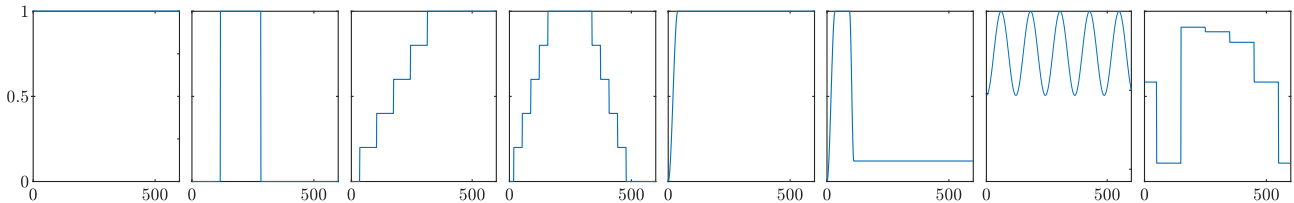


Figure 12: Examples of randomly seeded flow profiles

4.2.4 Choke opening

The choke opening is the exit valve of the main well line and has a direct effect on the amplitude of the ADP response of the well and is used to control both the annulus discharge rate and well

pressure. As active well control is outside the scope of this paper, a static choke opening has been used during simulations. The choke opening does, however, have an effect on the ADP and with the increased well pressure it can help induce mud loss. For the algorithm to understand the different ADP responses and to include loss cases the choke opening was changed between simulation. To opening, the value was randomly initialized by eq 6. Heavily favoring a large opening but allowing for some simulations to be run with a restricted choke to induce mud loss and further vary the data set.

$$C_{opening} = 0.98 \cdot (1 - U^5) \quad (6)$$

4.2.5 Tripping & Drilling

Both tripping and drilling pose an increased risk of influx or loss. As such an option for generation data-sets with this increased complexity was designed to test and train models. During these scenarios, the drilling control needs to be run manually in OpenLab. To generate larger data sets with these scenarios the simulation script has to operate the controllers. This includes setting the desired ROP, Surface RPM and top of string velocity. When running operation along with the depth, OpenLab automatically simulates connection-/disconnecting pipe sets.

The simulation script first chose which scenario to run, this was done independently of the choices done in section 4.2.1, with a 30% chance of tripping and 40% chance of drilling. The increased chance of drilling scenarios compared to tripping was to compensate for the lower string velocity in these cases and allow for more of the sets to include pipe connections.

4.2.6 Top of string position, string velocity ROP and Surface RPM

Top of string position, string velocity ROP and Surface RPM where set according to the scenario simulated. If neither drilling nor tripping was simulated, string velocity, ROP and surface RPM was set to zero, while Top of string position was set to a random valid position.

When simulating tripping the top of the string position would be set by favoring a short distance to travel before the disconnection of pipes needed to be done. To prevent this from happening at the same time in every simulation a random number seeded a function making a nonuniform distribution favoring a large top of sting position, as seen in eq 7.

$$L_{TopOfString} = \begin{cases} L_{min} + U^3 \cdot (L_{max} - L_{min}), & \neg tripping \wedge drilling \\ L_{min} + (1 - U^3) \cdot (L_{max} - L_{min}), & tripping \wedge \neg drilling \\ L_{min} + U \cdot (L_{max} - L_{min}), & \neg tripping \wedge \neg drilling \end{cases} \quad (7)$$

Furthermore, depending on if it is either drilling, tripping or keeping the bit at rest, the top of string velocity is set according to eq 8. While the ROP will override the top of string velocity while drilling it still needs to be initialized at a positive velocity (moving it further into the

hole) for the bit to achieve contact with the ground, even if its initialized at bottom of the hole. While moving upwards the velocity gets a negative value, favoring higher speed to increase the pressure below the bit and as such the chance of an influx.

$$V_{TopOfString} = \begin{cases} U^3 \cdot V_{Max}, & \neg tipping \wedge drilling \\ (1 - U^2) \cdot V_{Max}, & tipping \wedge \neg drilling \\ 0, & \neg tipping \wedge \neg drilling \end{cases} \quad (8)$$

During drilling scenarios the desired rate of penetration (ROP) where set by eq 9. With a uniform distribution slightly favoring a higher ROP to increase the chance of drill pipe occurrence during drilling

$$ROP = \begin{cases} ROP_{max} \cdot (1 - U^2), & drilling \\ 0, & \neg drilling \end{cases} \quad (9)$$

For drilling simulations, the RPM of the string at the surface had to be manually set. If this was set to low the desired ROP would not be reached as the actual ROP is the full simulated response of the drilling scenario. As the mechanics of the drilling were not of interest in this study, and the simulator offered no way to automatically control this value to reach the desired ROP, the surface RPM would decide the lower ROP bound and the desired ROP would decide the upper bound. To allow for a reasonable chance of a high drilling speed but still keep some variation in the surface RPM this speed was set by a two-thirds contribution from the ROP and the rest by uniform distribution, as seen in eq 10.

$$\omega_{surface} = \begin{cases} \frac{\omega_{max} \cdot 2}{3} \cdot (ROP) + \frac{\omega_{max}}{3} \cdot U, & drilling \\ 0, & \neg drilling \end{cases} \quad (10)$$

4.2.7 Drill bit depth

For initializing the drill bit depth the nonuniform distribution was designed to favor maximum depth. When a drilling scenario was initialized the drill bit was always placed at maximum depth.

$$MD_{Drillbit} = \begin{cases} MD_{Well\ Depth} - U^4 \cdot (MD_{Well\ Depth} - MD_{Openhole}), & \neg drilling \\ MD_{Well\ Depth}, & drilling \end{cases} \quad (11)$$

5 Data handling and preprocessing

This chapter will discuss the method used for storing, labeling and preprocessing the data for use with the Artificial Intelligence training methods and to ensure all models are trained and tested on equal grounds for a fair comparison between them. It will also discuss the method applied to ensure data consistency to reliably follow the simulation source of each generated data-point both as a tool to debug models and simulations but also to ensure consistency in this thesis.

5.1 Data storage system

5.1.1 Selection criteria

To find a suitable method for storing and handling the data for this project a few key parameters were identified to be:

Capacity:

For training, the more advanced model's large quantities of data were used. The storage method thus needed to store by an efficient and easily scalable method.

Relationships:

Through the development and production process, multiple methods and several different simulations conditions will be tested, many of which are generated automatically. For different reasons, specific types of simulations will at times be needed based on their setup conditions or run time results. These reasons are hard, if not impossible to predict beforehand, however, while a simulation is a time consuming to run, storage capacity is not a realistic limit with this type of data. To best design, a system with minimum loss of simulation data as many relational data points as reasonably possible will be stored with each simulation.

Speed:

The data generated will throughout the entire thesis be frequently accessed and searched both for result generation and for model training. For this, the data should be easily accessible and searchable at a reasonable speed.

Cross-platform compatibility:

To be able to run multiple simulations in parallel over an extended time Linux servers with a high CPU-core count where utilized, most designs were done on a windows machine and training of the more advanced models was done on specialized AI Research servers running Linux with a high GPU core count (NVIDIA DGX-1). Due to the use of several computers and servers in this process, a cross-platform compatible, centralized and redundant storage system was preferred.

Results*:

To increase the efficiency of result gathering and model comparisons, a system that could simultaneously store simulation data, relational data, resulting models, their reliability's and predictions was also seen as a bonus.

5.1.2 System selection

Data storage systems are a large industry, with active research and specialized file-types and software for a wide variety of purposes. As an optimized storage solution for AI research on drilling data is not within the scope of this paper, a widely supported and openly available solution was desired. The requirements described chapter 5.1.1 equals some of the key design parameters of a web database. As such, a well established relational database management system (RDBMS), MySQL, for web platforms was selected and deployed on the simulation server.

The deployed MySQL solution is robust and easily scale the number of data rows (in this case number of simulations). It also supports backups and connections from other computers through Java Database Connectivity (JDBC), making it well suited for cross-platform development. Compared to other database solutions this system is not flexible in the number of columns in each table (in this case the number of data headers and sensors being stored). As such the database design needed to be done in a way that does not exclude data points that might be usable at a later point. Moreover, to properly optimize search and storage size, the database has strongly defined data types for each column in the tables, decided upon on design time.

5.2 Database Design

The design goal of the database schema was a system where each data point from a simulation or model prediction could be traced back with all relevant data to the original code that created it, to ease development of simulation algorithms that could run unsupervised for hours, an event and error log was also included in the design. For reference, the full ER diagram can be found in Appendix A.1

5.2.1 Log

The log in the database was used to track all simulation batch runs and the individual simulations for debugging. To account for unplanned needs it was also designed to be usable with other error messages during development. By automatic increment, each entry was given an integer as the primary key, called idLog. By this value, each entry is uniquely identifiable and the error table can be joined with other tables through this relationship, see Sim and Run (5.2.2). Due to the database being accessed simultaneously by a multiple of parallel processes while running simulations a predefined Universal Unique ID (UUID) was used to help locate the auto-generated



Log	
idLog	INT(11)
event	VARCHAR(45)
startTime	DATETIME
endTime	DATETIME
msg	TEXT
errors	INT(11)
errorMsg	MEDIUMTEXT
UUID	VARCHAR(14)

idLog. While the UUID is possible to use as the relationship between the two entries, it is more efficient to only keep one index and search an integer number instead of a 14 character string of number. The event field was implemented to easily sort out the desired event to examine, start- and end-time (where applicable) helped to evaluate code and simulation efficiency. The msg field where used in both simulations and simulation batch runs to keep track of initializing

settings to help debug should any error occur that did not allow the code to complete. Errors and error message were used to count the number of occurred errors and the relating message respectfully.





5.2.2 Run - A batch of simulations

As described in chapter 4, the simulations were run in batches of different sizes. To allow the database to account for these variations in batch settings and the reproduction of the batch run a table was designed to hold the information of each run and its key MATLAB scripts and function, as read at run time. Each run was automatically given a unique key, 'idRun', by the same procedure as described in section 5.2.1, and connected to a corresponding log entry to track the event. 'name' and 'date' was given to each simulation to easily distinguish tests and different runs from the table alone. 'simulations' contains the goal count of simulations, but do not account for unfinished or crashed simulations, as such this has to be seen in accordance with the error log. As step time and amount of steps in each simulation is a key parameter that is not trivial to mix up in different NN models these were also clearly labeled on each run. To easily sort out run's based on drilling, tripping and/or active-flow has been allowed these were stored in a TINYINT(4-bit) value as a 0 or 1 boolean value. The script running the batch of parallel simulations and the simulation function itself was stored respectfully in 'minerFile' and 'simulationFile' as plain text. The field description allowed for additional information to be registered on each run.

Run	
	idRun INT(11)
	idLog INT(11)
	name VARCHAR(45)
	date DATETIME
	simulations INT(11)
	steps INT(11)
	stepTime DOUBLE
	tripping TINYINT(4)
	drilling TINYINT(4)
	activeFlow TINYINT(4)
	minerFile TEXT
	simulationFile TEXT
	description TEXT

5.2.3 Sim - Simulation settings table

Key information and initialization settings for each simulation were stored in the 'Sim' table. With initialization settings represented on the right side in the table and other meta-data represented on the left. Like with the above mention tables, each simulation entry was given a unique id to be uniquely referenced and joined with the corresponding simulation data. Furthermore each simulation

Sim		
	idSim INT(10)	ConfigurationName VARCHAR(45)
	idRun INT(11)	SimulationName VARCHAR(45)
	runNr INT(11)	InitialBitDepth DOUBLE
	idLog INT(11)	UseReservoirModel TINYINT(4)
	idCase INT(11)	ManualReservoirMode TINYINT(4)
	totalInflux DOUBLE	ManualInfluxLossMassRate DOUBLE
	totalLOSS DOUBLE	ManualInfluxLossTotalMass DOUBLE
	flowFun VARCHAR(45)	ManualInfluxLossMD DOUBLE
	tripping TINYINT(4)	TopOfStringPosition INT(4)
	drilling TINYINT(4)	ManualInfluxLossMD DOUBLE
	fileName VARCHAR(45)	UseTransientMechanicalModel TINYINT(4)
		stepTime DOUBLE

was defined by being part of a batch by its 'idRun' and have a corresponding log entry, connected by 'idLog'. As described in section 4 several different geological profiles and initial

densities were used for the simulations. Each of these cases were represented in the database Case table (sec 5.2.5) and is joined by the unique id in 'idLog'. 'totalInflux' and 'totalLoss' was stored on each simulation to easily query simulations based on these values. By design these were only filled in at the end of a successful simulation where no errors occurred, as such a NULL in one or both of these were used to separate unsuccessful simulations without the need for joining the log entry. The fileName was used for redundant storage to .csv file in case the database connection should drop during simulation.

5.2.4 Data - Simulated drilling data

All the simulated drilling data were stored together in the Data table, connected to the individual simulations through the 'id-Sim' field. Although 'idSim' and 'step' would create a unique addressable location for each entry, multiple primary keys proved problematic in the deployment of this database, therefore each data entry was given a unique key. To increase search performance in this table indexing was also done on 'idSim' and 'step'. As storage capacity was abundant all data points were stored for each simulation, this allowed for increased flexibility in later model design

Data	
idData INT(10)	step INT(10)
idSim INT(10)	flowIn DOUBLE
flowOut DOUBLE	flowBack DOUBLE
pressureSPP DOUBLE	pressureBHP DOUBLE
pressureBit DOUBLE	pressureADP DOUBLE
chokeOpening DOUBLE	depth DOUBLE
depthBit DOUBLE	surfaceRPM DOUBLE
stringVelocity DOUBLE	ROP DOUBLE
densityIn DOUBLE	influxMass DOUBLE
influxRate DOUBLE	mudLoss DOUBLE
lossRate DOUBLE	changeRate DOUBLE

based on the same simulation sets. As influx-, loss- and change rate were not provided by the simulator these were calculated from total influx- and mud-Mass at run-time.

5.2.5 Case list and Fracture profiles

For dynamic selection during batch simulations, a database copy of all cases generated on the OpenLab Drilling simulator was made, with key parameters stored for easy sorting. As several cases used the same fracture profiles these were also represented in a separate table connected by the 'idFractureProfile' variable.

Case	FractureProfile
idCase INT(10)	idFP INT(10)
idFP INT(10)	name VARCHAR(45)
name VARCHAR(45)	description TINYTEXT
depth INT(11)	maxDepth INT(11)
openHole INT(11)	
density DOUBLE	
description TEXT	

*FP: FractureProfile

5.2.6 TrainingSet, SimUse and Use

To ensure consistent use of simulations when training, validating and testing ML models, the training set structures were made as database relationships. This

TrainingSet	SimUse	Use
⚡idTrainingSet INT(10) name VARCHAR(45) pTrain DOUBLE pVal DOUBLE pTest DOUBLE description TEXT	⚡idSimUse INT(10) ⚡idSim INT(10) ⚡idSet INT(10) ⚡idUse INT(10)	⚡idUse INT(10) name VARCHAR(45) description VARCHAR(45)

allowed several sets to be generated based on different parameters, furthermore the use of each simulation is specified by a database relationship and does not duplicate sets being stored in different files or folders, saving storage space and increasing the data integrity. The training set parameters were stored in the table 'TrainingSet', where the fractional size of Training, Validation and Test data were stored to 'pTrain', 'pVal' and 'pTest' respectively. Furthermore it was given a unique id, name and description. To distinguish Training, Validation and Testing data these were given a numerical value and optional description in the 'Use' table by its id. This was done as an integer value is preferable to use while programming, both in terms of storing the relation and being more efficient to check for equality when handling long lists of simulations. With this design, each simulation can be part of many training sets, in each set its part of it can only be connected to one 'Use'. This describes a many to many relationship, which is unsupported by the MySQL database being used. To circumvent this a joining table is used, this can hold a one to many relationship with the 'TrainingSet', 'Use' and 'Sim' tables, thus in practice allowing for a many to many relationship to be described in the database.

5.2.7 Network and SensorSets

The trained networks and ML models were kept track of in the 'Network' table. This ensured a readily available overview of the networks tested, their performance and key training parameters. To reflect the use of different models, training sets and sensors used these were identified by relationships to joining tables. Furthermore, the loss (RMSE for regression networks) were stored both from the final validation in training and for the total test set. To keep track of different training parameters of varying types the 'description' files were used, while the resulting model

Network	SensorSets
⚡idNetwork INT(10) name VARCHAR(45) ⚡idSet INT(10) ⚡idSensor INT(10) ⚡idModel INT(10) lossVal DOUBLE lossTest DOUBLE fileName TEXT description MEDIUMTEXT gen DOUBLE netFile BLOB	⚡idSensorSet INT(10) name VARCHAR(45) Sensors TEXT

file-names was stored in the 'fileName' field. To allow for redundancy, the ML model can also be stored in the 'netFile' field as binary data.

5.2.8 Results and ResultComment

For storing the predictions from the models trained, two tables were designed as this poses the same many to many relationship problem as described in 5.2.6. In this case any simulation, 'idSim', can have many results from different networks, 'idNet', and from different iterations or epochs of that network model, 'iteration'. As simulations might have a different numbers of steps

ResultComment	Results
idResultComment INT(10)	idResults INT(10)
idSim INT(10)	idRes INT(10)
idNet INT(10)	prediction DOUBLE
headline VARCHAR(45)	
comment TEXT	
RMSE DOUBLE	
ititeration INT(11)	

the results itself were stored in a separate table connected to the 'ResultComment' by its id. Through a design fault the resulting data is not identified by its step or data id, making it impractical to join this table directly with the simulations original data. To circumvent this fault the auto incremented 'idResults' was used to sort the data and the data table and result table where joined externally when evaluated together.

