# Early Sign Detection for the Stuck Pipe Scenarios using Unsupervised Deep Learning

**Citation for published version:**
Mopuri, KR, Bilen, H, Tsuchihashi, N, Wada, R, Inoue, T, Kusanagi, K, Nishiyama, T & Tamamura, H 2022, 'Early Sign Detection for the Stuck Pipe Scenarios using Unsupervised Deep Learning', *Journal of Petroleum Science and Engineering*, vol. 208, no. Part C, 109489. https://doi.org/10.1016/j.petrol.2021.109489

**Digital Object Identifier (DOI):**
10.1016/j.petrol.2021.109489

**Link:**
Link to publication record in Edinburgh Research Explorer

**Document Version:**
Peer reviewed version

**Published In:**
Journal of Petroleum Science and Engineering

# Early Sign Detection for the Stuck Pipe Scenarios using Unsupervised Deep Learning

Konda Reddy Mopuri[a,*], Hakan Bilen[a], Naoki Tsuchihashi[b], Ryota Wada[b], Tomoya Inoue[c], Kazuya Kusanagi[d], Tazuru Nishiyama[e] and Hitoshi Tamamura[f]

[a] *School of Informatics, University of Edinburgh, UK*
[b] *University of Tokyo, Japan*
[c] *JAMSTEC (Japan Agency Marine-Science and Technology), Japan*
[d] *JOGMEC (Japan Oil, Gas and Metals National Corporation), Japan*
[e] *Japex (Japan Petroleum Exploration Co., Ltd.), Japan*
[f] *INPEX Corporation, Japan*

ARTICLE INFO

*Keywords*:
Stuck prediction
field drilling data
unsupervised machine learning
deep learning
autoencoder

ABSTRACT

In this paper we present a novel approach for detecting early signs for the stuck events in drilling using Deep Learning. Specifically, we adapt neural network based unsupervised learning tool called Autoencoder for anticipating the 'stuck' events during the drilling process. We build Autoencoders on Recurrent Neural Networks (RNNs) to model the normal drilling activity, thereby detecting the stuck incidents as anomalous activity. We conduct experiments on the actual drilling data collected from 30 field wells operated by multiple drilling sources with diverse well profiles and demonstrate that our approach obtains promising results for the stuck sign detection. Furthermore towards explaining the trained model's prediction, we present reconstruction analysis on the individual drilling parameters.

## 1. Introduction

Stuck pipe is one of the important issues faced by the drilling industry and results in significant financial and time loss during the drilling for companies and in some cases even leads to eventual desertion of the borehole. Various industry reports (e.g. Muqeem et al. (2012), Shadizadeh et al. (2010), Siruvuri et al. (2006), Elmousalami et al. (2020), etc.) estimate that the costs caused by the stuck pipes may exceed several hundred million US dollars. In addition, according to the same sources, stuck pipe cases account for 25% of NPT (Non Productive Time) which translates to a cost of about 2 rig years annually. Hence, it is of great interest to develop systems for their early detection and enable engineers to avoid such cases.

Conventionally, predicting anomaly events in drilling operation such as stuck pipe is based on alerts generated when data indicates deviation between the measured value and predicted value (e.g. Salminen et al. (2017)). However, majorly, the accuracy of predicted value is often low, and this raises many false alarms. Several researches have utilized machine learning approaches to address the drilling related problem. Here we briefly discuss the existing works and motivate the necessity for the proposed approach.

Jahanbakhshi et al. (2012) developed a support vector machine (SVM) based approach for predicting a differential stuck. Also, Heinze et al. (2012), Zhu et al. (2019) and Alshaikh et al. (2019) developed data-driven supervised models to detect stuck incident based on Decision Tree (DT), Artificial Neural Network (ANN), and Support Vector Machine (SVM) using the surface drilling parameters.

Tripathi et al. (2020) presented a hierarchical approach that consists of Fuzzy Rule Based and Random Forest classifier to identify drilling activities. General drawback of Fuzzy logic methods is that (Godil et al. (2011)) it is tedious to develop the required fuzzy rules and the membership functions. Further, the outputs of such systems can be interpreted in multiple ways making the analysis difficult. Ramba et al. (2020) developed a methodology to estimate and monitor the hookload for improving the drilling performance. However, their mathematical model relies on a

[*] Corresponding author. Present Address: Dept. of Computer Science and Engineering, Indian Institute of Technology Tirupati, Tirupati – Renigunta Road, Settipalli Post, Tirupati, India, 517 506

✉ mkreddy.402@gmail.com (K.R. Mopuri)
ORCID(s): 0000-0001-8894-7212 (K.R. Mopuri)

single parameter to detect complex anomalies leaving out crucial parameters such as torque, SPP, ROP, etc. from the analysis. Given the diverse nature of the drilling problems, these systems are expected to capture the complex correlations that exist among the drilling parameters (time series drilling data).

Recently Machine Learning (ML) has witnessed a giant leap in several challenging Artificial Intelligence (AI) problems including automatic scene understanding (Krizhevsky at al. (2012)), speech processing (Van den Oord et al. (2016)), and natural language processing (Young et al. (2018)). Particularly, advances in Deep Learning or Deep Neural Networks (DNNs) research, have drastically reduced the gap between the performance of the machines and that of humans in some of these tasks. These ML tools have been swiftly adapted to other branches of science such as astrophysics (e.g. Allen et al. (2019)), fluid dynamics (e.g. Lye et al. (2020)), etc.

In this paper we adapt some of these advancements for the first time to the stuck pipe early detection problem. Drilling data is heavily imbalanced with volumes of data collected during normal activity and very little during the stuck activity (stuck events are rare and diverse). This poses a severe challenge for learning intelligent systems in a supervised machine learning context and leads to incorrectly fit models that do not generalize well (refer to section 5 and Table 4). In addition, for early sign detection for stuck pipes, critical concern is that there is lack of an exact sign of stuck before it occurs, which makes it difficult for supervised modelling of the problem. Analysing the drilling data for detecting early signs for stuck scenarios need to understand the composite correlations among the drilling parameters. In general, the existing data driven methods towards the stuck analysis (e.g. Agwu et al. (2018), and Magana-Mora et al. (2019)) do not effectively learn on the long time series drilling data and often suffer from the extreme imbalance in the drilling data and limited modelling capacity (e.g. shallow neural networks by Lind et al. (2014)). This motivates us to consider the tools that can handle the data imbalance and long-term relations in the data. Therefore, in this paper, we frame the problem of early predicting the stuck condition in the drilling process as an unsupervised machine learning problem and employ deep neural networks to solve it. We utilize the LSTM-AE (Long Short Term Memory - Autoencoder) model because, (i) LSTM is a kind of artificial neural network that can capture the long-term dependencies present in the multi-dimensional input and hence is ideal to learn from the time series drilling data, (ii) Autoencoder is an unsupervised learning technique that can learn from a single class data (e.g. measured during the normal operation only).

## 2. Overview of Machine Learning Methods

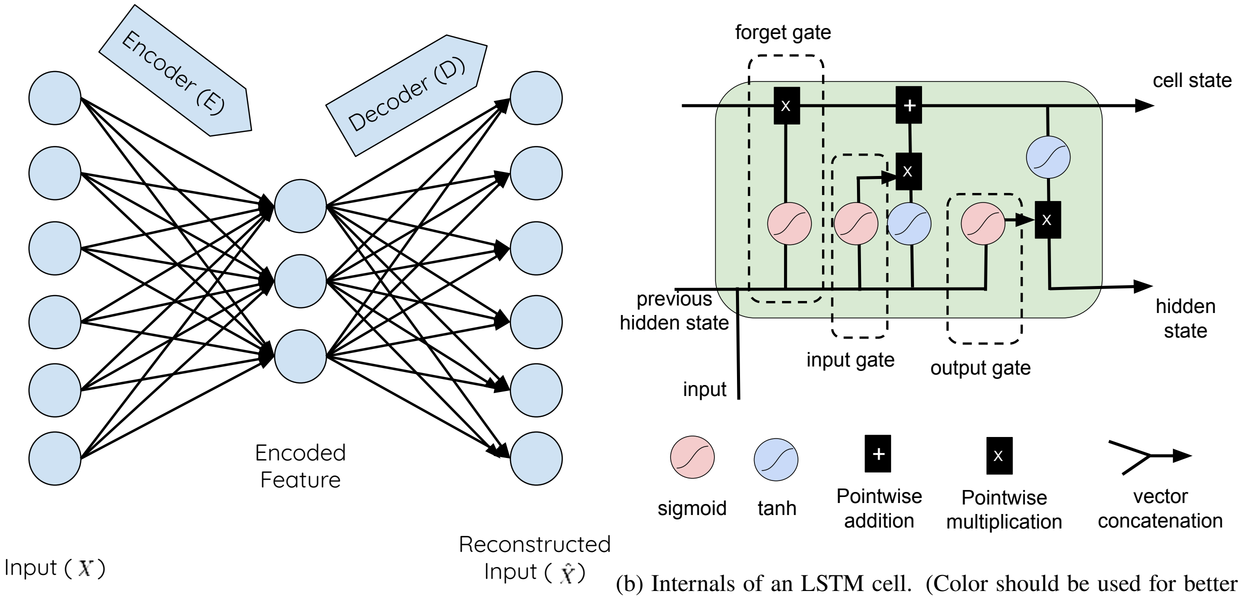
In this section we briefly introduce the Autoencoder and LSTM concepts to the readers before presenting the details of the proposed method in the later sections.