5.3 Generating Training, Validation and Test sets

Different training sets were generated by the MATLAB code included in Appendix A.2. The division was done by the use of pseudo-random functions reserving a predetermined factor of simulations to either training, validation or testing in the given training set. The results were then stored in the database according to section 5.2.6.

There is no agreed-upon ratio in which training, testing and validation data are divided. However. in general, one sees a 70/30 relationship between training and validation in most publications. To ensure a well-represented data set for testing the different models in a variety of cases for this paper it was desired to reserve more than 1000 simulations for this purpose, as such the final ratio for the influx-loss scenarios where set to be; 70% reserved for training, 10% for validation and 20% for testing.

6 Prediction Models

6.1 Training features

From chapter 2.2 it is known that it is a nontrivial task to analyze a NN or DNN to identify its key features. Furthermore, just as too few features will have a negative effect on a NN so will including features with no relevance for its goal. There are also large differences on the data available on different drilling rigs, many of which have no real-time data from the bottom hole, or no data at all, as described in chapter 2.1. As such there are two main goals in training the ML models explored in this paper on a few different feature sets.

First off, by analyzing the resulting predictions from the different feature sets we can infer which features are of importance to the model, and possibly also how much of a contribution is given. Secondly, due to differences in drilling rigs, all features will not be available in all cases, and there might be limited bandwidth or computational capacity. Evaluation the accuracy trade-off from removing certain features in the model is therefore an interesting aspect for this technology in terms of its availability for after-marked or new installation without cutting edge technology.

Feature set 1a & 1b: This feature set is designed to contain all pressure and flow information at the key areas of the rig. The SPP, BHP, and ADP are all affected by secondary values not relating to an influx. For the BHP the depth of the drill bit is essential. With the choke fully open there is no response on the ADP, when closing the opening there is an inverse proportional effect on the ADP. The SPP pressure is at 1 bar when there is no flow into the well and gradually increases depending on the flow rate. Due to these relationships, these additional values were added to the set.

Feature set 2a - Top Deck: As previously mentioned, bottom hole data is not always readily available on a rig. As such a set was designed to evaluate the difference in accuracy when the bottom hole readings are unavailable.

Feature set 3a - Only flow: From the theory chapter it is known that flow often is referenced as the most important factor in detecting an influx or mud loss. This set will help evaluate if there is anything to gain by adding more features to prediction.

Feature set 4a - Only Pressure: In the state of art, we see several advanced methods using pressure information to assist the prediction of influx. While pressure waves move at the speed of sound in the well and are affected by factors such as gas content, fluid properties, and

Table 3: Features sets explored in this paper

Feature set#	1	2	3	4	5
Flow In	✓	✓	✓		✓
Flow Out	✓	✓	✓		✓
SPP	✓	✓		✓	✓
ADP	✓	✓		✓	✓
BHP	✓			✓	✓
Choke opening	✓	✓		✓	✓
Bit depth ^a	✓				✓
ROP					✓
Surface RPM					✓
Sting Velocity					✓
Influx Rate	a	a	a	a	a
Change Rate ^b	b				b

^aMeasured depth (MD)

^bCalculated sum of influx rate and mud loss rate

pressure [28], these differences might be difficult to pick up on with data frequency of 1Hz. As such evaluating the effect of pressure data alone at this frequency is of interest.

Feature set 5a & 5b: In this set features relating to the movement of the drill pipe and the drill bit was added to the feature list to evaluate if this could help increase accuracy during tripping and drilling.

6.1.1 Standard score

The raw data from the simulations operate on severely different scales, with pressure values being on the scale of 10^7 and flow rates being scaled to 10^{-3} . This large difference of scale posed a problem for training the network. To solve this the standard score was calculated on the data set. (eq 12)

$$z = \frac{x - \mu}{\sigma} \quad (12)$$

Where μ equals the mean and σ equals the standard deviation of the sensor value x over the training set.

6.2 Conventional Comparison

To compare results from the ML approaches, an optimized classification trigger on delta flow was designed. This method was built to classify an Influx whenever the delta flow surpassed a limit. This is built as a comparable method to either a drilling operator observing the flow rates in and out during operations or a simple threshold alarm that can be used on platforms today. To properly compare this method against the ML models, it was given a best case scenario of being optimized on minimum loss. To achieve this the delta flow was calculated in eq 13. An initial trigger value ($T_{\Delta Q}$) was set to be the mean value of delta flow.

$$\Delta Q = Q_{Out} - Q_{Inn} \quad (13)$$

```

1 % dQ: Pre-calculated flow
2 % Truth: Pre-loaded boolean list with true on influx
3 T_dQ = fminsearch(calculateLoss, mean(dQ))
4
5 function Loss = calculateLoss(T)
6     Predict = dQ > T; % Boolean list with true on influx
7     Loss = 1-nnz(Truth==Predict)/length(Predict);
8 end

```

6.3 Network design

The networks designed and tested in this paper have been built and trained using MATLAB's deep learning toolbox. This toolbox eases the technical design process and ensures the designed network is optimized for both training and model performance on the latest technology. Significantly reducing both development and training time. For reference to the setup described below a one layer LSTM regression function can be found in appendix A.4.

For the input layers of the models presented in this paper a *sequenceInputLayer* has been utilized. This is an input layer that allows for training and predictions using sequences of data. This layer's input is a predefined sequence length during training, this length is not strongly defined as training data can include shorter sequences in the same training but not longer. While it is trained on a full sequence of data, the resulting model can be used both to predict a full sequence of responses at once or by being updated with one time-step of features at the time and returning the resulting prediction at this given time-step.

In this paper, three different types of hidden layers have been used, LSTM, BiLSTM and fully connected. All of these are seamlessly integrated with the preceding and following layers in the network. Only the numbers of layers and nodes need to be defined.

If a classification model is to be trained, the third to last layer in the model is a fully connected layer with as many nodes as there are classes to be predicted, in this case, two nodes representing either Influx or No-Influx. The second to last layer is a softmax layer. This layer uses the softmax function 14, also known as the normalized exponential function to normalize and scale each value within the range of $(0, 1)$.

$$\sigma_i = \frac{e^{N_i}}{\sum_{j=1}^K e^{N_j}} \quad (14)$$

Where: σ_i each nodes adjusted value, N output value of a node in the preceding layer, K number of nodes in preceding layer.

The last layer used in the classification network is the classification layer, this layer calculates the loss during training and returns the class with the highest probability according to the softmax layer during classification.

When designing a regression model, a fully connected layer with one node for each value to be predicted is used as the second to last layer. The last layer used is the loss function that is used to calculate the accuracy of the network during training. While there are many types of loss functions for regression networks, the root mean squared error (RMSE) has been used for this paper. This is the default loss function used by MATLAB, but also a well-documented approach. The RMSE is calculated as seen in eq 15, where y_t is the true measured response and y_p is the predicted response

$$RMSE = \sqrt{(y_p - y_t)^2} \quad (15)$$

While the loss function in both the classification and regression models can be fully customized to suit the optimization problem at hand. Generating a highly efficient cost function, and its

derivative for the optimization functions is a non-trivial task and has been omitted for this paper.

As there are no generally accepted rules for the number of layers and nodes used in the model, section 2.2, a trial and error method have been used to explore an efficient setup.

6.3.1 Alternate Classification approach

While the classification network may prove efficient, it has the disadvantage of having to be retrained if the sensitivity to be tuned. An alternate way to classify a kick event can be done by tuning a trigger value on a regression network. This may prove beneficial as it allows for real-time tuning of the network sensitivity. This method could also be compatible with the advantages of adaptive alarms found in [48].

6.3.2 Training Option

Optimization Algorithm: Both SGD and Adam optimization algorithms are well-proven training algorithms. For this paper, it was decided to only use the Adam algorithm as this offered fewer parameters to tune. Reducing the number of parameters that requires tuning, thus allowing for further exploration of different networks and training options.

By choosing the Adam algorithm, early stopping methods were not supported during training. In place of an early stopping algorithm, continuous checkpoints of the model were saved during training, one for each epoch, this way the result could be analyzed and the model with the best generalization was chosen as the final model.

While the recommended default for the learning rate is set to be 0.001 for the Adam algorithm, the fastest results were achieved using 0.004 on this data-set. Going above this value resulted in the optimization algorithm being unstable and predicting invalid values. While reducing it significantly reduced the algorithms convergence time. The learning rate appears to be independent of the network types trained within this study and the same value has therefore been used on all networks presented in the results.

Mini batch size: Based on the publications cited regarding mini-batch sizes in section 2.2, a range of 32-512 was tested. A mini-batch size of 32 proved to be much slower at an early stage of training, as such the minimum limit explored on larger data sets was set to 128.

Sequence length: The sequence length set during training is the maximum length of the feature input array used during training. With the data frequency of the simulations being 1Hz the sequence length during training equals the sequential time in seconds each model had available during training. With each simulation running for 600 seconds the sequence length would evenly match every multiple of three. By varying the sequence length on different models the impact of the available history could be analyzed.

Epochs: Due to the randomness used during training of a neural network there is no guarantee as to when a minimum will be reached. For the most part 500 epochs were sufficient for a comparison between two networks as it would near a convergence within this span as seen in figure 13a, and for the complexity of the models used in this paper it would use between 20-45

minutes to test a theory, a reasonable time for trial and error approaches. However, as seen in figure 13b, the random nature of the training could result in jumps and initialization in a local minimum outside of the scope. While figure 13 illustrates training on two different data batches similar variations also occurred with identical training data and parameters. To counteract this effect, secondary networks would be trained when encountering unexpected deviations in the result and the final proposed networks were trained to over 5000 epochs.

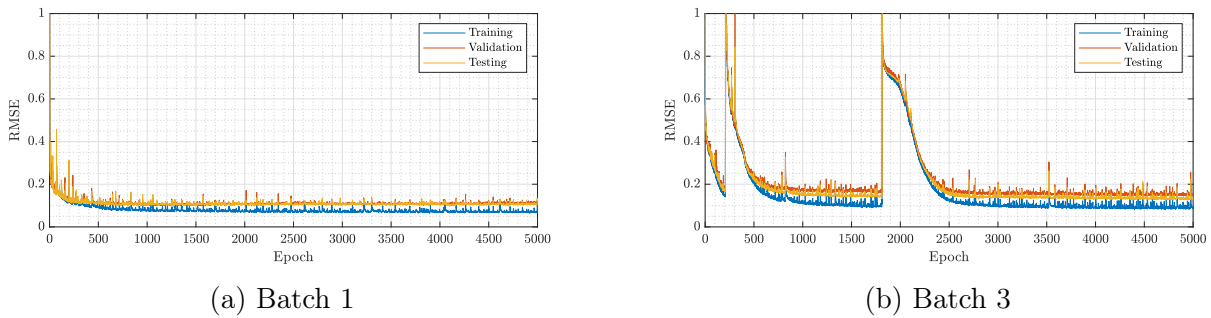


Figure 13: Training progression examples on 100 node single LSTM layer

6.3.3 Testing

The networks tested by the use of a test set in each batch that was withheld from the training and validation data. As this data has not been introduced to the model during training, this allows for testing the generalized accuracy of the network. As multiple batches of data were used during training, one test set was generated for each batch. The loss or RMSE value for each model was compared to find the best performing solution.

For models trained on Batch 1, the final epoch of the model was used to compare the network. When developing the training functions and test modules for Batch 2 and 3, a method for testing a model from each epoch of the training sequence was developed, as such the model from the best performing epoch was chosen to compare the given training modules.

While the test sets included 1.100-2000 simulations each, three different simulations have been selected to compare the actual predictions of the networks. The three simulations were selected from Batch 1 so that the network performance can be compared between the batches. Figure 14 - 16 shows the simulated response values of feature set 1 on the chosen simulations.

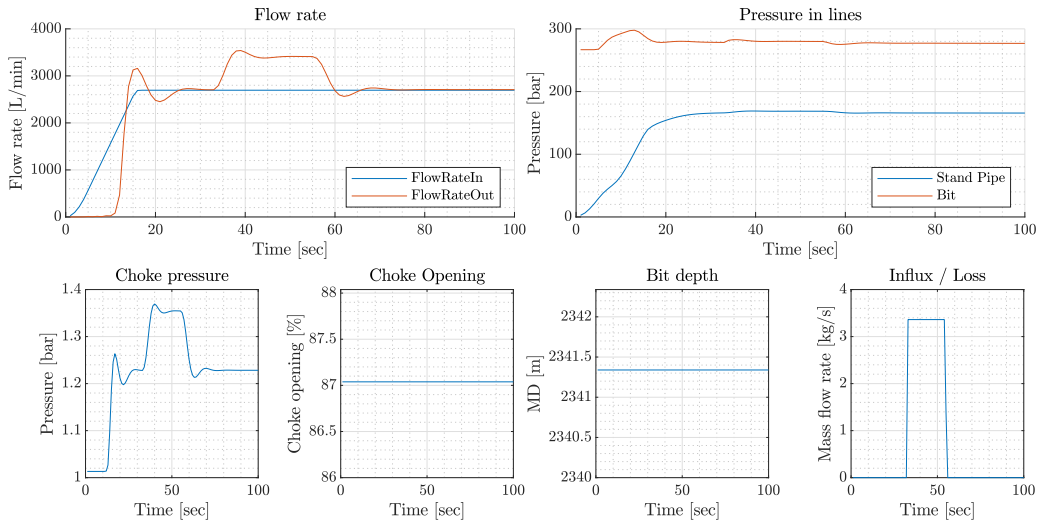


Figure 14: Simulation #6974 - Artificial influx

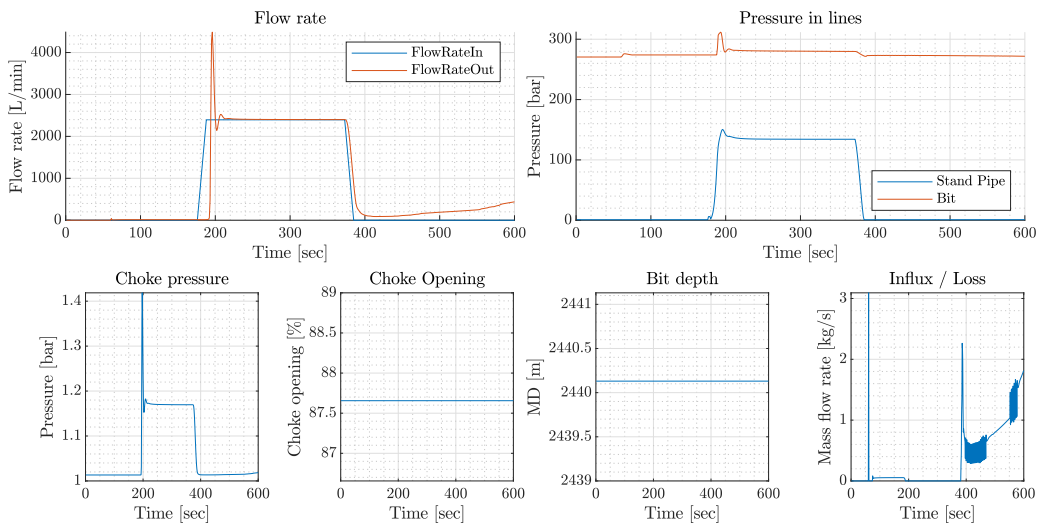


Figure 15: Simulation #6786 - Geopressure based influx

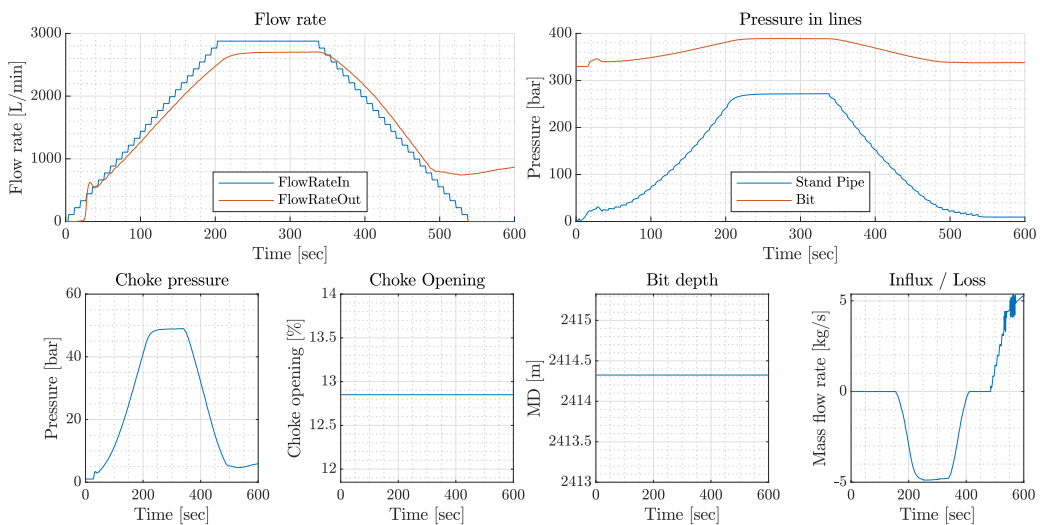


Figure 16: Simulation #6554 - Geopressure based influx and loss

7 Results

7.1 Simulation data

Due to incremental development data was gathered in two main simulation runs. The first one, Batch 1, was run early in the development process and focused on simulations with no drilling and tripping. The second major simulation run included tripping and drilling simulations. This was supplemented by Batch 1 and divided up in two new batches, Batch 2 including tripping and drilling data and Batch 3 excluding tripping and drilling.