*Autoencoder (AE) Bourlard et al. (1988)* is an example for Feed-forward Neural Network trained in unsupervised fashion. In particular, this neural network contains two parts, an encoder (E) that encodes the input into a lower dimensional representation, a decoder (D) that reconstructs the input by taking the decoder's output as input. This process can be seen as a compressing operation followed by a decompression. The goal in using Autoencoder is to model the data so that it can extract the salient properties of input into a small compressed representation which is sufficient to reconstruct the input back from it. The compressed representation is generally referred to as the 'encoding' or 'encoded feature' and the decompressed representation as the 'reconstructed input'. Figure 1a shows a simple two layer Autoencoder. Note that it operates on 6 dimensional input ($x$) and encodes into a 3 dimensional feature, from which the input is reconstructed back ($\hat{x}$). The parameters of the Autoencoder ($\theta$) (weights connecting the layers) are learned by minimizing a sum of Euclidean ($l_2$) distances between the input $x$ and the reconstructed input $\hat{x}$ over a training set with $N$ samples:

$$\arg\min_{\theta} L = \sum_{i=1}^{N} \left\| x^i - \hat{x}^i \right\|_2^2 \tag{1}$$

where $\hat{x}^i = D(E(x^i))$. In other words, via learning, we attempt to find the optimal weights that minimize the reconstruction loss over the training data. Note that, since this is an unsupervised technique, the label ($y$) is not involved, but only the data samples are sufficient for learning about the data.

*Long Short Term Memory (LSTM)* (Hochreiter et al. (1997); Greff et al. (2016)) is an example RNN commonly used for modeling sequential data in Natural Language Processing (e.g. Sutskever et al. (2014)), Speech Processing (e.g. Sak et al. (2014)) and Computer Vision (e.g. Graves et al. (2008)) applications. It can encode sequential



(a) A simple 2-layer Autoencoder model.

(b) Internals of an LSTM cell. (Color should be used for better visualization of the graphics.)

**Figure 1:** Illustrations of a simple Autoencoder and an LSTM cell.

data by capturing long term dependencies in it. For instance it can be used to read a piece of text (e.g. product review) and encode it into a feature representation without losing the references from the start of the text till the end.

LSTM is equipped with a cell state and a mechanism called gates which can regulate the information flow and learn which data in the sequence is more important to retain and less important to throw away. One can think of the cell state as the memory of the network that carries relevant information during the processing. The cell state can be considered as a pathway that transfers relative information all the way down the input sequence. It is referred to as the memory of the model. Therefore information from even the earlier time steps can be carried to later time steps thereby reducing the effects of short-term memory. Information gets either accumulated to or removed from the cell state via the arrangements known as gates. Gates are different neural network arrangements that determine (via learning) what information is allowed on the cell state and what not. By doing so, LSTM can pass the required information down the long sequences to make accurate predictions.

As illustrated in Figure (1b), LSTM has three gates. The forget gate decides what information needs to be thrown away. Information from the previous hidden state and current input is combined via concatenation (stacking one vector on top of another) and passed through a sigmoid function ($S(x) = \frac{1}{1+e^{-x}}$). This nonlinear activation function squashes the information between 0 and 1, closer to 0 means forgetting and closer to 1 means retaining. Input gate updates the cell state with the current information from the sequential input. Current input and the previous hidden state are passed through a tanh function ($T(x) = \frac{e^{x}-e^{-}x}{e^{x}+e^{-}x}$) to squash between $-1$ and 1. Then this output is multiplied with the sigmoid output which determines which information is important to retain. At this point, the cell state is ready to be updated. Previous cell state gets point-wise multiplied with the forget gate output (for dropping some information) and the output of the input gets point-wise added (for updating with the current input). Finally, the output gate decides what the next hidden state should be. Hidden state contains information from the previous inputs and is used for predictions. Previous hidden state and the current input are passed through a sigmoid before multiplying with the updated cell state which is tanh squashed. Output is the new hidden state. This along with the updated cell state are carried over to the next time instant. In summary the LSTmM operation can be captured by the following control flow.

- First, the previous hidden state is concatenated with the current input. Let us refer to it as the combined signal.
- Combined signal is passed through the forget layer for removing non-relevant information.
- A prospect layer is created from the combined signal which holds the possible values to be passed onto the cell state via addition.

| | Agency 1 | Agency 2 | Agency 3 |
|---|---|---|---|
| # wells | 34 | 5 | 7 |
| # stuck events | 43 | 1 | 5 |
| Drill length (days) | 3382 | 305 | 131 |
| Drill site location | Onshore | Onshore | Offshore |

**Table 1**
Basic information about the drilling data analyzed in this paper.

| S No. | Property | Description |
|---|---|---|
| 1 | TD_spd | Top drive speed |
| 2 | TD_trq | Top drive torque |
| 3 | ROP_avg | Average rate of penetration |
| 4 | Bit_Depth | Depth at which the bit is present |
| 5 | TotDepth | Total depth of the well |
| 6 | Hookheight | Height at which the hook is present |
| 7 | HookLoad | Load on the hook |
| 8 | WOB | Weight on the bit |
| 9 | MRetFlow | Rate at which the mud flow returns |
| 10 | SPP_press | Stand pipe pressure |
| 11 | MPP_SPM 1 | Strokes per minute of Multi phase pump-1 |
| 12 | MPP_SPM2 | Strokes per minute of Multi phase pump-2 |
| 13 | FlowIn | Flow rate from mud pumps into the well |

**Table 2**
Description of the drilling properties considered by the proposed method.

- Combined signal is also fed into the input layer. This layer is intended to decide what information from the prospect should be supplied to the new cell state.
- After computing the forget layer, prospect layer, and the input layer, the cell state is computed using these vectors along with the previous cell state.
- Next, the output is computed from the combined signal.
- Pointwise multiplying this output with the new cell state (computed in the $5^{th}$ item in this list) gives rise to the new hidden state. This gets passed onto the next time instance as previous hidden state.

Since the drilling dataset contain sequences of drilling measurements recorded over time, LSTM is an excellent tool for modelling the sequential drilling data.

## 3. Dataset Details

### 3.1. Data Description

In this paper, we train and evaluate our models on real-world drilling data collected by multiple drilling agencies from multiple drilling wells. Our drilling data consists of several physical properties recorded during the drilling process over time with a resolution of 4 seconds. Table 2 lists the properties considered by the proposed method. Table 1 presents more details about our data including the number of wells, stuck events, drill lengths and site location. Note that we use anonymous names for the agencies and the exact parameters used in this paper. Our data has variety in terms of the formations (geographical variation), sensors used for measurement and other operational conditions. The hole size varies from 6 to 36 inches. Data has been collected using both steerable and rotary steerable systems. Also, various well profiles (vertical, horizontal, and inclined wells) have contributed to the collected data. Depending upon the drilling site (onshore versus offshore) and the agency, list of the recorded properties vary slightly across the three agencies. Therefore, we train the models with a subset of these properties that are common across all the agencies (for smoother cross verification of the models). Along with these drilling properties, the status of drilling process is also recorded to as either 'normal operation' or 'recovery action'. In typical Machine Learning terminology, the properties are referred to as the ‘input’ ($x$), the status as ‘target or label or class’ ($y$) and the tuple $(x, y)$ is referred to as a data sample.

### 3.2. Data Pre-processing

Raw drilling data is pre-processed before training the models. First step is to adjust missing sensor data and fixing apparent errors in the recording. Drilling engineers from the corresponding agencies and academic experts provided the dynamic ranges of the drilling properties for correcting the apparent data errors. Based on this advice, we limit the noisy sensor readings to the specified range.

Similar to any machine learning problem, drilling data is divided into training and testing portions which are non-overlapping. We denote the training data as $N$ sequences $X^1, \ldots, X^N$. Since these sequences are very long (several days), each training sequence is divided into smaller portions using a sliding window (e.g. 1 hour each as shown in Figure 3). This facilitates easy processing and with overlapping division we can create more training data. Hence, each sequence $X^i = \{x_1, x_2, \ldots, x_T\}$ contains $T$ measurements, each measurement $x_i$ is $d$ dimensional, where $d$ is the number of drilling parameters considered for processing. Specifically, we considered dividing the long sequences into 10 minute long portions, which results in 10 minutes × 15 measurements per minute , a total of 150 measurements (since the parameters are recorded every 4 seconds). Each measurement is a vector of 13 ($d$) drilling parameters, hence each portion is $150 \times 13$ dimensional real valued tensor. We collect such 10 minute long portions from all the training drilling sequences which form the training data for the proposed LSTM-AE model.

After that, we perform data normalization. Specifically, we perform min-max normalization (Juszczak et al. (2000)) where in we map the dynamic range of each drilling parameter linearly to $[0, 1]$ range. We have to perform similar data preparation and processing during the testing process.

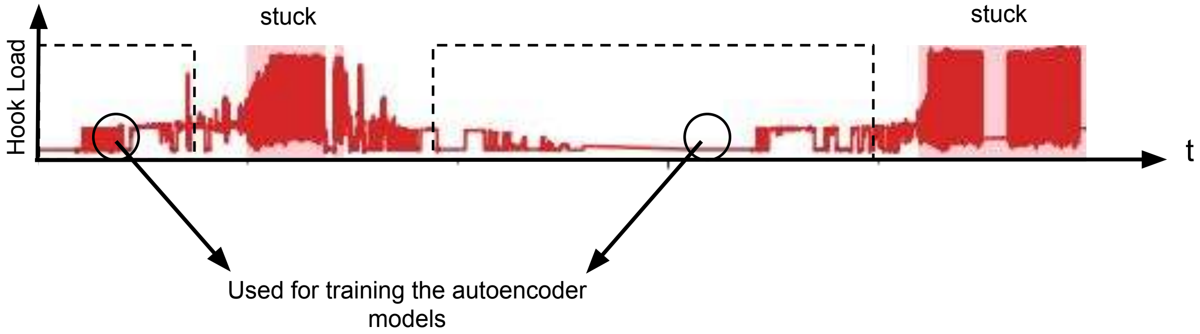


**Figure 2:** Data annotation: Instance of a drilling data (Hook Load) is shown with the ‘stuck’ portions highlighted in light red color. Note that, since the proposed model (LSTM-AE) is an unsupervised deep learning technique, only the normal data portions (shown in dotted black boxes) are used for training.

## 4. Materials and Methods

In this section we detail the proposed method, Autoencoder model for early sign detection for stuck events in the drilling process.

### 4.1. Learning Model Selection

The drilling data (Table 1) has a vast majority of ‘normal’ activity where no stuck events occur. Compared to this ‘normal’ portion, ‘stuck’ portion is very small. This extreme class-imbalance (between ‘normal’ and ‘stuck’ data) makes the supervised learning methods unsuitable (Please refer to section 5 for their performance), because even an unsophisticated model that simply classifies every input as ‘normal’ also achieves high training accuracy. Supervised models typically require the training data to be free of strong class imbalance to learn. Therefore, we work with unsupervised learning tools, particularly the Autoencoders. As introduced in section 2, they attempt to learn data behaviour via reconstruction. Our idea is to train an Autoencoder that learns the behaviour of the 'normal' drilling activity. When an abnormal data related to the 'stuck' event occurs, since such data is not presented to the model, it rises an alarm about the abnormality. Further, in order to read the early signs of stuck, the system should be able to read the complex long-term relations among the drilling parameters, which is achieved in our method via using the LSTMs.

**Input and Output**: Autoencoders take the input data ($X$) and first they encode it into a compact representation known as encoding. Then, they decode it into the size of the input for recovering the original data (represented as $\hat{X}$). Parts of the autoencoder that are responsible for these steps are known as Encoder and Decoder respectively.

Since the proposed LSTM-AE accepts a time series data of the drilling parameters over 10 minutes. Since the drilling data is collected 15 times every minute (i.e. sampling interval of 4 seconds), the input consists of $10 \times 15 = 150$ continuous recordings of the drilling parameters. In all our experiments we used values of 13 such parameters which make the input to be a matrix of 150×13 dimensions. Note that each row represents the values of the drilling parameters recorded at one time instance. This makes the output of the LSTM-AE (which is the reconstructed version of the input) also of the same size. Once the data is reconstructed, our method computes the reconstruction error $(X - \hat{X})^2$ (as mentioned in equations 1 and 4). This error is expected to be more for the data that has the signs of stuck than for the data denoting the normal operation.

## 4.2. Label Refinement

The drilling data has 'normal' and 'recovery' annotations. Since the Autoencoder models are learned to capture the 'normal' behaviour, we train them with the 'normal' data. For instance, Figure 2 shows a portion of labelled drilling parameter (Hook Load) with associated labels. 'Stuck' portions (based on the 'recovery' event reported from DDR) are shaded with light red color and the rest is 'normal' operation. Leaving a safe portion near the 'stuck' portions, the rest of the 'normal' portion is used for training our Autoencoders. These portions are indicated with dotted rectangles in the Figure. While the 'normal' portion just before the stuck region is used for evaluating the models for early sign detection on the subsequent stuck events. In other words, we refine the data labeling to match the task of early prediction as followed:

(i) Normal operation: signal recorded when the normal drilling action is being performed.

(ii) Stuck occurrence: signal recorded at time 't' when we know that there is recovery action during $[t, t + T_r]$.

## 4.3. Training

The principle in the proposed framework is to train an LSTM Autoencoder (LSTM-AE) model to reconstruct the drilling (parameters) data from the normal operation. One can use vanilla Autoencoder for non-sequential data but since we have sequential data and want to perform unsupervised learning, we propose to use LSTM Autoencoder to model the drilling data. It is inspired from the encoder-decoder based language translation model by Sutskever et al. (2014). As shown in the Figure 3, our LSTM Autoencoder model has an encoder LSTM (shown in blue) and a decoder LSTM (shown in green). Encoder LSTM encodes the 1 hour long portions of the sequential multivariate input into a hidden state feature representation (e.g. 128 dimensional vector). This can be represented by the following equation

$$h^i = \text{LSTM}_E(\{x_1^i, \dots, x_T^i\}; \theta_E) \in \text{R}^D \tag{2}$$

From these encoded feature representations decoder attempts to reconstruct the original input, thereby realizing the function of an Autoencoder. This operation, known as decoding (or reconstruction) is represented as follows

$$\{\hat{x}_1^i, \dots, \hat{x}_T^i\} = \text{LSTM}_D(h^i; \theta_D) \tag{3}$$

The learnable parameters of the autoencoder $(\theta_E, \theta_D)$ are learned via minimizing the reconstruction loss shown below

$$\arg\min_{\theta_E;;\theta_D} \sum_{i=1}^{N} \left\| X^i - \hat{X}^i \right\|^2 \tag{4}$$

Memory element present in the LSTM's construction is responsible for sequence to vector transformation (encoding) and vice versa (decoding). Note that the encoder-decoder pair operates continuously on the input in an overlapping sliding-window fashion as shown in Figure 3. It can be loosely thought of as a probe sliding on the training data processing overlapping chunks of it from left to right.

Aim of the proposed method is to detect the stuck related anomalies ahead of their occurrence. However, one can easily notice that the method attempts to reconstruct the current sample $X^t$ at time $t$. We assume (and also observe) that the reconstruction error starts to grow before the actual stuck event. Hence the reconstruction error slightly ahead of the stuck event is a reliable evidence to early predict the stuck.