Table 4: Simulation set 1 summary

Batch 1	Simulations	Data sets ^a	Time (w:d:h:m:s) ^b
Total	5,781 (100%)	3,468,600 (100%)	5:5:3:30:0
Normal	2,858 (67%)	3,083,306 (89%)	5:0:16:28:26
Influx	1,625 (28%)	268,195 (8%)	0:3:2:29:55
Loss	264 (5%)	117,099 (3%)	0:1:8:31:39
Both	34 (1%)	0 (0%)	0:0:0:0:0
Geopressure Influx/Loss	984 (17%)	327,198 (9%)	0:3:18:53:18
Artificial Influx	939 (16%)	58,096 (2%)	0:0:16:8:16

^aOne set being defined as reading all 18 sensor values at a single time step

^bw: Week, d: days, h: Hours, m: Minutes, s: Seconds

Table 5: Simulation data and data set 3 summary

Batch 2	Simulations	Data sets	Time (w:d:h:m:s)
Total	12,800 (100%)	7,680,000 (100%)	12:4:21:20:0
Normal	7,482 (58%)	6,671,676 (87%)	11:0:5:14:36
Influx	4,626 (36%)	751,356 (10%)	1:1:16:42:36
Loss	560 (4%)	256,968 (3%)	0:2:23:22:48
Both	132 (1%)	0 (0%)	0:0:0:0:0
Geopressure Influx/Loss	3,191 (25%)	870,111 (11%)	1:3:1:41:51
Artificial Influx	2,127 (17%)	138,213 (2%)	0:1:14:23:33
Stationary	8,069 (63%)	5,482,912 (71%)	9:0:11:1:52
- Influx	2,320 (29%) ^b	506,685 (9%) ^b	0:5:20:44:45
Tripping	2,112 (17%)	826,365 (17%)	1:2:13:32:42
- Influx	1,216 (58%) ^b	129,756 (16%) ^b	0:1:12:2:36
Drilling	2,619 (20%)	1,370,723 (18%)	2:1:20:45:23
- Influx	1,216 (58%) ^b	129,756 (16%) ^a	0:1:12:2:36

^aIn reference to parent value

Table 6: Simulation set 2 summary

Batch 3	Simulations	Datapoints	Time (w:d:h:m:s) ^a
Total	8,069 (100%)	4,841,400 (100%)	8:0:0:50:0
Normal	5,367 (67%)	4,286,213 (89%)	7:0:17:23:33
Influx	2,271 (28%)	375,161 (8%)	0:4:8:12:41
Loss	382 (5%)	170,026 (4%)	0:1:23:13:46
Both	49 (1%)	0 (0%)	0:0:0:0:0
Geopressure Influx/Loss	1397 (17%)	465,623 (10%)	0:5:9:20:23
Artificial Influx	1305 (16%)	79,564 (2%)	0:0:22:6:4

7.2 Network total RMSE comparison

Table 7 compares the results from different layer and node setups trained on the same data-set, with feature set 1 and identical training parameters. The results indicate that a single 1000 node layer of LSTM performs best, with a minor loss if the layer size is reduced to 100 nodes. Note that this change in performance is within the bounds of training noise experienced.

Table 7: Layer size RMSE on small training batch with feature set 1a

Nodes	RMSE Val $\frac{kg}{s}$
10	0.1189
10 × 10	0.1145
100	0.0958
100 × 100	0.1301
1000	0.0946
1000 × 100	0.1347

Table 8 compares the achieved influx rate RMSE on the different feature sets. Each set was trained on a 100 node single layer LSTM network using data set 1 and identical training options. These results show that using feature set 1, with both flow and pressure data produces the best results. Feature set 2 shows that removing the bottom hole readings from the set negatively impacts the accuracy of the results. While only keeping the flow data in feature set 3 further limits the accuracy. However, predictions with flow alone can produce much higher accuracy than the pressure readings by itself, seen in feature set 4.

Table 8: Feature set RMSE with single 100 node LSTM layer on batch 1

Feature Set	RMSE Test $\frac{kg}{s}$
1a	0.1049
2a	0.1607
3a	0.3272
4a	1.207

Table 9 shows the best RMSE value achieved after 500 epochs of training on the mentioned network types. NN only using fully connected layers achieve near identical performance while the LSTM and BiLSTM layers much more closely estimate the flux rate.

Table 9: Network type RMSE with feature set 1a, batch 3

Network	Nodes	RMSE Test $\frac{kg}{s}$
NN-Linear	0	0.9258
NN	30×20	0.9226
NN	$100 \times 100 \times 100 \times 50$	0.9229
LSTM	100	0.1040
BiLSTM ^a	100	0.0744

^aNot real-time results

Table 10 compares the achieved results using different mini-batch sizes and sequence lengths over 500 epochs of training. In this result, the best performing epoch of each model is compared. The results show that the number of training iterations is a function of the mini-batch size and the sequence length used to train each model. Not including the first try of 128:600 and 512:600, there is only a small difference between the results. The difference between the first and second try of 128:600 illustrates how given the same parameters the results can differ due to randomness. While this was an abnormally large jump using the same training parameters it illustrates the uncertainty. The last entry shows the poorest performance, as well as the least amount of iterations of the network tried.

Table 10: mini-batch and sequence length RMSE on Batch 2, feature set 5a, max Epoch 500

Mini -batch	Sequence length	RMSE Test $\frac{kg}{s}$	Epoch	Iteration
128	150	0.1453	484	133,584
128 ^{1st}	600	0.2795	499	34,431
128 ^{2nd}	600	0.1480	374	25,806
256	300	0.1511	466	31,688
512	150	0.1542	482	32,776
512	600	0.2370	302	5,134

Table 11: Best performing model accuracy after 5000 epochs of training

Network	Feature set	Batch	RMSE Test $\frac{kg}{s}$	Epoch	Iteration
LSTM	1a	3	0.0998	1425	62,700
LSTM	1b	3	0.1223	3522	154,968
BiLSTM	1a	3	0.0562	4590	201,960
LSTM	1a	2	0.1260	4950	336,600
LSTM	1b	2	0.1811	881	59,908
BiLSTM	1a	2	0.0757	3386	230,248
LSTM	5a	2	0.1275	4686	318,648
LSTM	5b	2	0.1577	3367	228,956
BiLSTM	5a	2	0.0805	4935	335,580

Further examining the best performing model on batch 2, BiLSTM 1a, we find that the average test simulation had an RMSE of 0.0389, with a standard deviation of 0.0650 and a median of

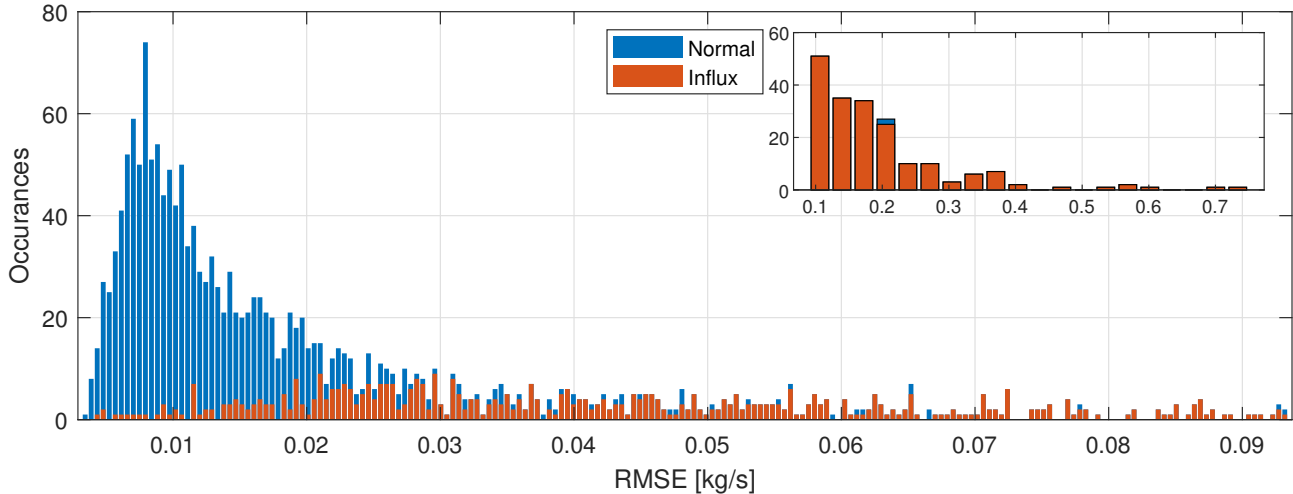


Figure 17: RMSE Histogram of BiLSTM B2 1a

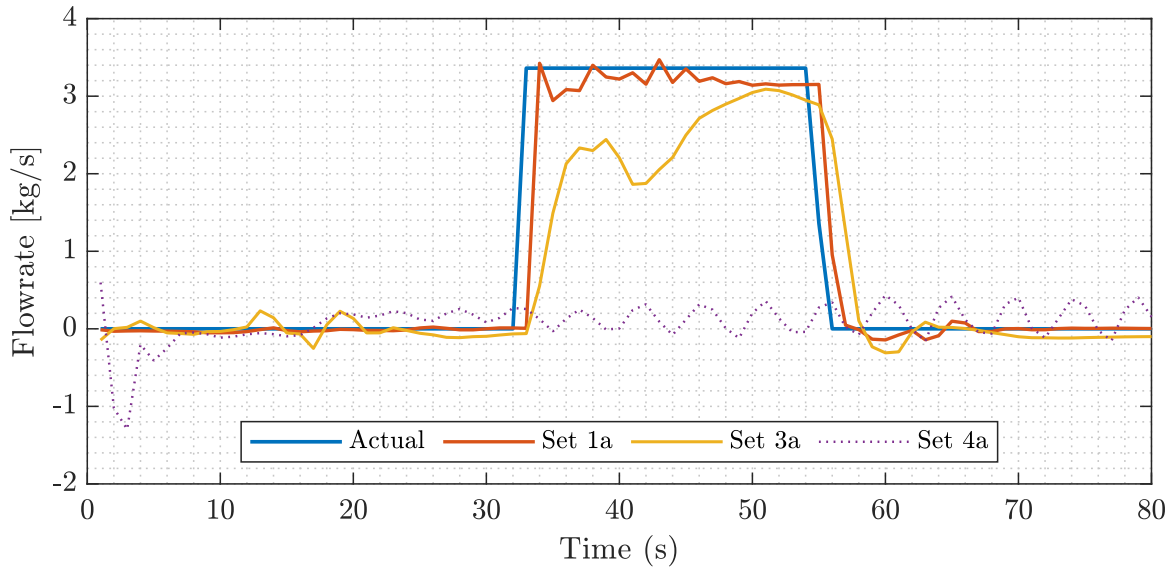
0.0155. A full histogram of the RMSE spread in this model can be seen in 17, and shows that 90% of the 1919 simulations reserved for testing archived an RMSE of less than 0.1kg/s .

7.3 Comparing feature sets and classification methods on Batch 1 feature set 1-4

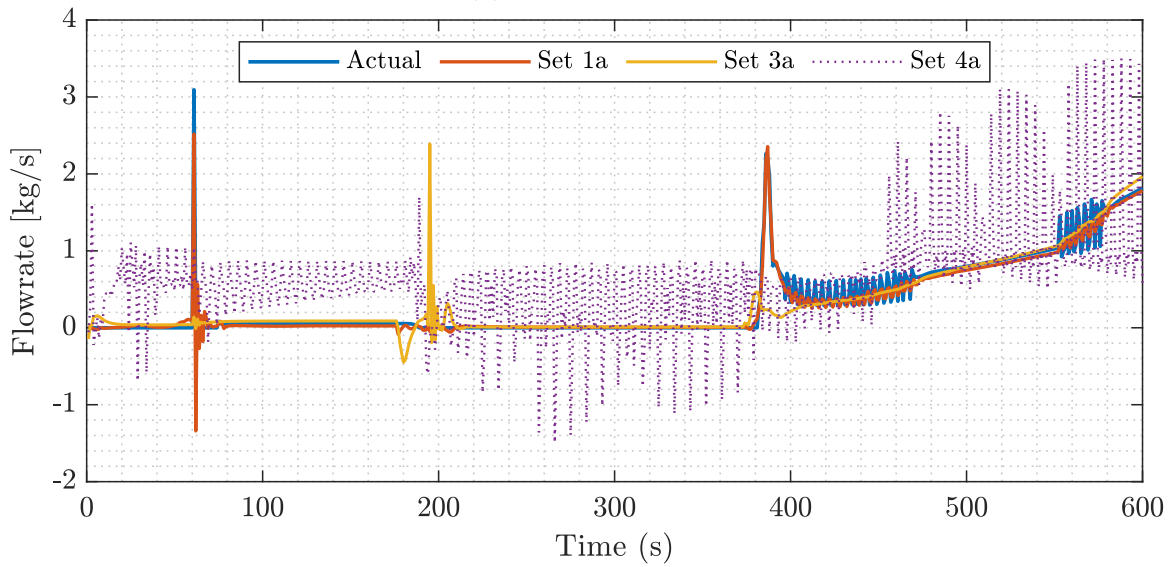
Regression: Figure 18a and 18b shows the prediction of the models on a selected artificial and geothermal influx case from the test set which has not been shown to the model during training or validation. From the figures, we see that set 1a performs most accurately in both cases by closely matching the actual influx. While in 18a it suffers from a 1s lag in the prediction it closely follows the actual influx real time in the geopressure based simulation, fig 18b. As with set 3a and 4a it misses out on the small geopressure based influx of 0.058kg/s spanning from $\sim 75\text{s}$ to $\sim 195\text{s}$ mark. Set 3a takes some time to build up towards the artificial influx and the last geopressure based influx while it also misses the first peak during the geothermal influx, and experience a false positive around the $\sim 195\text{s}$ mark, this correlates to the mud pump being turned on in this simulation. The model trained on set 4a shows no apparent detection of the artificial influx, and although there seems to be some correlation between the influx rate and the prediction in the geopressure based case it is apparent that the pressure sensor data by itself makes for an unreliable model in this case.

Examining the best performing model, set 1a, on the whole test set we achieve a root mean square error (RMSE) of 0.10 kg/s influx mass rate on the test set of the total data set. Further analyzing the test data shows that a larger part of this error comes from some uncertainty during an influx, with an RMSE of 0.34 kg/s , while it tends to be smaller during stable operations, with an RMSE of 0.04 kg/s .

The results demonstrate the effectiveness of the proposed method and show that it can effectively detect a kick in the early phases of the influx. This concludes that the proposed method can increase influx rate prediction accuracy and reduce the need for rig modifications, specialized equipment, and advanced physics-based models to detect discrepancies during operations.



(a) Artificial influx



(b) Geopressure based influx

Figure 18: Prediction of influx mass rate. Where the blue line represents the simulated influx rate, and set 1, 3a & 4a represents the predicted responses

Classification: In fig. 19, accuracy, and loss of the different influx classification methods are shown. 18c and 18d reflects different trigger values on the classification of an influx on the predicted influx rate. While 18c uses a lower trigger value to reduce false negatives, and the total loss, 18d almost completely eliminates false positives by increasing the trigger value and accepting a larger total loss on the influx classification. The limit used on 18d was 0.97 kg/s while 18c used a limit of 0.13 kg/s.

The results from the LSTM classification network are shown in 18e, and performs similar to a regression network tuned to reduce the number of false positives. In fig. 18f influxes were classified purely by a trigger value on flow rate deviation between Flow in and Flow out of the well, where the threshold was giving a best case scenario of being optimized on the minimum loss for the given test set. All classification methods presented using DNN's outperformed the traditional best case scenario. With the best network giving a $\times 2.5$ improvement.