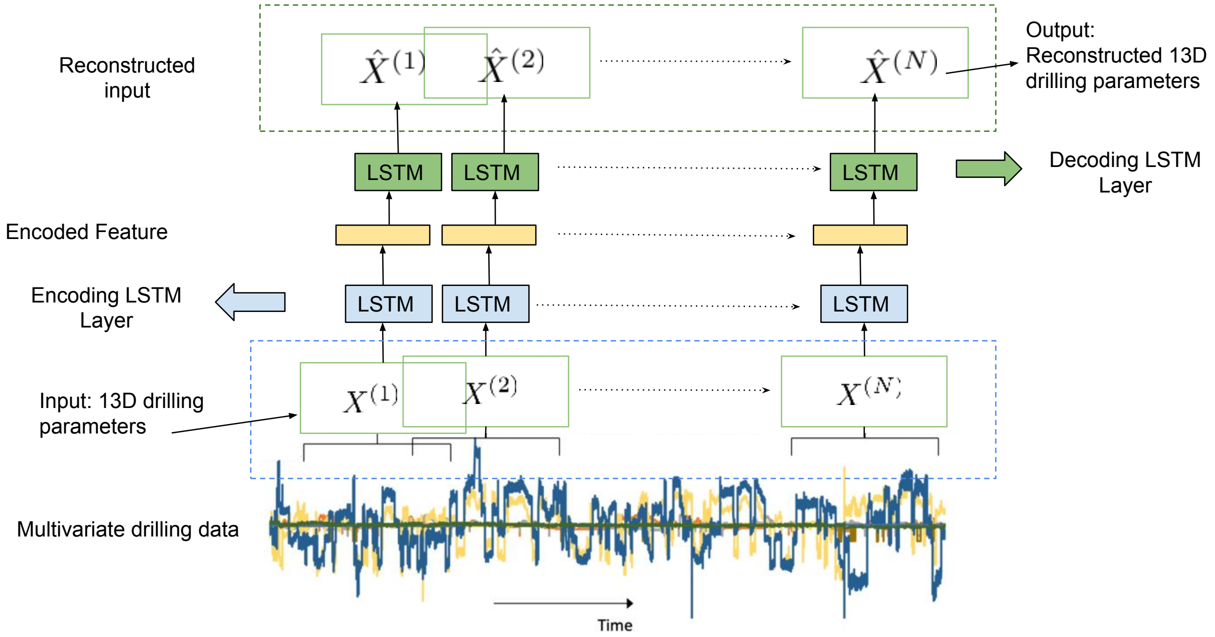


**Figure 3:** Overview of the proposed LSTM-AE model in action for early predicting the stuck events. Multivariate drilling data is processed in a sliding window fashion to encode and decode. When the decoding (reconstruction) fails, the system raises an alarm indicating anomalous activity (stuck). Note that color should be used for better visualization of the graphics.

LSTM-AE is an unsupervised Deep Learning tool that can learn patterns in the data via reconstruction. Hence, we train it on only the annotated 'normal' operation of the train split and test on both operations from other splits. Splits mean the non-overlapping partitions of the data divided for training and testing purposes. This enables to capture the generalizability of the trained model. Training involves learning the weights of the LSTM Autoencoder via iteratively minimizing a loss function similar to Equation 1 and updating the weights (parameters) connecting the network layers.

In our experiments, we processed the drilling data in clips of 10 minutes (sequence length). Our Encoder (E) and Decoder (D) are a single layer LSTMs as shown in Figure 3 and we used an encoding length of $128D$ and a ReLU nonlinearity. We follow a mini-batch stochastic gradient descent (SGD) optimizer with batch size of 50, momentum of 0.9 for computing the gradient of the loss function and update the network parameters. We observed that the training typically requires $50 - 90$ epochs with an initial learning rate of 0.01 and step-wise decaying (division by 10) it after every $20 - 30$ epochs.

## 4.4. Inference

Once the LSTM-AE is trained using the training data, next step is to evaluate its performance on the test split. It is generally referred to as 'inference' or 'testing'. Inference using the trained LSTM-AE model is different from the typical supervised classifiers trained on both the class labels. In this case, the model does not predict the labels directly, instead it attempts to reconstruct the input signal and outputs the reconstruction error. Based on the fidelity of the reconstruction, the label should be inferred. In other words, the reconstruction loss (which is not restricted to be

| | | Prediction | |
|---|---|---|---|
| | | Negative | Positive |
| Target | Negative | True Negative (TN) | False Positive (FP) |
| | Positive | False Negative (FN) | True Positive (TP) |

**Table 3**
Confusion matrix for a classification problem.

in $[0,1]$) also works as the anomaly score output by the model based on which the inference could be derived. If the input drilling parameters are from 'normal' operation, then the reconstruction error is expected to be very small. On the other hand, if the input drilling data is fed from just before the 'stuck' occurrence, the model is expected to output a large reconstruction error indicating a poor reconstruction of the input.

## 4.5. Evaluation Metric

As described above, we use reconstruction error as anomaly score. As the error signal is continuous but not a hard decision based on a threshold, one has to compute the separability of these scores for normal and stuck samples.

Table 3 shows the confusion matrix (Powers et al. (2007)) that presents various types of prediction possibilities for a binary classification scenario. For instance, a True Positive (TP) prediction means that a positive sample correctly classified by the model, and a False Negative (FN) prediction means that a positive sample wrongly classified as negative by the model. Performance characteristics (metrics) of a model are derived from this confusion matrix. For instance, classification accuracy can be computed as follows

$$Accuracy = \frac{TN + TP}{TN + FN + FP + TP} \tag{5}$$

Similarly, other important model characteristics can be obtained as follows

$$\text{True Positive Rate (TPR or Recall)} = \frac{TP}{FN + TP} \tag{6}$$

$$\text{False Positive Rate (FPR)} = \frac{FP}{TN + FP} \tag{7}$$

Note that these metrics depend on the classification threshold we decide for the anomaly scores. However, a fixed threshold gives limited understanding of the model's performance. Instead, it is a common practice to aggregate model's performance at different thresholds via metrics such as Area Under the Curve (AuC) and Average Precision (AP) (Powers et al. (2007)). For instance, Area under the Curve (AuC) is computed as the area under the TPR vs FPR curve drawn by varying the classification threshold from minimum to maximum value. The idea behind AuC is to evaluate model's ability to correctly identify the True Positives as we reduce the False Positives by varying the threshold.

AuC of a classifier is equivalent to the probability that the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance. For the application of anomaly detection, it practically means that if we randomly pick a 'normal' and an 'anomaly (stuck)' signal from our test set, the 'anomaly' one will have a higher score with a probability equal to the AuC. Higher values of AuC are required to be delivered by the deployed system, since we want to make sure that abnormal (stuck) events get highlighted with an anomaly score larger than that of normal events. The AuC value lies in $[0,1]$. An AuC of 1 indicates an ideal anomaly detector that perfectly separates the two classes ('normal' and 'anomaly' events). If the AuC is below 1, that means that some 'normal' events have larger scores (or reconstruction loss) than some of the 'stuck' ones.

# 5. Results

In this section we present our experimental analysis. As noticed from Table 1, data from Agency-1 has the most number of stuck cases. Hence, we focus our analysis mainly on the data from this agency. The data from this agency is divided into 4 splits (A, B, C and D) and cross validation is performed. This means, model is trained on one of the

| Train on | Test on (LSTM-AE) | | | Test on (SVM) | | | Test on (Random Forest) | | |
|---|---|---|---|---|---|---|---|---|---|
| | A1-A | A1-B | A1-C | A1-A | A1-B | A1-C | A1-A | A1-B | A1-C |
| A1-D | 0.57 | 0.76 | 0.84 | - | - | - | - | - | - |
| A1-C | **0.6** | **0.67** | 0.84 | 0.51 | 0.66 | **0.99** | 0.45 | 0.52 | **0.99** |
| A1-B | 0.67 | 0.6 | **0.84** | 0.73 | **0.99** | 0.47 | **0.79** | **0.99** | 0.69 |
| A1-A | 0.72 | 0.66 | **0.85** | **0.99** | 0.65 | 0.22 | **0.99** | 0.65 | 0.62 |

**Table 4**
Performance of the proposed LSTM-AE model for early sign detection in terms of AuC. Table presents cross validation on different splits of Agency-1's drilling data obtained by the proposed (LSTM-AE) method and the well-known supervised approaches SVM, and Random Forest respectively. Note that split D has no stuck events and hence is omitted from testing. However since the proposed method is unsupervised, we can train the LSTM-AE on it. Further, note that the AuC performance of supervised method using 1D-CNNs is 0.53, the crude baselines of predicting same category for every input and predicting a random probability for stuck are both 0.5. Best results for each test set (sub-column) is shown in bold. Although the supervised setting achieves best performance on the training data (diagonal elements in each method), they fail to generalize well (except one instance) onto the held out data (test set). This can be observed looking at the non-diagonal numbers in each classifier column.

| Train on | Test on | | | |
|---|---|---|---|---|
| | A1-A | A2-B | A3-C | A4-D |
| A2 | 0.6 | 0.63 | 0.68 | - |

**Table 5**
Performance (in terms of AuC) of our LSTM-AE model trained on data from Agency-2 when tested on different splits of Agency-1 data. Note that split D has no stuck events.

4 splits and tested on the rest (3), this process is repeated 4 times by changing the train split every time. Note that the split D has no stuck events, hence the model is not evaluated on that split. Also, we cannot train supervised models on that for the same reason.

In all these experiments, we infer about the ‘Stuck’ events that occur $[0, 6]$ minutes in the future of the currently processed clip. In other words, $T_r$ is 6 minutes and the models are evaluated to predict the stuck cases that occur anytime between now and 6 minutes from now. Table 4 shows the AuC computed on the train and test splits. We can observe that in majority of the cases, our models exhibit excellent detection. In some cases, the models achieve high AuC value of 0.85.

In order to compare the proposed unsupervised approach, we also trained supervised classifiers: Support Vector Machine (SVM), Random Forest, and a 3 layer Convolutional Neural Network (CNN) trained on the same input features (vector of 13 dimensions) as the LSTM-AE. While learning with supervision, in case of data imbalance, there are two strategies typically followed, (i) undersampling, and (ii) oversampling. Undersampling refers to discarding the samples of the majority class samples thereby achieving the balance between the two classes. This process may remove samples that are useful, often critical to learn a robust classification boundary. On the other hand, oversampling involves duplicating minority class samples and adding to the training data in order to improve the balance. Since it uses all the data and balances the number of samples from both the classes, we followed oversampling in our supervised experiments.

Further, we evaluated the proposed approach against a couple of unsophisticated baselines. First is classifying every input to one (or other) of the two categories, called All $1/0$ classifier. Second is randomly assigning a uniform probability in $[0, 1]$ range for the input to be stuck. From Table 4 one can notice that SVM and Random Forest methods achieve the best performance on the training data (i.e, trained and tested on the same subset; diagonal elements of the last 3 rows in each classification method). However, they fail to generalize well onto the held out data (test set). This can be observed looking at the non-diagonal numbers in each classifier column. It demonstrates that in cases where the training data is heavily imbalanced, supervised techniques are prone to overfitting to the training data and because of which they may fail to generalize well onto the unseen test data. On the other hand, unsupervised techniques are less prone to such risk. This clearly highlights the necessity of a sophisticated unsupervised technique such as the proposed framework.

We have also conducted a generalization experiment across the agencies. We have trained an LSTM-AE model on the data from Agency-2 and evaluated on the splits of Agency-1. Table 5 presents the AuC performance of such a model. It is clear that the model's performance is consistently above 0.6 and hence generalizes across these datasets. However, in some cases (such as Agency-3) we observed that the dynamic ranges of some of the drill properties have significant differences which may affect the models' generalization ability to such datasets.

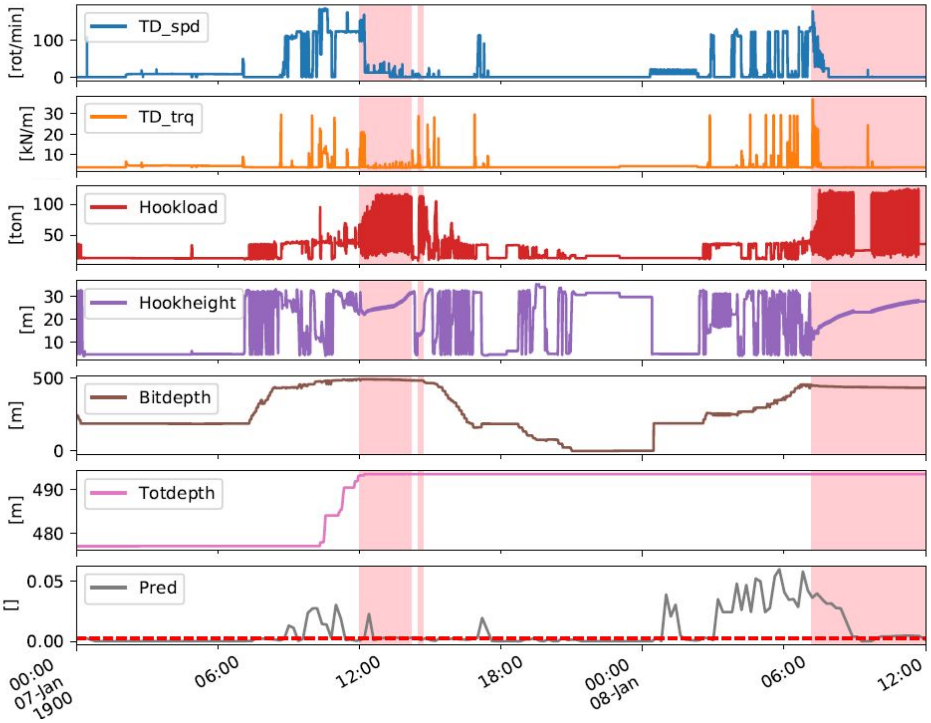


**Figure 4:** Case study 1: Model successfully detects early signs of a 'pack-off' stuck event. Bottom panel shows the stuck risk (anomaly score or reconstruction error) predicted by the model while the rest of the panels show some of the parameters recorded during the drilling. (Color should be used for better visualization of the graphics.)

## 6. Discussion

In this section we analyze the trained models in terms of reconstruction error computed from their output. It is very important for us to be able to interpret the learned AI models, failing to do so may lead to their acceptance just as black boxes. However, at this point in time, providing human-level interpretation for the Neural Network Models has not matured and is a very active ongoing research interest among the AI community.

With the help of drilling engineers from the corresponding agencies, we consider data instances from interesting (abnormal) periods of drilling in order to analyze the models. The aim of this analysis is to provide explanations for the model's predictions and to extract meaningful interpretations. Our analysis revealed very interesting observations about the model's operation. For instance, the model can identify the responsible properties which might have triggered the alarm. In other words, in case of a stuck event, apart from alerting ahead of its occurrence, our model can help the on-site drilling engineers with providing the drilling parameters to look at. Such information could be vital for the experts in order to quickly assess the situation and handle it effectively.

(i) Case Study 1: Figure 4 shows the drilling data collected by Agency-1 (test split C) and the real-time stuck risk prediction (reconstruction error or anomaly score) by the trained model (on train split D). The plot shows only a selected set of drilling parameters in the top panels and the stuck risk prediction in the bottom panel. One can observe that the regions that are labelled as 'stuck' are shaded in light red color. There are two stuck events in the plotted data. Our model outputs higher scores ahead of the stuck event occurrence, thereby successfully performing early sign detection. This portion of the data is labelled as 'pack-off' stuck by the experts during the drilling. Hence, this