Predicted Classification	Normal	635697 91.7%	16682 2.4%	97.4% 2.6%	Predicted Classification	Normal	637278 91.9%	31114 4.5%	95.3% 4.7%
	Influx	1665 0.2%	39556 5.7%	96.0% 4.0%		Influx	84 0.0%	25124 3.6%	99.7% 0.3%
		99.7% 0.3%	70.3% 29.7%	97.4% 2.6%			100.0% 0.0%	44.7% 55.3%	95.5% 4.5%
		Normal	Influx				Normal	Influx	
		True Classification					True Classification		

c: Low regression trigger d: High regression trigger

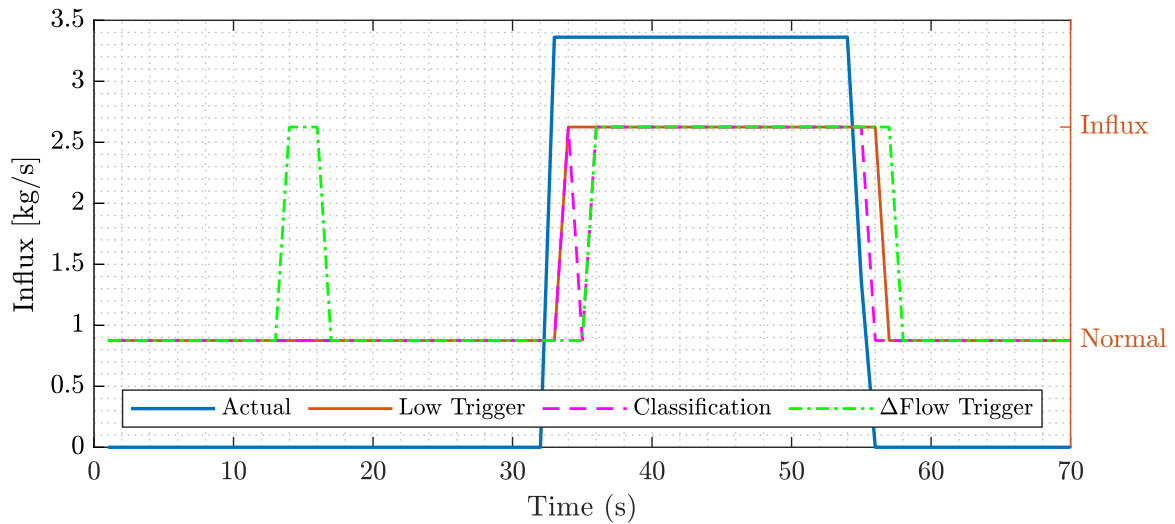
Predicted Classification	Normal	637304 91.9%	31869 4.6%	95.2% 4.8%	Predicted Classification	Normal	630332 90.8%	37427 5.4%	94.4% 5.6%
	Influx	58 0.0%	24369 3.5%	99.8% 0.2%		Influx	7624 1.1%	18817 2.7%	71.2% 28.8%
		100.0% 0.0%	43.3% 56.7%	95.4% 4.6%			98.8% 1.2%	33.5% 66.5%	93.5% 6.5%
		Normal	Influx				Normal	Influx	
		True Classification					True Classification		

e: Classification Network f: Δ Flow trigger

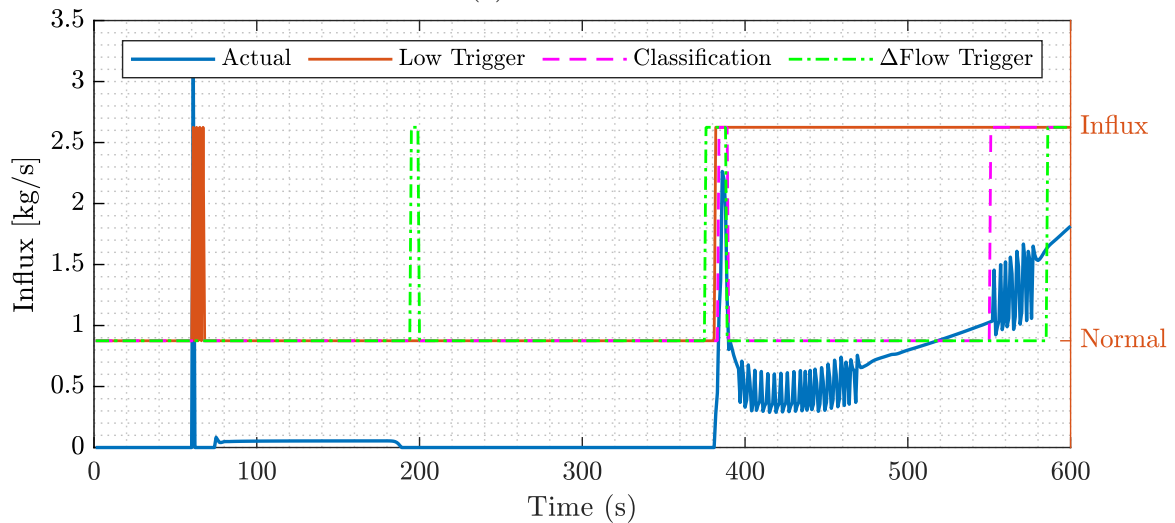
Figure 19: Classification results the whole test set

Fig. 20a and 20b compare the different influx classification methods on both the artificial influx and the geopressure based influx. The results show that the Δ Flow trigger is prone to errors. The false positive at 15s mark in fig. 20a and 195s mark in fig. 20b both correlate to the mud flow into the well being ramped up. The apparent early influx indication around the 375s mark in fig. 3 correlates to the mudflow shutdown in this scenario. During the artificial influx, it suffers from a 3 second lag in both the start and end of the influx.

The LSTM Classification network is unstable during the start of the artificial influx, first detecting it at a 1s delay and then achieving stable detecting after a 3s delay, giving a mild improvement on the Δ Flow method. It detects the end at a 1s delay, better than both the other methods. For the geopressure based influx, it also improves on the Δ Flow method with no false positives. However, it misses out on much of the main influx at the end. Detecting the first peak after 2 seconds, and then giving a false negative as soon as the influx rate reaches below 1kg/s and not detecting it again before it builds up to more than 1kg/s. The trigger on the predicted influx rate experience a 1s lag at the start of the artificial influx and 2s lag at the end. It's the only one to pick up the peak at the beginning of the geopressure influx but suffers from some noise afterward. It correctly identifies the last influx in 20b with no lag. None of the methods were able to pick up on the small influx of 0.058 kg/s in 20b.



(a) Artificial influx



(b) Geopressure based influx

Figure 20: Kick classification. Where the blue line represents the simulated influx rate on the left axis and the remaining lines is the binary classification of influx or no influx on the right axis

7.4 NN Influx prediction on Batch 1

Figure 21 shows the response of the two fully connected neural networks shown at the top of table 9. Comparing these responses with the flow rates show in the simulation response in section 6.3.3, indicates that the predicted influxes are derived from a linear relationship of flow in and flow out. Adding two layers with more nodes does not appear to improve the solution based on the linear relationship a neural network with no hidden layer can achieve.

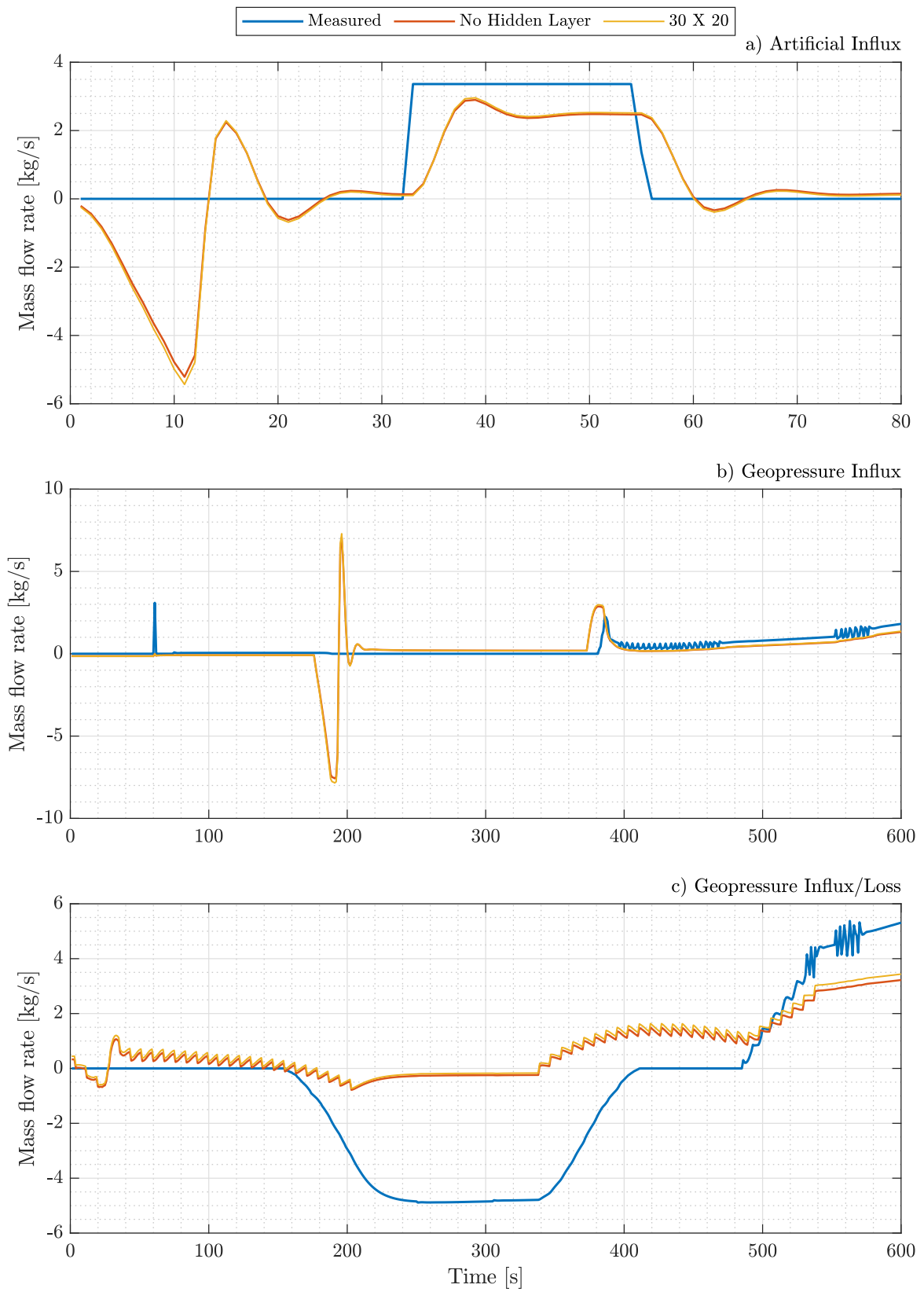


Figure 21: Non-recurrent network responses on Batch 3 with feature set 1a/b

7.5 Results from Batch 3, excluding drilling & tripping

Figure 22 show three different networks trained on Batch 3 with feature set 1. While *LSTM 1a*, only predicting influx has a 23% increase in accuracy compared to *LSTM 1b*, also predicting loss. The difference in accuracy is hard to notice on randomly selected influx simulations as seen in figure 22a) and 22b).

BiLSTM 1a performs noticeably better both at determining the edges of an influx, and more closely estimating both the flat influx mass rate in 22a), and the jigsaw pattern, in the end, on 22b). *BiLSTM 1a* is also the only one to correctly identify the 1 second influx on the 80 second mark in 22b). While it does overshoot, it is closer to the actual rate than both *LSTM 1a and 1b*. Examining the ends of the sequences in 22b) and 22c), the *BiLSTM 1a* prediction it noticeably less accurate in during the last 5-35 seconds, illustrating a BiLSTM networks reliance on both looking forward and backward in the sequence to make its predictions.

Figure 22c) illustrate how *LSTM 1a* and *BiLSTM 1a* ignores a loss while *LSTM 1b* correctly estimates its value.

7.6 Results with drilling & tripping

Figure 23 shows the responses of the three networks listed in table 11 that was trained on Batch 2, including drilling and tripping simulations, with feature set 1. While the results are near identical to those described in section 7.4, there do are some more noise during the flow in ramp up at the 170s mark in figure 23b), both in *LSTM 1b* and *BiLSTM 1a*, at the influx on the 60s mark all the models appear to have an increased accuracy.

7.7 Results with drilling & tripping including extra sensors

Figure 24 shows the responses of the three networks listed in table 11 that was trained on Batch 2, with feature set 5. The predictions shows only minor difference from figure 24 and 22.