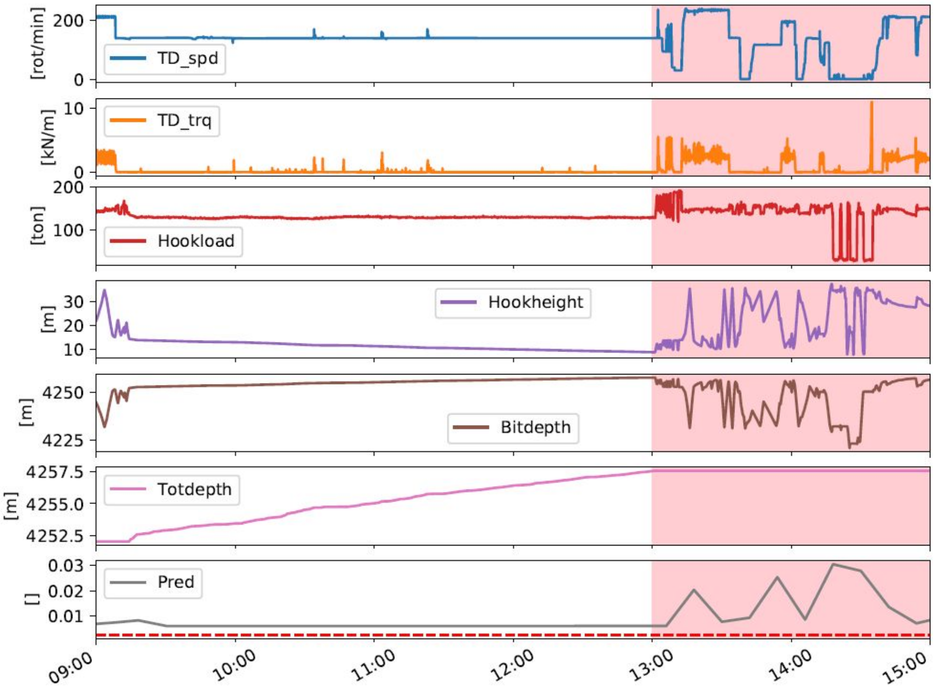


**Figure 5:** Case study 2: Model identifies a 'differential stuck' event. However, it fails to respond ahead of the occurrence. Bottom panel shows the stuck risk output by the model, while the rest show some of the drilling parameters. Note that the stuck period is highlighted in light red color. (Color should be used for better visualization of the graphics.)

case is a successful detection of a pack-off stuck scenario by the trained model. A pack-off stuck means to plug the wellbore around the drill string. Small pieces of the formation settle around the drill string and pack off the annulus surrounding it. This may happen for a variety of reasons, however the most common being that either the drilling fluid is not properly fetching the resulting cuttings out of the annulus or portions of the wellbore wall collapse around the drill string.

We can observe that for the majority of the time, reconstruction loss to be very low during the normal operation till the first half of day 7. However, shortly before the recovery action (which is a response to the stuck scenario) in the second half of the day 7, reconstruction error output by our model clearly shows the signs of an anomaly. Bottom panel of the figure shows the reconstruction error. In this panel, the mean reconstruction error over the training data (normal) is shown in dotted red line. Note that even during testing, the reconstruction error for normal operation is very close to the mean training error, indicating better generalization of the model across the splits.

(ii) Case study 2: In this case ( Figure 5) we demonstrate an instance where the model's behaviour falls short of satisfactory. In this case, the model attempts to detect a 'differential' stuck labeled by the experts in the test split A. Differential stuck (Reid et al. (2000)) is a condition where the drill string cannot be moved along the axis of the wellbore. This typically occurs when high-contact forces caused by low reservoir pressures and/or high wellbore pressures are exerted over a sufficiently large area of the drill string.

One can observe that the recovery action is highlighted in light red and the bottom panel shows the reconstruction error output by the model. We can observe that the reconstruction error clearly shoots up during the stuck period, thereby indicating abnormality. However, the model doesn't respond ahead of the stuck occurrence, which is desired. That way, the model is only partially successful in this case. Further, we notice that the reconstruction error is consistently more (even for the normal operation) than the mean reconstruction error computed on the train split (D) shown in the dotted red line.

## 6.1. Providing Explanations for Model's Predictions

Here we analyze the model's output in terms of its reconstruction error. This is one way for probing into it's operation and seeking explanations for its predictions. For instance, at any point in time, we can identify the important

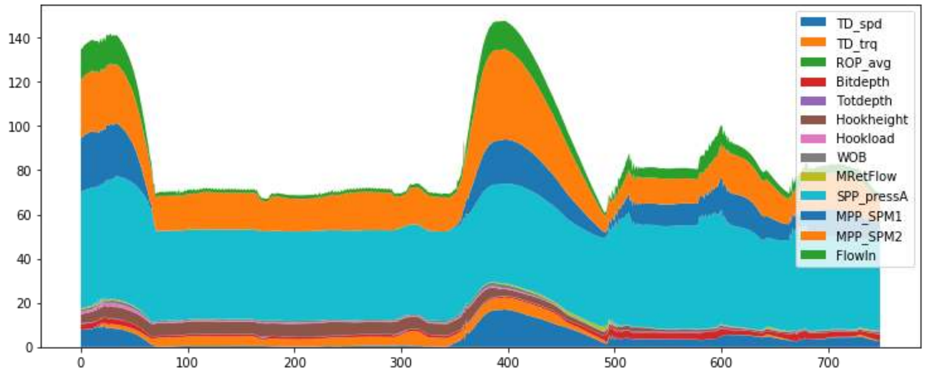


**Figure 6:** Analysis of the model's prediction during a pack-off stuck in terms of the reconstruction analysis. Stack plot of the individual reconstruction errors for the 13 drilling properties during the first 1 hour drilling duration extracted from split C. X-axis is the time and every unit measures 4 seconds (sample interval). The learned model starts to predict continuously after the first 10 minutes (150 samples), therefore the indices span from 0 to $900 - 150 = 750$. (Color should be used for better visualization of the graphics.)

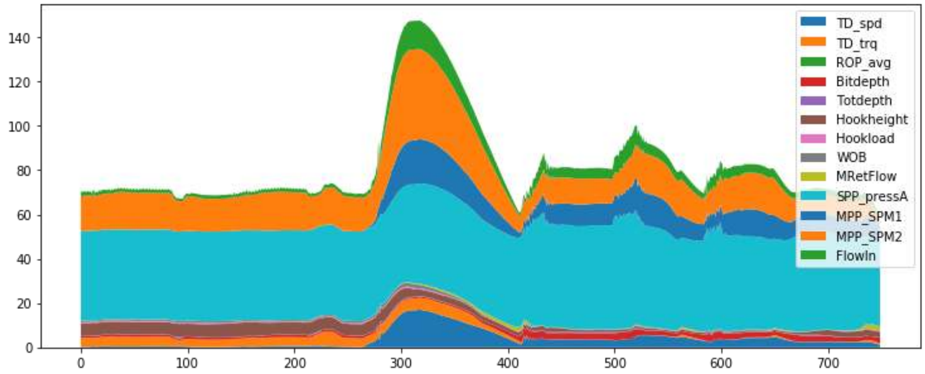


**Figure 7:** Analysis of the model's prediction during a 'pack-off stuck' in terms of the reconstruction analysis. Stack plot of the individual reconstruction errors for the 13 drilling properties during the second 1 hour drilling duration extracted from split C. Note that the x-axis is the time and every unit measures seconds (sample interval). The learned model starts to predict continuously after the first 10 minutes (150 samples), therefore the indices span from 0 to $900 - 150 = 750$. (Color should be used for better visualization of the graphics.)

properties that majorly contribute to the reconstruction error (quantity computed in eq. 1 and 4). In other words, we can find the drilling properties that cause the model to fire when an 'anomaly' is encountered during the drilling.

Specifically, at every instant, the system reads the drilling parameters (13 variables) and attempts to reconstruct it. If it is successful in reconstructing with very small error, the system infers that the input is normal drilling data. However, if the reconstruction error is strikingly more, the system infers that the observed drilling data is abnormal or anomaly. This is when it alarms the drilling crew about a possible stuck scenario. The reconstruction loss is computed based on all the constituting drilling parameters. Each of them contributes to the total error differently. In case of a stuck scenario, it may be possible to understand more about the nature of the stuck if we know which of the drilling parameters are behaving abnormally. In other words, at that point in time which of them are strongly contributing to the reconstruction error.

We consider three specific 'stuck' cases in the test drilling data (split C) identified by the experts (two of which are discussed in the previous subsections): (i) pack-off, (ii) Bad-hole condition, and (iii) differential.

(a) Pack-off case: Here, we examine the drilling data labelled as an instance of 'pack-off' stuck scenario by the drilling experts. Note that this data is part of testing split C. We encode and decode the multivariate drilling data by sliding over it as shown in Figure 3. At every time instant (4 seconds) we computed the reconstruction error incurred

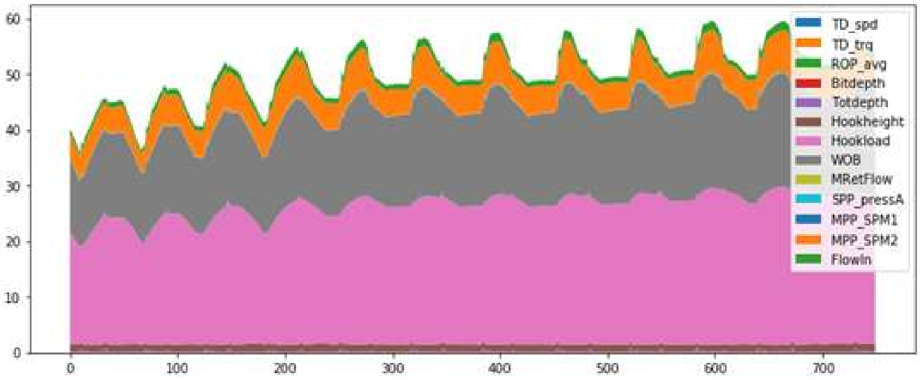


**Figure 8:** Analysis of the model's prediction during a 'Bad Hole Condition', in terms of the reconstruction analysis. Stack plot of the individual reconstruction errors for the 13 drilling properties during the first 1 hour drilling duration extracted from split C. Note that the x-axis is the time and every unit measures seconds (sample interval). The learned model starts to predict continuously after the first 10 minutes (150 samples), therefore the indices span from 0 to $900 - 150 = 750$. (Color should be used for better visualization of the graphics.)

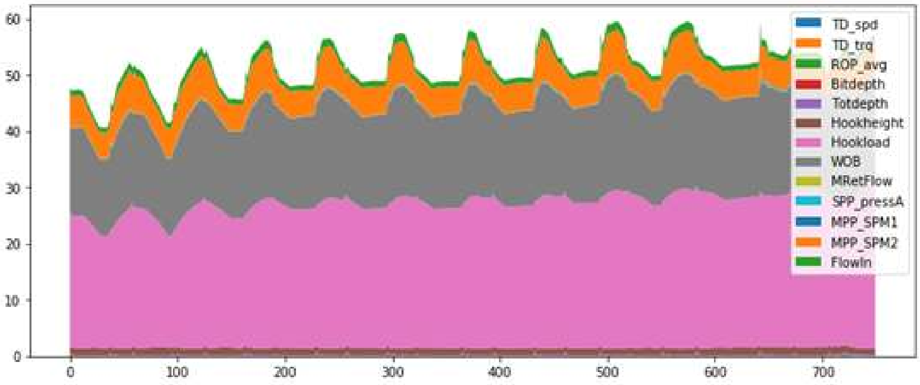


**Figure 9:** Analysis of the model's prediction during a 'Bad Hole Condition', in terms of the reconstruction analysis. Stack plot of the individual reconstruction errors for the 13 drilling properties during the second 1 hour drilling duration extracted from split C. Note that the x-axis is the time and every unit measures seconds (sample interval). The learned model starts to predict continuously after the first 10 minutes (150 samples), therefore the indices span from 0 to $900 - 150 = 750$. (Color should be used for better visualization of the graphics.)

by the individual properties (13 in total). Figures 6 and 7 show these individual error contributions as stack plots for two 1 hour durations extracted from the long stuck period. Note that these portions are extracted from the second stuck instance shown in Figure 4.

In these plots X-axis denotes the time for 1 hour duration. Since the data is collected with 4 seconds time interval, there would be 15 time instances per minute and 900 instances per hour. Note that, the model processes and reconstructs every 10 minutes duration at a time. Hence, continuous sliding of the model happens only from $10^t h$ minute till the $60^{th}$ minute. Therefore, the plot has only 750 time instances instead of 900. Y-axis represents the reconstruction error contributed by each of the drilling parameters at a particular instant of time. Note that each parameter is represented in a different color. At each instance, the height of a particular color represents the contribution of the corresponding parameter. If it is tall, that means that parameter is strongly contributing to the error at that point in time. If it is short, that means the corresponding parameter contributes less to the error, indicating that may not be the reason for the anomaly. This allows us to identify the possible reasons behind the specific stuck/anomaly.

In this case (Figures 6 and 7), the top 5 dominant properties are identified as SPP_pressA, TD_trq, TD_spd, Hook height, and MPP_SPM1 throughout the stuck period. Interestingly, the experts had identified that the SPP pressure was large and recorded the same in the daily drilling report. Therefore, the proposed AI system not only detected the early signs, but also correctly identified the associated drilling parameters.

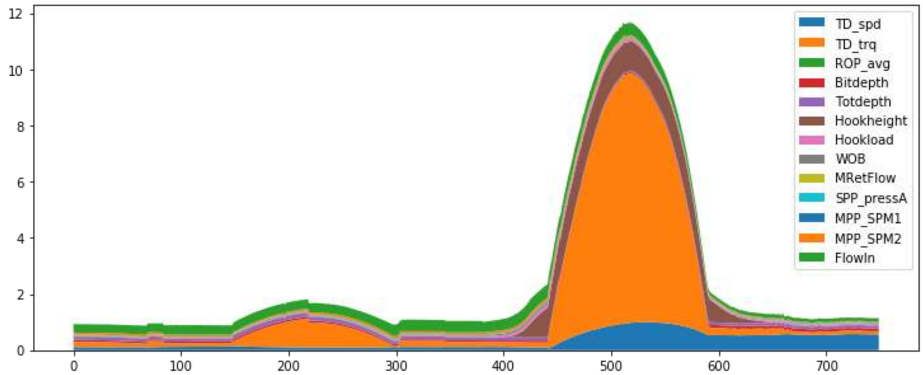


**Figure 10:** Analysis of the model's prediction during a 'Differential Stuck', in terms of the reconstruction analysis. Stack plot of the individual reconstruction errors for the 13 drilling properties during the first 1 hour drilling duration extracted from split C. Note that the x-axis is the time and every unit measures seconds (sample interval). The learned model starts to predict continuously after the first 10 minutes (150 samples), therefore the indices span from $0$ to $900 - 150 = 750$. (Color should be used for better visualization of the graphics.)

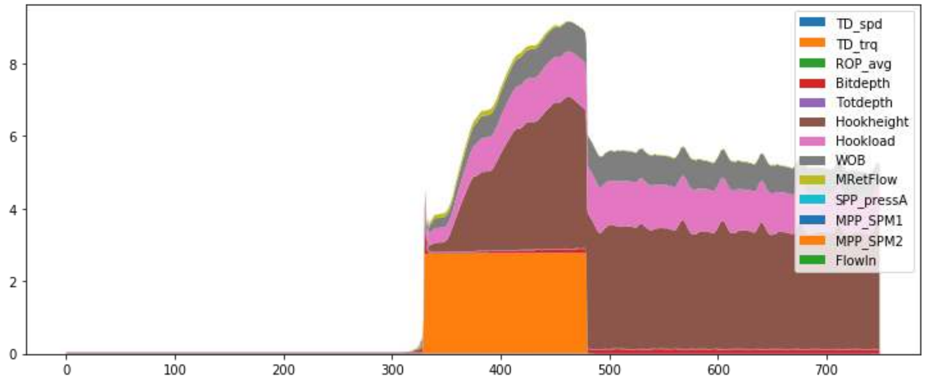


**Figure 11:** Analysis of the model's prediction during a 'Differential Stuck', in terms of the reconstruction analysis. Stack plot of the individual reconstruction errors for the 13 drilling properties during the second 1 hour drilling duration extracted from split C. Note that the x-axis is the time and every unit measures seconds (sample interval). The learned model starts to predict continuously after the first 10 minutes (150 samples), therefore the indices span from $0$ to $900 - 150 = 750$. (Color should be used for better visualization of the graphics.)

(b) Bad hole Condition: Here, we have analyzed the drilling data from a 'Bad hole condition' scenario. A bad hole scenario is when the borehole is not to gauge or is corrugated. This scenario generally refers to the detrimental effect that such a borehole has on the response of logging measurements.

Similar to the previous stuck case, we computed the error contributions for two separate 1 hour durations from the lablled data. Figures 8 and 9 show the stack plot of the error contributions for the 13 individual drill properties. We have observed no difference in the relative ordering of these properties throughout the stuck period. The top−5 contributors for the error are identified as Hook Load, WOB, MPP_SPM2, Hook Height, and FlowIn.