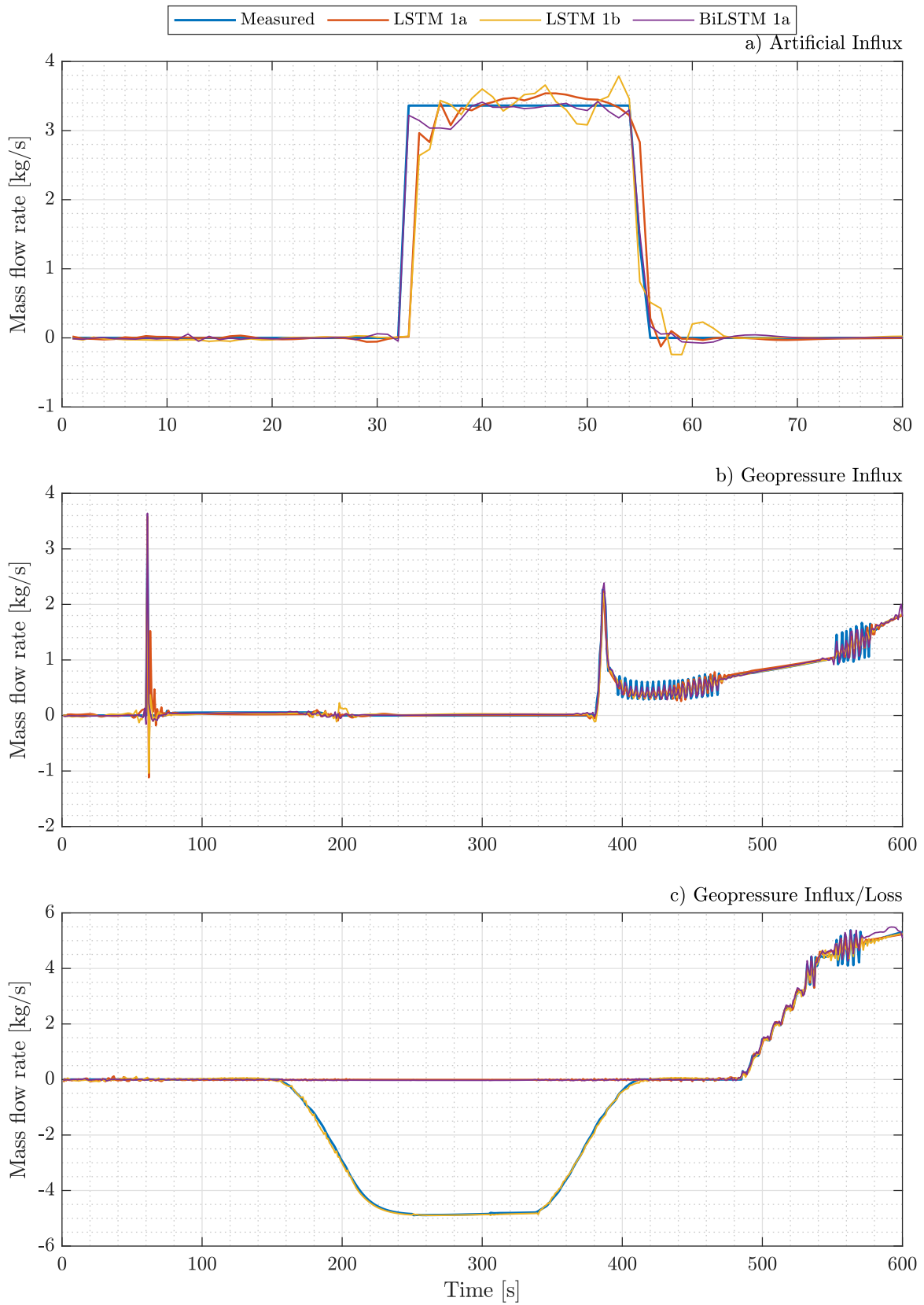


Figure 22: LSTM and BiLSTM response on Batch 3 with feature set 1a/b

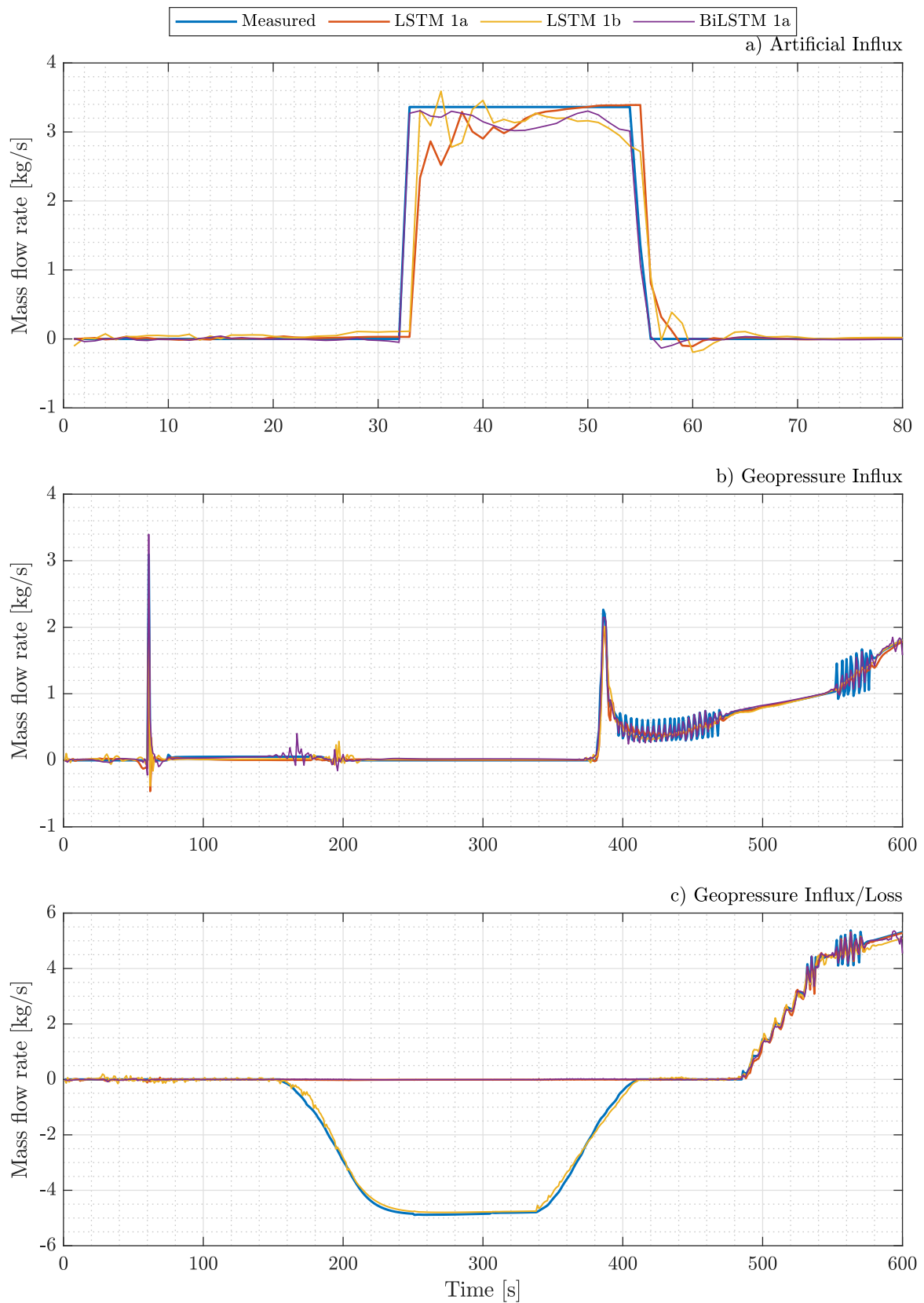


Figure 23: LSTM and BiLSTM response on Batch 2 with feature set 1a/b

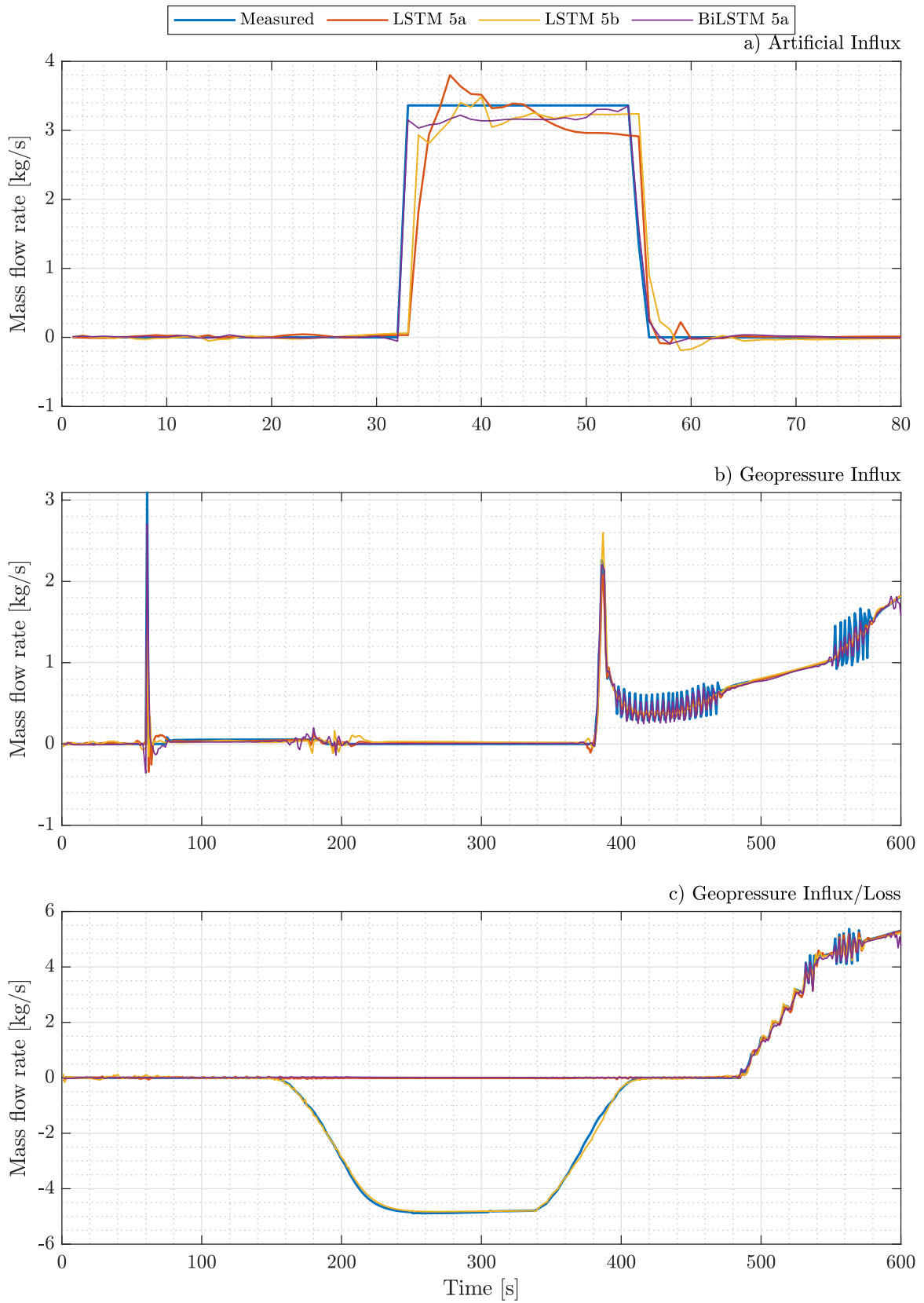


Figure 24: LSTM and BiLSTM response on Batch 2 with feature set 5a/b

7.8 Predictions during drilling

Further examining the best performing model on batch 2, BiLSTM 1a, we find that the average RMSE during a drilling simulation is 0.053, 36% higher than the overall average. For evaluating the predictions model response simulation #12450, was selected as an example due to its having an above average RMSE of 0.087, and containing an influx. The simulation response is shown in figure 25. In this simulation, a pipe connection was done between $\sim 320 - 420$ s, and is illustrated by the discontinuity in the bit depth. During this sequence, the SPP pressure is pressure sensor also displays zero. While the maximum influx rate peaked at just over 40 kg/s, prediction figure 26 and 27 has been scaled to show the rest of the graph in finer detail.

Figure 26 is the prediction done by the best performing network, BiLSTM B2 1a, this network is trained on simulations including drilling and can predict the actual influx rate with great accuracy, even throughout the pipe connection sequence.

Figure 27 is the best performing model that has not been trained on drilling simulations, BiLSTM B3 1a. While this network has trouble predicting the influx when rising and lowering the drill pipe for the connection, it closely predicts the actual influx during the rest of the simulation.

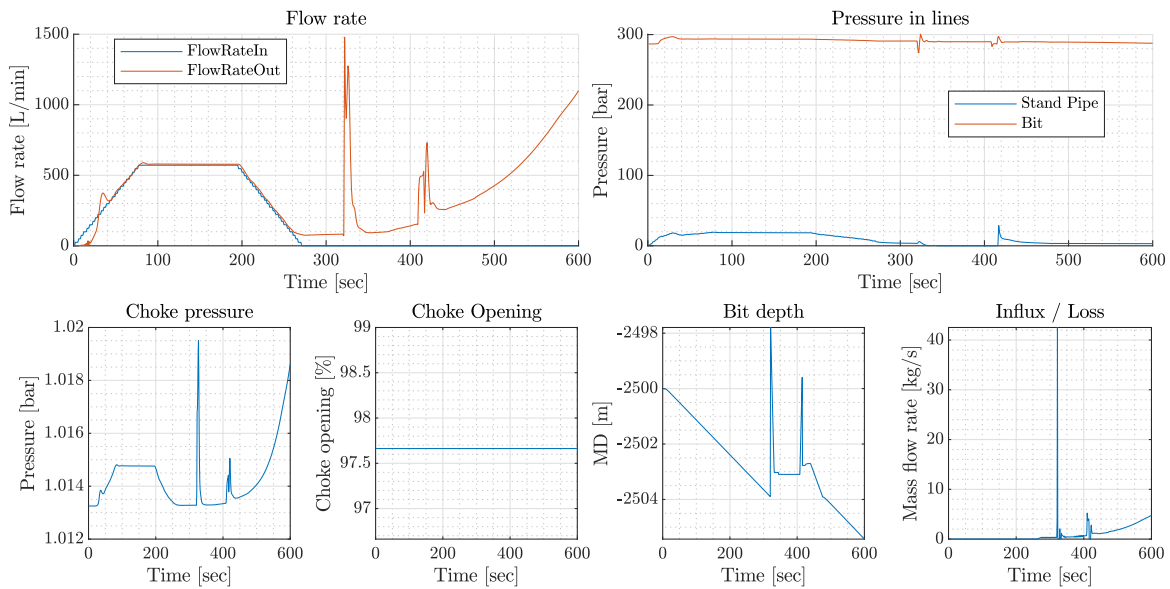


Figure 25: Simulated response of drilling simulation #12450

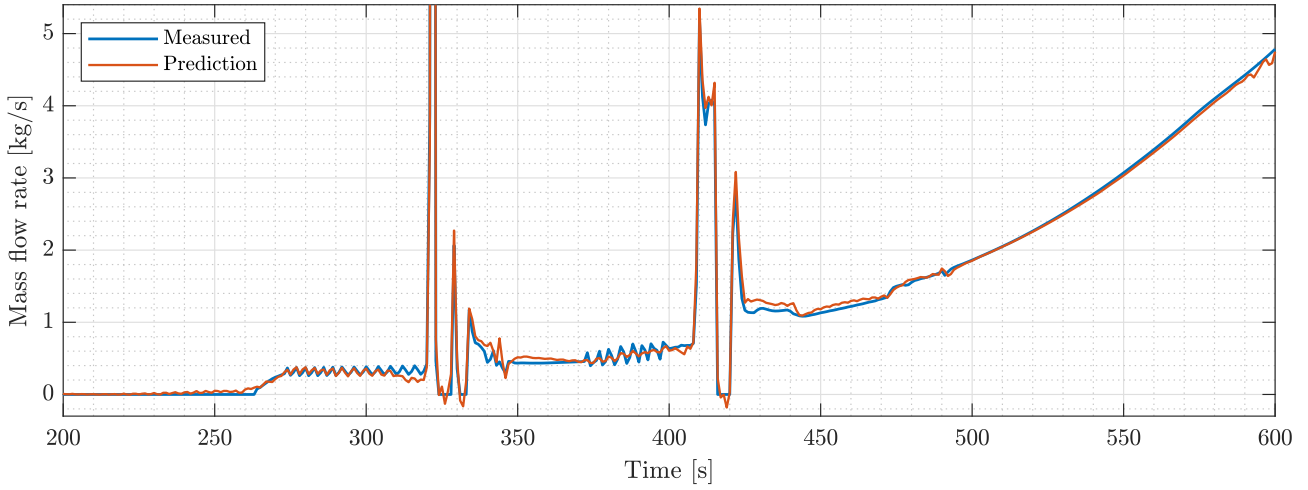


Figure 26: BiLSTM B2 1a - prediction on drilling simulation #12450

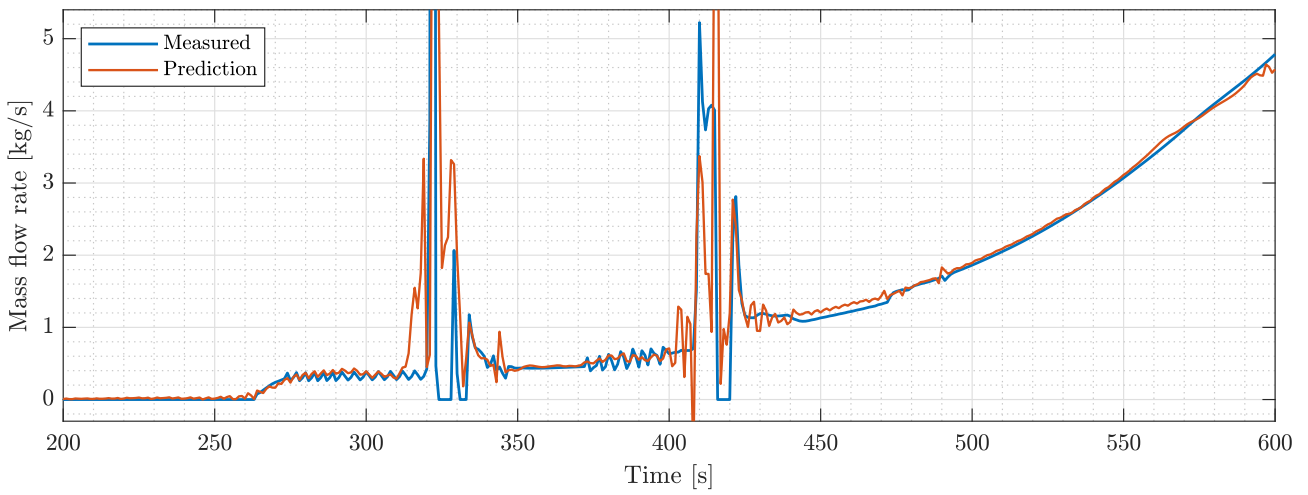


Figure 27: BiLSTM B3 1a - prediction on drilling simulation #12450

7.9 Predictions during tripping

examining the performance of model BiLSTM B2 1a on tripping simulations we find that the average RMSE per simulation is 0.0807 kg/s. For further evaluation simulation, #16276 was chosen, this simulation has an RMSE of 0.08368 and contains three pipe connections during its 600 seconds of simulation, indicated by the plateau's seen the bit depth plot of figure 28.

Figure 29 shows the prediction of BiLSTM B2 1a, as in the drilling scenario in the previous section, the model accurately predicts the influx rate throughout the simulation with only minor deviations. In this figure, the peaks at the pipe connections are also visible and the prediction closely matches the actual peak.

Figure 30 again shows the best performing model not trained on tripping or drilling, BiLSTM B3 1a. In this figure, it's apparent that the model has trouble predicting the peaks of influx that occurs during the pipe connection. Furthermore, the predictions are quite inaccurate during the 20 seconds before and after these connection peaks. While some noise is observable on the remaining sequence the model generally give an accurate prediction.

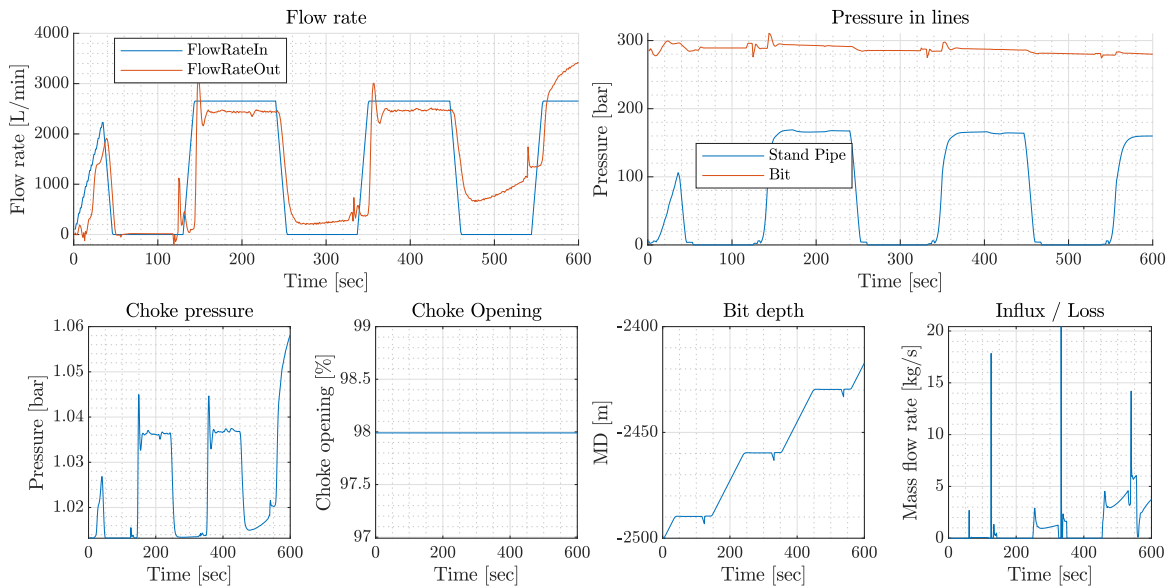


Figure 28: Simulated response of tripping simulation #12450

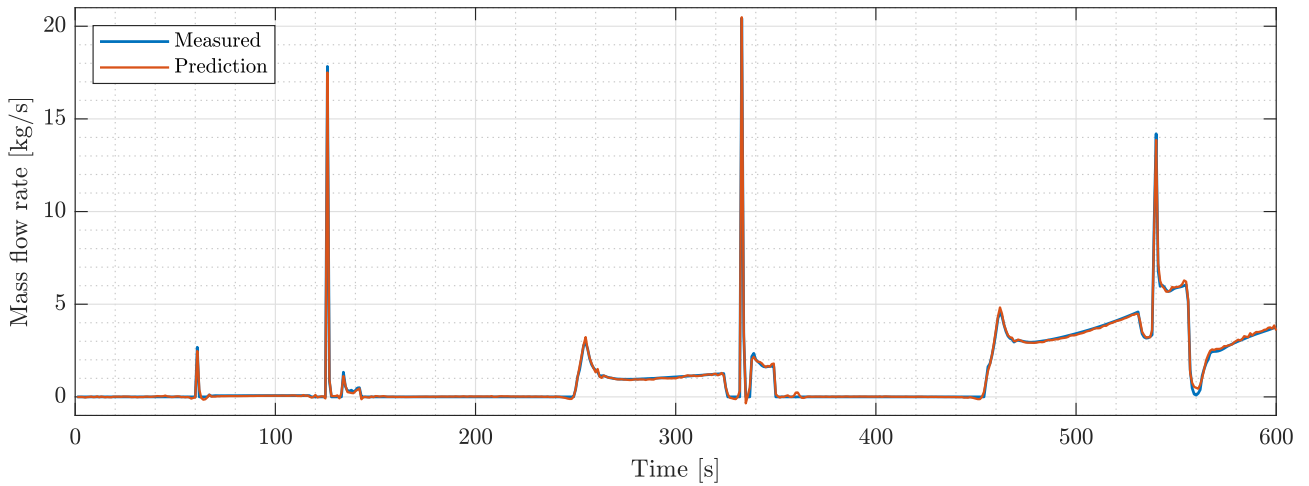


Figure 29: BiLSTM B2 1a - prediction on tripping simulation #12450

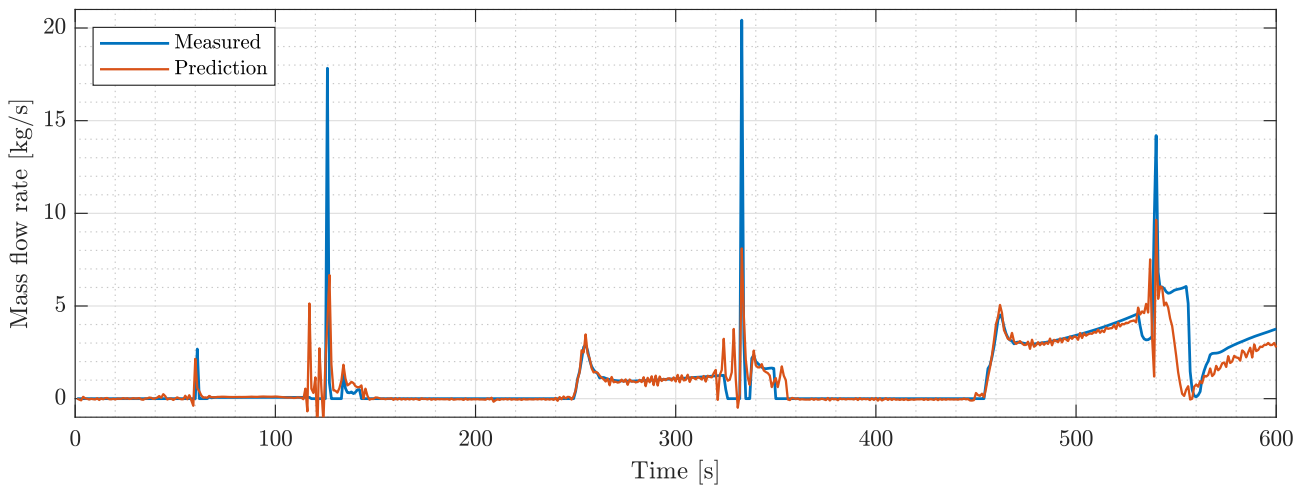


Figure 30: BiLSTM B3 1a - prediction on tripping simulation #12450

7.10 Examining badly performing cases

For the BiLSTM B2 1a model the worst performing prediction (fig 31) occurred on simulation #3104 (fig 32) with a RMSE of 0.7435 kg/s. In this simulation, an artificial influx occurs with a total mass of 48.5 kg and a mass flow rate of 6.82 kg/s. The model is unable to pick up any influx. Examining the simulation data we can observe that the choke opening is limited to 6%, that there is a continues difference in flow in and out during the first 180s but no loss of mud. From this, it appears the mud has been compressed due to high pressure resulting from the small choke opening. Examining other test results with a high RMSE value indicates that this is a reoccurring pattern.

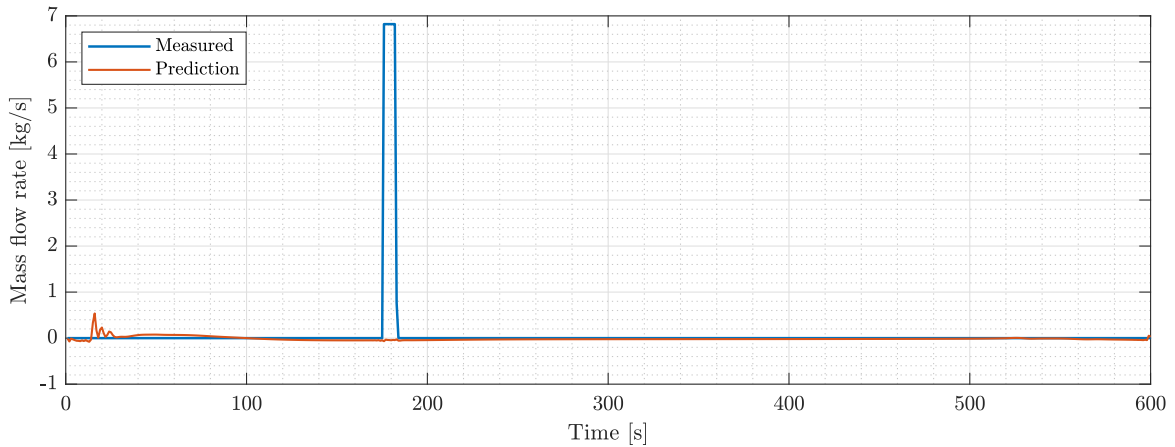


Figure 31: BiLSTM B2 1a - prediction on simulation #3104

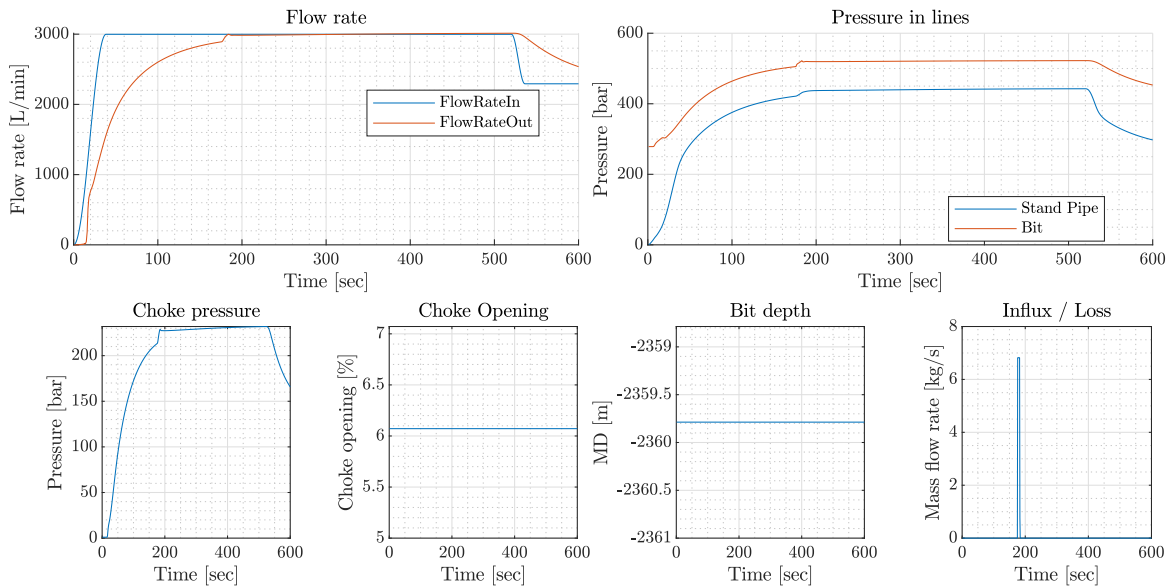


Figure 32: Simulation #3104

The second worst performing prediction (fig 33) occurred on simulation #17458 (fig 34) with a RMSE of 0.6999. This represents another reoccurring pattern of poor performing test set prediction. In this simulation, there is a large and rapid influx with a total mass of 9634 kg. While the prediction is following the actual influx rate, an increasing deviation can be seen as the influx mass rate closes in on 50 kg/s.

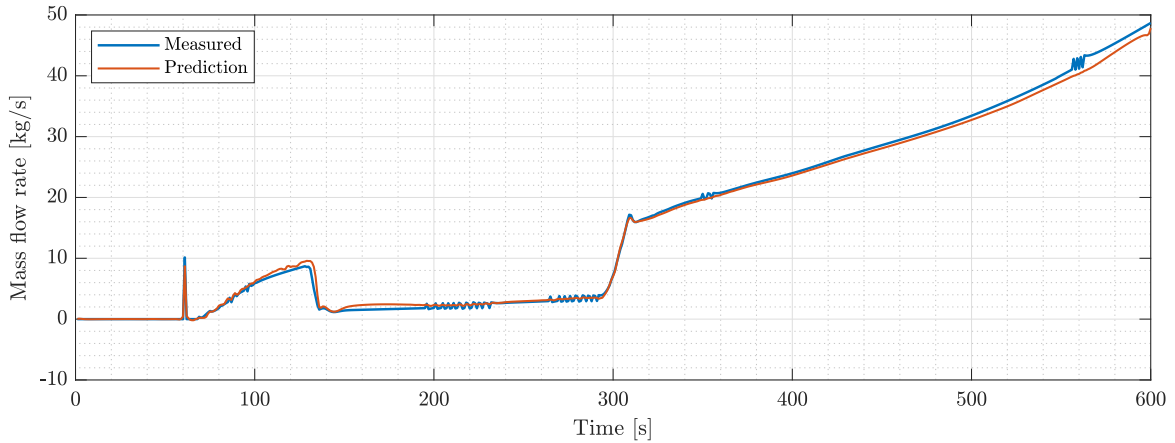


Figure 33: BiLSTM B2 1a - prediction on simulation #17458

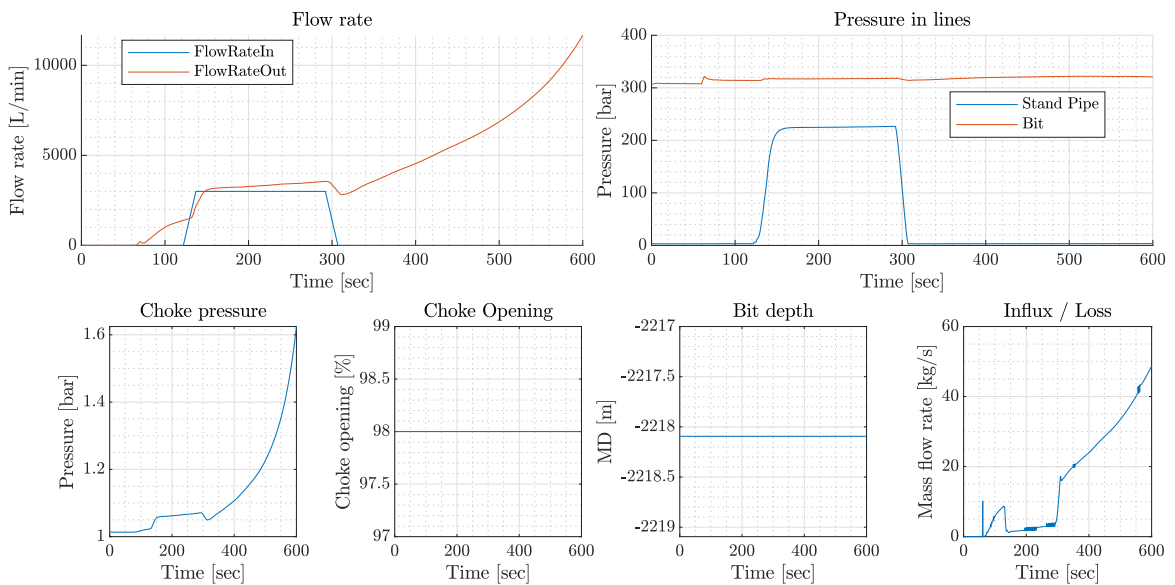


Figure 34: Simulation #17458

7.11 Training progress BiLSTM B2 1a

Figure 35 illustrates the training progression of the BiLSTM B2 1a network. With a mini-batch size of 256 and a sequence length of 300, this network evolved through 68 iterations for each epoch. The unfiltered RMSE on the training set is shown in green. For ease of comparison, the rest of the lines have been filtered with a moving average of 25 epochs.

The figure shows that the model performs best on the training set. However, it also shows that the general fitness of the predictions on the validation and test set generally follows the training set. While the minimum on the training set was found on epoch 3386, the network appears to be quite stable after epoch 2250, with only random fluctuations around a local minima.

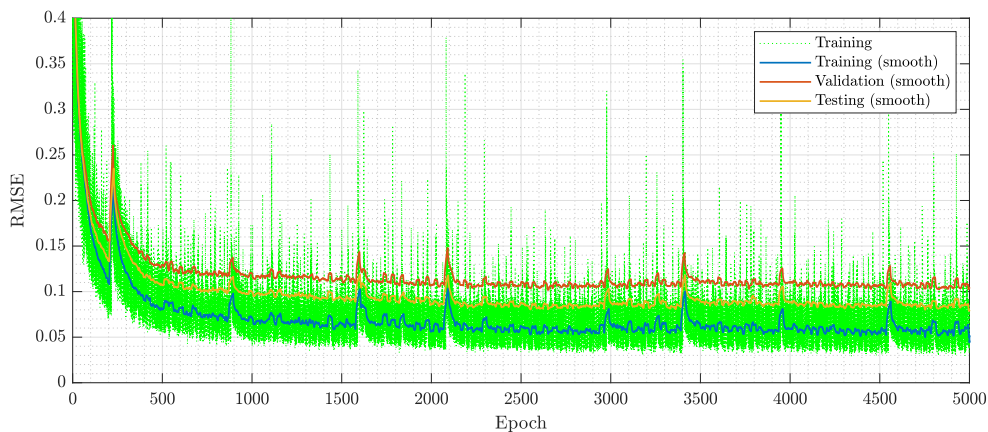


Figure 35: Training progress BiLSTM B2 1a

7.12 Training progress LSTM B2 1b

Figure 35 illustrates the training progression of the BiLSTM B2 1a network. The figures show how the training progress of these networks jumps out from the best performing local minimum towards one with a higher error.

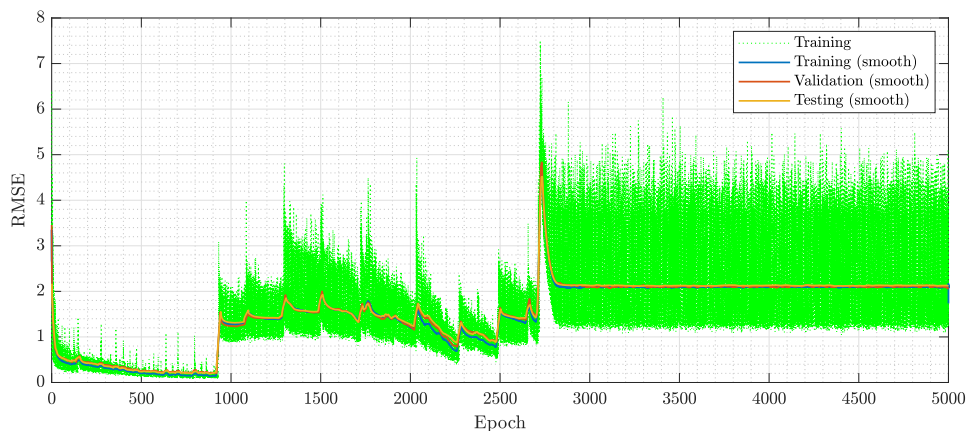


Figure 36: Training progress LSTM B2 1b

8 Discussion

8.1 Simulation data

The simulated data allows for a large variation in the training data. The increased control of the data allows for generating augmented training sets, including a greater number of edge cases and simulated responses of the well based on, at times, unrealistic input. As presented in section 2.2, this is a well known and recommended method for increasing the reliability of a model. Especially around areas of interest like an influx, which is an infrequent incident that would be dangerous to induce on a live rig to obtain more data for training a model.

The frequency at which data has been sampled through this research does not allow for the detection of a kick in under one second (one simulation step). From section 3 we know that there are patterns of interest that operate on a higher frequency than 1Hz. While one of DNN's main strengths is to pick up subtle patterns in a data sequence this might impact the performance of the network. Increasing the data frequency would however severely increase the time needed for simulations. Furthermore, data sets of lower frequencies cannot be used together with the higher frequencies. As such deciding upon a frequency early in this project, and features to be logged, helped ensure a steadily growing database of simulations that could be included in data sets with simulations of increasing complexity. The low frequency also increased training time for networks, making it easier to compare reliability in different training scenarios.

As mentioned in section 2.1, real drilling data generally include a large amount of noise, while the simulated readings represent ideal sensors. As discussed in section 4, a mitigating factor here is the low frequency at which data is read, as this allows for a quite thorough filtering of the data. To further close the gap between simulated data and real data, artificial noise could be generated on the simulated data. This would help evaluate the efficiency at which a DNN by itself can handle noisy data with no filtering. With the noise augmented simulation data its possible and often valuable to create several noise augmented data sets from the original, increasing the training size. With the increased complexity of the data, it is not unlikely that more training data is needed to achieve similar results. While this is an interesting field, this was not done in this paper due to time constraints.

In section 4.2.2 a method of varying the mud density to shift the formation pore- and fracture-pressure limits with respect to the operational conditions were implemented. While this allowed for an increased variety of well responses based on the same input data. While this had the intended effect on an influx estimation, of increasing the variety of responses to any given input, it might have had a negative effect on the mud loss estimation. This is due to the mud loss rate was given in mass per second (kg/s), and with different densities and compositions, this would affect the mass rate at which the mud left. In comparison an Influx would always be composed of methane gas, giving the same volumetric change and effects on the well at a given pressure zone. This can explain some of the lost accuracies in the prediction accuracy when including mud loss.

Moreover, while influx mass estimation proves to be able to predict the mass rate of an influx, generally with a small margin of error ($<0.15kg/s$), in the prediction models, it is uncertain how much of this comes from the difference in relationship between displaced mud and the changing

mass rate depending on the depth, and pressure of the influx. This can be particularly difficult to evaluate where an extended influx takes place and the gas keeps expanding while moving upwards in the well. To counteract the negative effects on mass rate prediction mentioned here and in the previous paragraph, a volumetric displacement might give more accurate results. Furthermore, this could give the drilling controller information about the influx in volume, and potentially the depth, based on this data the content of the influx could be determined by the controller or a connected and/or centralized expert team.

8.2 Data storage

While the database system proved to take more time and effort to design than originally intended, the system proved vital for ensuring both continuity in data use and the ability to cross reference the use and origin of all the data. With this system, it proved easy to locate simulations of predictions with a higher than normal error or to query the database after simulations containing specific characteristics.

The database proved efficient for exploring single simulations and continues injection of data from a parallel process running simulations. However, the database had trouble with efficiently exporting large batches of simulation data for training over the net. While a system could be designed to handle this process it proved easier to manually export the training sets from the database to the training server used in an optimized file format for MATLAB to interpret. As only a limited number of training sets have been generated this did not offer any issues in keeping track of where data was being used.

A flaw in the database was also noticed where the entries in the results table are not uniquely relatable to simulation data due to the lack of a time index. As the Result id is automatically set with an increment and has only been written through incrementally from a single process on a single computer in this test this has been circumvented by referencing the relative index for each simulation instead. Where multiple processes to write to this table at ones or not complete predictions for each time step in a simulation injected, this system would not work. As the tables can not join through SQL requests this data has to be patched together after extracted from the database, increasing development complexity.

8.3 Prediction Models

For this paper, both non-recurrent networks and recurrent networks have been tested. Examining the results of the non-recurrent NN models from table 9 and figure 21. It is apparent that regardless of the number of layers, fully connected non-recurrent nodes are in this case not able to predict the influx rate beyond a scalar relationship on the delta flow. This fault in this method becomes apparent as any change of flow rate into the well results in an influx prediction caused by the delayed flow rate response out of the well. This indicates that we cannot predict the influx accurately without taking into account what has happened. The absence of any improvement, beyond the bounds of expected result noise, when adding hidden layers further indicates that the accuracy achievable on a single time step is not a question of the complexity

of the describing model, but rather essential characteristics of an influx missing in the available data.

Due to the poor response of the NN models, and limited training capacity, these models were not trained on the more complex training sets, including tripping and drilling simulations. This freed up training time to further explore the RNN's.

In the simulated responses presented in figures 15 - 16. Several time dependencies can be observed in the dynamics of the well, while some are short term dependencies, like the delay in which the flow out has before it matches up with flow in, there are also longer dependencies. An example of this can be the concave pressure pulse experienced at the start of an injected influx, or the convex pressure curve at the end of one (fig 9). Since these events can be far apart, and the knowledge that one or the other has happened could be of value, a network supporting both long and short time dependencies seems preferential. As such LSTM and BiLSTM networks have been examined for this paper. While it is possible that RNN models could prove to be equally efficient, the reduced capacity for context makes this seem unlikely, especially if noise or a higher sampling frequency were introduced. Based on this, traditional RNNs have not been further examined in this paper.

The results presented in section 7.2 shows that all LSTM and BiLSTM networks performed significantly better than the NN network, regardless of training settings. With the one exception being when the features only included pressure readings. Further examining the impact of different feature sets in table 8, it is found that the flow rates in and out alone (set 3) is the most important feature for the network. While the pressure set (4) alone has difficulties in producing any accurate results. From figure 18 we can see that while the pressure based prediction comes out quite noisy, there appears to be some correlation between the prediction and the measured influx rate. The best results are found when using feature set 1, combining pressure and flow readings. Comparing these results with the final models presented in table 11, the finding stands that feature set 1 delivers the most accurate predictions. These findings indicate that the network can deduce the movement of the drill string and the resulting volume change in the well by the features represented in feature set 1 alone. As this set includes both drill bit depth and bit pressure, both directly impacted during tripping or drilling, it is likely that these are the indicators the network picks up on. In future work, this could be comparing the results with a network trained on feature set 2. It should be noticed that while the ROP and string velocity both can help describe an expected volume or pressure change in the well, the surface RPM does not correlate with any of the known causes of a simulated pressure based influx. As such the inclusion of this seemingly unrelated feature for the influx prediction may impact feature set 5 negatively. Furthermore, as String Velocity can be deduced by the change in bit depth, ROP might be the only feature adding new meaning full information in this feature set.

While stacking LSTM layers have been the cause for breakthroughs in speech recognition accuracy [58], the result presented in table 7 shows that stacking LSTM layers do not improve the results in the models trained for this paper. There are several reasons why this might not be the case in this research. Firstly, with the low data frequency there might just not be enough context for a multiple layer LSTM network to pick up on, if this is the case adding layers should be explored if the frequency is increased or noise is introduced to the layer, as both of these builds upon the complexity of the data set. Secondly; the networks trained for 7 were trained

for an equal amount of epochs, by stacking large layers the number of weights to be adjusted significantly increases. By this logic, further research should more closely examine the learning curve of networks with different layer size, for this paper this comparison was done before a system for evaluating the learning curve was developed, and the applied solution of 100 nodes performed sufficiently well. As such no further experiments were done towards optimizing the layer size was done.

For evaluating different training options, feature sets and network layouts 500 epochs have generally been used. While the training progression presented in figure 13a) and 35 both shows that the training is nearing a minimum here, 13b) and the difference in identically trained network in table 10 shows that the random nature of the training progress is reason for caution in any absolute conclusion of neural networks with near similar performance. To counteract this several of the networks presented in this paper have been retrained when encountering unexpected results or abnormalities in the training progression. For the final sets of networks to compare in table 11, the training period was also extended to 5000 epoch, saving the network on each epoch to be able to manually pick an earlier iteration if overtraining occurred or the network did a jump to a worse performing local minimum. While this, in general, produced a better result, the training progress of LSTM B2 1b (fig 36) indicates that even during longer training processes, the network can evolve to a significantly worse performing local minimum. Due to limited training capacity, this network was not retrained.

While feature set 1a is the most examined solution in this paper, this was selected as a common reference between the and not as the final proposed network. As presented in figures 22 to 24. Even a near doubling of the RMSE value (LSTM B3 1a to LSTM B2 1b) on the test set presents as nearly identical on the selected case. The more important finding seems to be the general efficiency of LSTM networks and

9 Conclusion

In this paper, a methodology for the detection of an unexpected influx during drilling operations is explored. The proposed detection methodologies are based on a flux estimator, which involves detecting and identifying the flux of fluids between permeable or fractured formations and the wellbore. The models are developed using deep learning algorithms to better detect kick and estimate the rate of influx in the well, based on readily available sensory data from the drill rig.

The results demonstrate the effectiveness of deep neural networks and show that they can effectively detect a kick in the early phases of the influx. In simulated drilling scenarios, the proposed methods can increase kick detection accuracy and reduce the need for rig modifications, specialized equipment, and advanced physics-based models to detect discrepancies during operations.

While most of the solutions examined in this paper were trained towards influx detection by use of flow readings and pressure readings relevant data from the top deck and the drill bit. The result shows that limiting bottom hole readings or including loss detection can be done with only minor loss in the model accuracy.

Furthermore, the results also show that while LSTM networks deliver high accuracy on real-time prediction, the use of BiLSTM network can improve the historical predictions. As such a potential hybrid system could be implemented with the LSTM network doing a real-time reading for high responsiveness, and a BiLSTM network could be used for the historical data. With only a 5-30 second delay on its improved estimation, it could potentially increase the human operator's trust in a decision if other data are uncertain.

However, even though the results show an adept ability to perform influx classification and influx mass rate estimation on simulated cases it is recognized that there will likely be a loss in this performance if this model were to be tested on real data. Furthermore, due to the complete lack of available real drilling data or even influx classification statistics, it is difficult to definitively make any conclusion on the performance of the proposed methods in a real scenario.

Through the development process, several research steps towards making a generalized deep neural network model for kick detection on real drilling data has been identified. Firstly the mass flow rate estimation should be generalized toward volumetric flow estimation as it's an impossible task, with current technology, to know the substance and density of an influx with high accuracy at an early stage. With a volumetric influx prediction and expert, or expert system, could classify the likelihood of the influx content.

Secondly, increasing the sampling time of the data from 1Hz could potentially greatly increase the accuracy of the models and further reduce the detection time, as the results presented by the real-time methods in this paper generally only lagged on reading behind the actual occurrence. Increasing the sampling frequency would also open up for adding noise to the data, further closing the gap between simulation data and real data. This could potentially benefit from the LSTM networks reported proficiency in filtering noise in the input data.

With the steps mentioned above taken toward reducing the gap between training on simulated data and real data, combined learning with real and synthetic data could show great potential. Especially due to the large quantities needed for training a complex neural network.

References

- [1] mathworks.com, “lstmLayer Documentation,” 2019. [Online]. Available: <https://se.mathworks.com/help/deeplearning/ref/nnet.cnn.layer.lstmLayer.html>
- [2] S. M. Testa and J. A. Jacobs, “APPENDIX A: History of Oil and Gas Production and Worst Oil Spills,” 2014. [Online]. Available: <https://www.accessengineeringlibrary.com:443/browse/oil-spills-and-gas-leaks-environmental-response-prevention-and-cost-recovery/apxA>
- [3] IRENA International Renewable Energy Agency, “Global Energy Transformation: A Roadmap to 2050,” IRENA, Tech. Rep., 2018. [Online]. Available: <http://irena.org/publications/2018/Apr/Global-Energy-Transition-A-Roadmap-to-2050>
- [4] BP, “BP Statistical Review of World Energy 2018,” Tech. Rep. 67, 2018.
- [5] J. J. Azar and G. R. Samuel, *Drilling engineering*. PennWell books, 2007.
- [6] R. Tinmannsvik, E. Albrechtsen, M. Bråtveit, I. Carlsen, I. Fylling, S. Hauge, S. Haugen, H. Hynne, M. Lundtelgenm, B. Moen, E. Okstad, T. Onshus, P. Sandvik, and K. {\O}len, “Deepwater Horizon-ulykken: Årsaker, lærepunkter og forbedrings- tiltak for norsk sokkel,” SINTEF, Tech. Rep., 2011.
- [7] W. H. Silcox, “Offshore Operations,” in *Petroleum Engineering Handbook*, 2nd ed., Richardson, Ed. Richardson, Texas: SPE, 1987, ch. 18.
- [8] M. Reeves, J. D. MacPherson, R. Zaeper, D. R. Bert, J. Shursen, W. K. Armagost, D. S. Pixton, and M. Hernandez, “High Speed Drill String Telemetry Network Enables New Real Time Drilling and Measurement Technologies,” in *IADC/SPE Drilling Conference*. Miami, Florida, USA: Society of Petroleum Engineers, 2006, p. 6. [Online]. Available: <https://doi.org/10.2118/99134-MS>
- [9] J. Hesthammer, M. Landrø, and H. Fossen, “Use and abuse of seismic data in reservoir characterisation,” *Marine and Petroleum Geology*, vol. 18, no. 5, pp. 635–655, 2001.
- [10] E. Hauge, *Automatic Kick Detection and Handling in Managed Pressure Drilling Systems*, 2013, no. March.
- [11] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Camebridge, MA: MIT Press, 2016.
- [12] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.
- [13] M. I. Jordan and T. M. Mitchell, “Machine learning: Trends, perspectives, and prospects,” *Science (New York, N.Y.)*, vol. 349, no. 6245, 2015.
- [14] H. Robbins and S. Monro, “A Stochastic Approximation Method,” *The Annals of Mathematical Statistics*, vol. 22, no. 3, pp. 400–407, 1951.
- [15] J. Kiefer and J. Wolfowitz, “Stochastic Estimation of the Maximum of a Regression Function,” *Ann. Math. Statist.*, vol. 23, no. 3, pp. 462–466, 1952. [Online]. Available: <https://projecteuclid.org:443/euclid.aoms/1177729392>

- [16] B. T. Polyak, "Some methods of speeding up the convergence of iteration methods," *USSR Computational Mathematics and Mathematical Physics*, vol. 4, no. 5, pp. 1–17, 1964.
- [17] L. Bottou, "Large-scale machine learning with stochastic gradient descent," *Proceedings of COMPSTAT 2010 - 19th International Conference on Computational Statistics, Keynote, Invited and Contributed Papers*, pp. 177–186, 2010.
- [18] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," *International conference on machine learning*, vol. 3, no. 28, pp. 1139–1147, 2013.
- [19] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization." San Diego: International Conference for Learning Representations, 12 2015. [Online]. Available: <https://arxiv.org/abs/1412.6980>
- [20] T. Lin, S. U. Stich, K. K. Patel, and M. Jaggi, "Don't Use Large Mini-Batches, Use Local SGD," 2018. [Online]. Available: <http://arxiv.org/abs/1808.07217>
- [21] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima," pp. 1–16, 2016. [Online]. Available: <http://arxiv.org/abs/1609.04836>
- [22] J. Sietsma and R. J. F. Dow, "Creating artificial neural networks that generalize," *Neural Networks*, vol. 4, no. 1, pp. 67–79, 1991. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/0893608091900332>
- [23] S. Eldevik, "AI +SAFETY," 2018. [Online]. Available: <https://ai-and-safety.dnvgl.com/>
- [24] Van Nhan Nguyen, "Automatic Power Line Inspection Powered by UAVs and Deep Learning." *Autonomikonferansen 2019 - NFEA*, 2019.
- [25] J. M. Speers, G. F. Gehrig, and others, "Delta flow: An accurate, reliable system for detecting kicks and loss of circulation during drilling," *SPE Drilling Engineering*, vol. 2, no. 04, pp. 359–363, 1987.
- [26] D. Hargreaves, S. Jardine, B. Jeffryes, and others, "Early kick detection for deepwater drilling: New probabilistic methods applied in the field," in *SPE Annual Technical Conference and Exhibition*. Society of Petroleum Engineers, 2001.
- [27] D. Reitsma and others, "Development of an automated system for the rapid detection of drilling anomalies using standpipe and discharge pressure," in *SPE/IADC Drilling Conference and Exhibition*. Society of Petroleum Engineers, 2011.
- [28] S. I. Stokka, J. O. Andersen, J. Freyer, J. Welde, and others, "Gas kick warner-an early gas influx detection method," in *SPE/IADC Drilling Conference*. Society of Petroleum Engineers, 1993.
- [29] H. Santos, C. Leuchtenberg, S. Shayegi, and others, "Micro-flux control: the next generation in drilling process for ultra-deepwater," in *Offshore Technology Conference*. Offshore Technology Conference, 2003.

- [30] H. M. Santos, E. Catak, J. I. Kinder, P. Sonnemann, and others, “Kick detection and control in oil-based mud: real well test results using micro-flux control equipment,” in *SPE/IADC Drilling Conference*. Society of Petroleum Engineers, 2007.
- [31] G. H. Nygaard, E. H. Vefring, K. K. Fjelde, G. Nævdal, R. J. Lorentzen, S. Mylvaganam, and others, “Bottomhole pressure control during drilling operations in gas-dominant wells,” *SPE Journal*, vol. 12, no. 01, pp. 49–61, 2007.
- [32] J.-M. Godhavn and others, “Control requirements for high-end automatic MPD operations,” in *SPE/IADC Drilling Conference and Exhibition*. Society of Petroleum Engineers, 2009.
- [33] J. Zhou, Ø. N. Stamnes, O. M. Aamo, and G.-O. Kaasa, “Switched control for pressure regulation and kick attenuation in a managed pressure drilling system,” *IEEE Transactions on Control Systems Technology*, vol. 19, no. 2, pp. 337–350, 2011.
- [34] G.-O. Kaasa, Ø. N. Stamnes, O. M. Aamo, L. S. Imsland, and others, “Simplified hydraulics model used for intelligent estimation of downhole pressure for a managed-pressure-drilling control system,” *SPE Drilling & Completion*, vol. 27, no. 01, pp. 127–138, 2012.
- [35] J. Zhou, J. E. Gravdal, P. Strand, and S. Hovland, “Automated kick control procedure for an influx in managed pressure drilling operations by utilizing PWD,” *Modeling, Identification and Control*, vol. 37, no. 1, pp. 31–40, 2016.
- [36] J. Zhou, “Adaptive PI Control of Bottom Hole Pressure during Oil Well Drilling,” *IFAC-PapersOnLine*, vol. 51, no. 4, pp. 166–171, 2018.
- [37] D. Hannegan, R. J. Todd, D. M. Pritchard, B. Jonasson, and others, “MPD-uniquely applicable to methane hydrate drilling,” in *SPE/IADC Underbalanced Technology Conference and Exhibition*. Society of Petroleum Engineers, 2004.
- [38] D. Reitsma and others, “A simplified and highly effective method to identify influx and losses during Managed Pressure Drilling without the use of a Coriolis flow meter.” in *SPE/IADC Managed Pressure Drilling and Underbalanced Operations Conference and Exhibition*. Society of Petroleum Engineers, 2010.
- [39] I. Mills, D. Reitsma, J. Hardt, Z. Tarique, and others, “Simulator and the first field test results of an automated early kick detection system that uses standpipe pressure and annular discharge pressure,” in *SPE/IADC Managed Pressure Drilling and Underbalanced Operations Conference and Exhibition*. Society of Petroleum Engineers, 2012.
- [40] F. Le Blay, E. Villard, S. C. Hilliard, T. Gronas, and others, “A New Generation of Well Surveillance for Early Detection of Gains and Losses When Drilling Very High Profile Ultradeepwater Wells, Improving Safety, and Optimizing Operating Procedures,” in *SPETT 2012 Energy Conference and Exhibition*. Society of Petroleum Engineers, 2012.
- [41] E. Cayeux, B. Daireaux, and others, “Precise Gain and Loss Detection Using a Transient Hydraulic Model of the Return Flow to the Pit,” in *SPE/IADC Middle East Drilling Technology Conference & Exhibition*. Society of Petroleum Engineers, 2013.

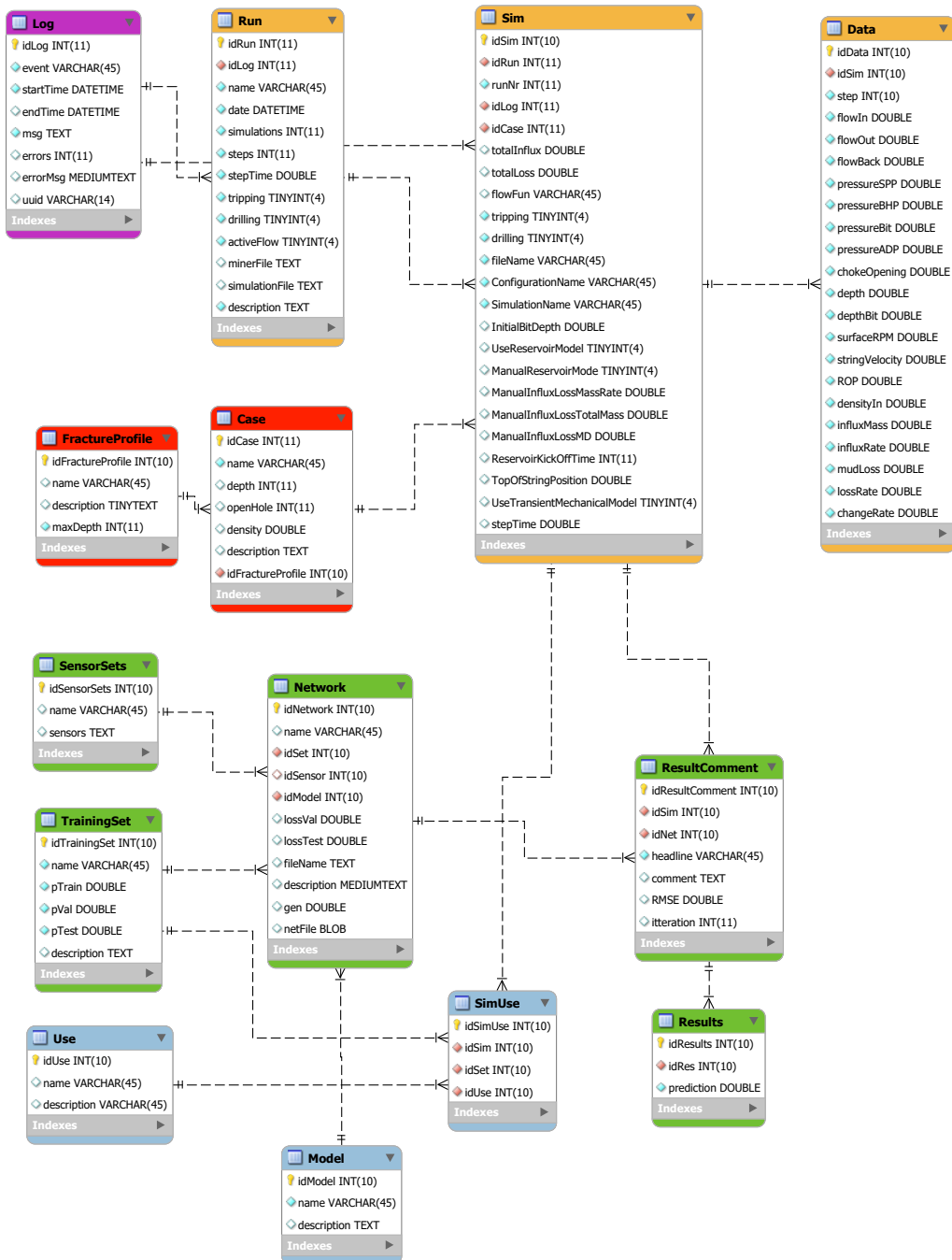
- [42] Y. Breyholtz, G. H. Nygaard, M. Nikolaou, and others, “Advanced automatic pressure control for dual-gradient drilling,” in *SPE Annual Technical Conference and Exhibition*. Society of Petroleum Engineers, 2009.
- [43] J. Zhou, Ø. N. Stamnes, O. M. Aamo, and G.-O. Kaasa, “Pressure regulation with kick attenuation in a managed pressure drilling system,” in *Proceedings of the 48th IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*. IEEE, 2009, pp. 5586–5591.
- [44] J. Zhou, G. Nygaard, J.-M. Godhavn, Ø. Breyholtz, and E. H. Vefring, “Adaptive observer for kick detection and switched control for bottomhole pressure regulation and kick attenuation during managed pressure drilling,” in *Proceedings of the 2010 American Control Conference*. IEEE, 2010, pp. 3765–3770.
- [45] J. Zhou and G. Nygaard, “Automatic model-based control scheme for stabilizing pressure during dual-gradient drilling,” *Journal of Process Control*, vol. 21, no. 8, pp. 1138–1147, 2011.
- [46] C. I. Noshi, J. J. Schubert, and others, “The Role of Machine Learning in Drilling Operations; A Review,” in *SPE/AAPG Eastern Regional Meeting*. Society of Petroleum Engineers, 2018.
- [47] O. Bello, C. Teodoriu, T. Yaqoob, J. Oppelt, J. Holzmann, and A. Obiwanne, “Application of Artificial Intelligence Techniques in Drilling System Design and Operations: A State of the Art Review and Future Research Pathways,” Lagos, Nigeria, p. 22, 2016. [Online]. Available: <https://doi.org/10.2118/184320-MS>
- [48] S. Unrau, P. Torrione, M. Hibbard, R. Smith, L. Olesen, and J. Watson, “Machine Learning Algorithms Applied to Detection of Well Control Events,” Dammam, Saudi Arabia, p. 10, 2017. [Online]. Available: <https://doi.org/10.2118/188104-MS>
- [49] M. Kamyab, S. R. Shadizadeh, H. Jazayeri-rad, and N. Dinarvand, “Early Kick Detection Using Real Time Data Analysis with Dynamic Neural Network: A Case Study in Iranian Oil Fields,” Tinapa - Calabar, Nigeria, p. 10, 2010. [Online]. Available: <https://doi.org/10.2118/136995-MS>
- [50] B. Larsen, “Drilling Process Control.” IRIS, 2018, pp. 36–38.
- [51] NORCE, “OpenLab Drilling.” [Online]. Available: <https://openlab.app/>
- [52] R. J. Lorentzen and K. K. Fjelde, “Use of slopelimiter techniques in traditional numerical methods for multi-phase flow in pipelines and wells,” *International Journal for Numerical Methods in Fluids*, vol. 48, no. 7, pp. 723–745, 7 2005. [Online]. Available: <https://doi.org/10.1002/fld.952>
- [53] R. J. Lorentzen, A. Stordal, G. Nævdal, H. A. Karlsen, and H. J. Skaug, “Estimation of Production Rates With Transient Well-Flow Modeling and the Auxiliary Particle Filter,” *SPE Journal*, vol. 19, no. 01, pp. 172–180, 2014. [Online]. Available: <https://doi.org/10.2118/165582-PA>

- [54] B. Corre, R. Eymard, and A. Guenot, “Numerical Computation of Temperature Distribution in a Wellbore While Drilling,” Houston, Texas, p. 12, 1984. [Online]. Available: <https://doi.org/10.2118/13208-MS>
- [55] E. Cayeux, T. Mesagan, S. Tanripada, M. Zidan, and K. K. Fjelde, “Real-Time Evaluation of Hole-Cleaning Conditions With a Transient Cuttings-Transport Model,” *SPE Drilling & Completion*, vol. 29, no. 01, pp. 5–21, 2014. [Online]. Available: <https://doi.org/10.2118/163492-PA>
- [56] Y. Yi, B. Lund, B. Aas, A. X. He, R. Rommetveit, L. D. Salem, S. Stokka, and F. Bottazzi, “An Advanced Coiled Tubing Simulator for Calculations of Mechanical and Flow Effects; Model Advancements, and Full-Scale Verification Experiments,” Houston, Texas, p. 12, 2004. [Online]. Available: <https://doi.org/10.2118/89455-MS>
- [57] Å. Kyllingstad, “Buckling of tubular strings in curved wells,” *Journal of Petroleum Science and Engineering*, vol. 12, no. 3, pp. 209–218, 1995. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/0920410594000467>
- [58] A. Graves, A. R. Mohamed, and G. Hinton, “Speech recognition with deep recurrent neural networks,” *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, no. 3, pp. 6645–6649, 2013.

A Appendices

A.1 Entry Relation Diagram

Table 12: ER Diagram SQL DB



A.2 Code for generating training sets

A.2.1 Main script

```
1 %% Settings
2 trainingSet = 3; % idTrainingSet
3 pTest = 0.2; % Fraction of test data
4 pVal = 0.1; % Fraction of Validation data
5 selectRuns = 6:9; %Simulation batches to include
6 useCaseRatio = true; % Override influx population
7 caseRatio = 0.5; % minimum of cases including influx
8
9 %% Fetch Simulation id's with influx and loss
10 runsS = strjoin(cellstr(num2str(selectRuns')), ', ');
11 whereQ = ['WHERE Sim.idRun IN (', runsS, ') ', ...
12          ' AND Sim.totalInflux IS NOT NULL '];
13 query = ['SELECT idSim, totalInflux, totalLoss FROM Sim ', whereQ];
14 conn = connectToDB();
15
16 SimUse = fetch(conn, query); % returns table with desired columns
17
18 %% count influx cases
19 if useCaseRatio
20     isInLoss = any(SimUse{:,2:3} > 0, 2); %number of influx or loss
21         simulations
22     nIntrest = nnz(isInLoss); %number of normal simulations
23     ratio = nIntrest/height(SimUse); % ratio
24     if ratio < caseRatio % if ratio outside goal
25         numNormalCases = round(nIntrest/caseRatio - nIntrest); %
26             Calculate normal simulations to keep
27         temp = SimUse(~isInLoss, :); % Copy to temporary variable
28         SimUse(~isInLoss, :) = []; % delete from main table
29         maxID = height(temp); % find total number of normal
30         idxL = randperm(maxID, numNormalCases); %select a
31             predetermined number of random simulations
32         SimUse = [SimUse; temp(idxL, :)]; % Join selected simulations
33             in complete list
34         clear temp idxL maxID numNormalCases ratio
35     end
36 end
37 SimUse(:, 2:3) = []; % Remove excess data
38 SimUse = sortrows(SimUse, 'idSim'); % Sort by sim id
39
40 SimUse.idSet = repmat(trainingSet, height(SimUse), 1); % Set training
41     sett ID
42 SimUse.idUse = ones(height(SimUse), 1); % preallocateidUse
```

```

38 % Get indexes for Train, Test and validation
39 [idxTrain, idxTest, idxVal] = partitionTestValData(pTest, pVal, height(
    SimUse));
40
41 SimUse.idUse(idxTrain) = 2; % Set training id
42 SimUse.idUse(idxVal) = 3; % Set validation id
43 SimUse.idUse(idxTest) = 4; % Set test id
44
45 %% Verify distribution
46 pTest = nnz(SimUse.idUse == 4)/height(SimUse)
47 pVal = nnz(SimUse.idUse == 3)/height(SimUse)
48 pTrain = nnz(SimUse.idUse == 2)/height(SimUse)
49
50 sqlwrite(conn, 'SimUse', SimUse) %% Push table to db

```

A.2.2 partitionTestValData

```

1 function [idxTrain, idxTest, idxVal] = partitionTestValData(pTest,
    pVal, listLength)
2
3 % preallocate logical arrays
4 idxTest = false(1, listLength);
5 idxVal = idxTest;
6
7 % Calculate number of validation simulations
8 numOut = round(listLength*(pVal+pTest));
9 vOut = floor(numOut*(pVal/(pVal+pTest)));
10
11
12 idxOut = randperm(listLength, numOut); % Select cases at random, in a
    random ordered list
13 V = idxOut(1:vOut); % Assign the first values to validation
14 T = idxOut(vOut+1:end); % Assign rest to Training
15
16 % create logical index arrays
17 idxTest(T) = true;
18 idxVal(V) = true;
19 idxTrain = ~(idxTest|idxVal); % NOT idxTest OR idxVal
20
21 end

```

A.3 Flow Functions

The following functions were used to generate the active flow patterns used in simulations. Except for the static function all are randomly seeded to ensure variety in the patterns. For the flow patterns to behave consistently throughout a simulation 5 random values were generated in the start of a simulation, and passed to the functions via the 'seed' variable. The total simulation time was given in 'maxI' and the current time in 'i'. Standardizing these values for all functions allowed for the parent code to choose any of the functions without errors.

Static

```
1 function [ x ] = flowStatic(i ,maxI , seed)
2 x = 1;
3 end % function
```

Static Step

```
1 function [ x ] = flowStaticStep(i ,maxI , seed)
2 startTime = maxI/3*seed(1);
3 endtime = (maxI-startTime)*seed(1);
4 if i>startTime && i<endtime
5     x = 1;
6 else
7     x=0;
8 end
9 end % function
```

Ramp up

```
1 function [ x ] = flowRampUp(i ,maxI , seed)
2 p = 10+seed(1)*50;
3
4 if i<p
5     x = sin(i/p*pi-pi/2)/2+0.5;
6 else
7     x = 1;
8 end
9
10 end % function
```

Ramp up and down

```

1 function [ x ] = flowRampBump(i ,maxI , seed )
2 p = 10+seed(1)*40;
3 pDown = 4+15*seed(3);
4 pPause = seed(2)*(maxI-pDown-p);
5 bmpSize = 1*seed(4);
6 if i<p
7     x = (sin(i/p*pi-pi/2)/2+0.5);
8 elseif i<p+pPause
9     x = 1;
10 elseif i < p+pPause+pDown
11     x = 1-bmpSize/2-(sin((i-p-pPause)/pDown*pi-pi/2)/2)*(bmpSize);
12 else
13     x = 1-bmpSize;
14 end
15
16 end % function

```

Step Up

```

1 function [ x ] = flowStepUp(i ,maxI , seed )
2 upTime = round(maxI*seed(1));
3 steps = 1+round(maxI/15*seed(2));
4
5 if i<upTime
6     x = round(i*steps/upTime)/steps;
7 else
8     x=1;
9 end
10
11 end % function

```

Step Up and down

```

1 function [ x ] = flowStepUpDown(i ,maxI , seed )
2 upTime = round(maxI/2*seed(1));
3 downTime = round(maxI/2*seed(1));
4 pauseTime = round((maxI-upTime-downTime)*seed(1));
5 stepsUp = 1+round(maxI/15*seed(2));
6 stepsdown = 1+round(maxI/15*seed(2));
7
8 if i<upTime
9     x = round(i*stepsUp/upTime)/stepsUp;

```

A. APPENDICES

```
10 elseif i<upTime+pauseTime
11     x = 1;
12 elseif i<upTime+pauseTime+downTime
13     x = 1-round((i-upTime-pauseTime)*stepsdown/downTime)/stepsUp;
14 else
15     x=0;
16 end
17
18 end % function
```

Sin

```
1 function [ x ] = flowSin(i ,maxI ,seed)
2 p=2+seed(1)*5;
3 a = seed(2);
4 x = 1-(cos(i/maxI*pi*2*p)/2+0.5)*a;
5
6 end % function
```

Random steps

```
1 function [ x ] = flowRand(i ,maxI ,seed)
2
3 steps = round(10*seed(1));
4 atStep = round(i*steps/maxI);
5 x =seed(mod(atStep ,length(seed))+1);
6
7 end % function
```

A.4 LSTM Training function

```

1 function trainLSTMNet(file , feature , mb, MaxEpochs, sequenceL , lr)
2
3 load([file , '.mat']) % Load training data
4
5 %%Set save path and name
6 folder =['res/' , file , '_E' , num2str(MaxEpochs) , 'B' , num2str(mb) , 'S' ,
7         num2str(sequenceL) , 'L' , replace(num2str(lr) , '.' , '')];
8
9 mkdir(folder) % Create checkpoint folder
10
11 %% Create layers
12 layers = [sequenceInputLayer(feature , 'Name' , 'inRaw') , ...
13          lstmLayer(100 , 'OutputMode' , 'sequence' , 'Name' , 'memberBerry') , ...
14          fullyConnectedLayer(1 , 'Name' , 'solver') , ...
15          regressionLayer('Name' , 'RMSE')];
16
17 %% Set Training Options
18 opts = trainingOptions('adam' , ...
19                       'InitialLearnRate' , lr , ...
20                       'MiniBatchSize' , mb , ...
21                       'Shuffle' , 'every-epoch' , ...
22                       'MaxEpochs' , MaxEpochs , ...
23                       'ValidationData' , {zDataVal , yDataVal} , ...
24                       'CheckpointPath' , folder , ...
25                       'SequenceLength' , sequenceL);
26
27 %% Train
28
29 [net , netStats] = trainNetwork(zDataTrain , yDataTrain , layers , opts);
30
31 %% Save resulting Workspace
32 save(['WS_' , folder])

```