(c) Differential Stuck: We have analyzed the drilling data recorded during a 'Differential stuck' period of split C test data (these are the clips extracted from the recovery region highlighted in Case study 2). The error analysis reveals in Figures 10, 11, and 12 that there is a transition in the top individual drilling properties contributing to the reconstruction error during the stuck period.The top contributors in the early region (Figure 10) are TD_trq, TD_spd, FlowIn, Tot_depth, MPP_SPM2, where as after the transition, the top contributors are observed (Figure 12) to be

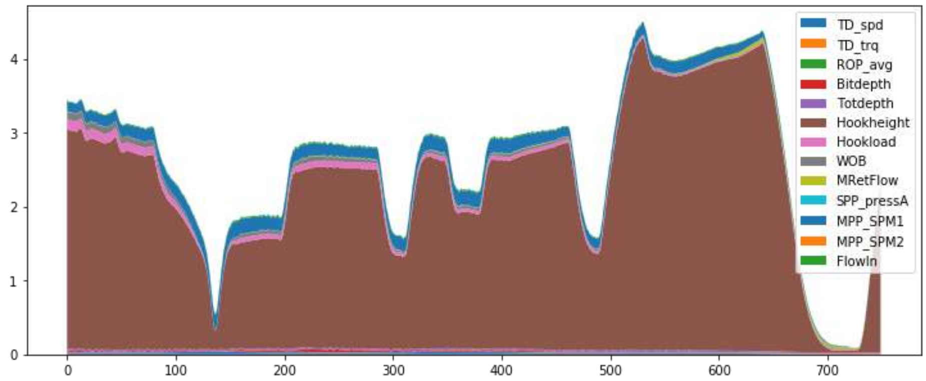


**Figure 12:** Analysis of the model's prediction during a 'Differential Stuck', in terms of the reconstruction analysis. Stack plot of the individual reconstruction errors for the 13 drilling properties during the third 1 hour drilling duration extracted from split C. Note that the x-axis is the time and every unit measures seconds (sample interval). The learned model starts to predict continuously after the first 10 minutes (150 samples), therefore the indices span from 0 to $900 - 150 = 750$. (Color should be used for better visualization of the graphics.)

Hook Height, MPP_SPM1, Hook Load, WOB, and TD_spd. Such real-time and dynamic explanations provided by the model during the drilling can be very helpful in better assessment of the well and drilling process.

### 6.2. Performance of the LSTM Autoencoder

Next we discussed the performance of our unsupervised LSTM Autoencoder model. The primary motivation for designing an unsupervised approach is the lack of sufficient number of stuck events in the collected drilling data. There are various mechanisms of stuck, thus there is not a single pattern in a single parameter indicating all of stuck pipe incidents [Alshaikh et al. (2018)]. Given the variety in stuck cases, we need sufficient amount of positive data to learn from, which is tedious and costly to collect. With a huge imbalance in the positive and negative data, supervised models can not be learned reliably. Also, in order to label a stuck case in advance (for early detection), it is very unclear as to how much in advance one should label. Our preliminary experiments with supervised models also demonstrated only tiny improvements from random label assignment baseline (AuC of 0.5). We believe, one has to incorporate valuable domain knowledge about the drilling data in order to reliably capture the patterns in the limited positive data. For instance, it can be achieved in the form of an effective feature engineering on the raw drilling data with the help of drilling experts.

However, we gathered large amounts of drilling data collected during the normal operation. Hence, it is possible to capture all the variety in it, thereby enabling to reliably learn the normal operation.

## 7. Conclusions

For problems such as early stuck sign detection, where data is severely imbalanced, supervised machine learning techniques are not suitable. Therefore, we proposed an unsupervised alternative in the form of an LSTM-AE. Our model is strongly motivated since (i) it can read the long term correlations that exist among various drilling parameters through which the early sign for stuck may manifest, and (ii) it trains on only one kind of (normal) data avoiding the imbalance issue. Our experimental results demonstrated that the model is very promising. To the best of our knowledge, this is the first work that applies an unsupervised deep learning technique for detecting early signs of stuck and opens up a new direction for future researchers.

In general, the response of the anomaly detector to the both normal and abnormal (stuck) scenarios is observed to be satisfactory. We have analysed field well data collected from 30 wells in our experiments. In terms of the quantitative performance, we consistently observed significantly better performance over the random assignment baseline ($AuC =$ 0.5). Also, in some of the test splits the AuC performance has been as high as 0.85. The reconstruction error analysis also reveals interesting observations about the learned model's ability to infer based on the reconstruction error. For instance, in cases such as the pack-off stuck scenario, the properties that dominated the reconstruction error are also observed to be strong by the drilling experts as well. For future work, we would like to explore the use of hybrid supervised and unsupervised models that can exploit both labelled and unlabelled data. In addition, we would like to study unsupervised learning at multiple temporal scales, e.g. different models looking at short and long-term temporal data.

## 8. Acknowledgements

This research was supported by a project funded by JOGMEC (Japan Oil, Gas and Metals National Corporation).

## References

Agwu, O. E., Akpabio, J. U., Alabi, S. B., and Dosunmu, A. (2018). Artificial intelligence techniques and their applications in drilling fluid engineering: A review. Journal of Petroleum Science and Engineering

Allen, G., Andreoni, I., Bachelet, E., Berriman, G. B., Bianco, F. B., Biswas, R., Kind, M. C., Chard, K.,Cho, M., Cowperthwaite, P. S., Etienne, Z. B., George, D., Gibbs, T., Graham, M., Gropp, W., Gupta,A., Haas, R., Huerta, E. A., Jennings, E., Katz, D. S., Khan, A., Kindratenko, V., Kramer, W. T. C.,Liu, X., Mahabal, A., McHenry, K., Miller, J. M., Neubauer, M. S., Oberlin, S., Jr, A. R. O., Rosofsky,S., Ruiz, M., Saxton, A., Schutz, B., Schwing, A., Seidel, E., Shapiro, S. L., Shen, H., Shen, Y., Sipőcz,B. M., Sun, L., Towns, J., Tsokaros, A., Wei, W., Wells, J., Williams, T. J., Xiong, J., and Zhao, Z.(2019). Deep Learning for Multi-messenger Astrophysics: A Gateway for Discovery in the Big data Era. arXiv preprint arXiv:1902.00522

Alshaikh, A., Magana-Mora, A., Gharbi, S. A., Al-Yami, A., et al. (2019). Machine Learning for Detecting Stuck Pipe Incidents: Data Analytics and Models Evaluation. In International Petroleum Technology Conference. International Petroleum Technology Conference.

Alshaikh, A. A., Albassam, M. K., Al Gharbi, S. H., Al-Yami, A. S., et al. (2018). Detection of Stuck Pipe Early Signs and the Way Toward Automation. In Abu Dhabi International Petroleum Exhibition and Conference. Society of Petroleum Engineers.

Bourlard, H. and Kamp, Y. (1988). Auto-association by multilayer perceptrons andsingular value decomposition. Biological Cybernetics, 59, 291–294.

Elmousalami, H.H., Elaskary, M. Drilling stuck pipe classification and mitigation in the Gulf of Suez oil fields using artificial intelligence. J Petrol Explor Prod Technol 10, 2055–2068 (2020).

Godil S.S., Shamim M.S., Enam S.A., Qidwai U. (2011). Fuzzy logic: A "simple" solution for complexities in neurosciences?. Surgical Neurology International, 2(24). doi:10.4103/2152-7806.77177

Graves, A., Liwicki, M., Fernández, S., Bertolami, R., Bunke, H., and Schmidhuber, J. (2008). A Novel Connectionist System for Unconstrained Handwriting Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 31(5):855–868.

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., and Schmidhuber, J. (2016). LSTM: A Search Space Odyssey.IEEE Transactions on Neural Networks and Learning Systems, 28(10):2222–2232.

Heinze, L., and Al-Baiyat, I. A. (2012). Implementing Artificial Neural Networks and Support Vector Machines in Stuck Pipe Prediction. Society of Petroleum Engineers. doi:10.2118/163370-MS

Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9:1735–80.

Jahanbakhshi, R., Keshavarzi, R., Aliyari Shoorehdeli, M., Emamzadeh, A., et al. (2012). Intelligent Prediction of Differential Pipe Sticking by Support Vector Machine Compared with Conventional Artificial Neural networks: An Example of Iranian Offshore Oil Fields. SPE Drilling and Completion, 27(04):586–595.

P. Juszczak, D. M. J. Tax and R. P. W. Duin (2002). Feature scaling in support vector data description. AAAI Technical Report, WS-00-05

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with Deep Convolutional Neural Networks. In Advances in Neural Information Processing Systems 25.

Lind, Y. B., Kabirova, A. R., et al. (2014). Artificial Neural Networks in Drilling Troubles Prediction. In SPE Russian Oil and Gas Exploration & Production Technical Conference and Exhibition. Society of Petroleum Engineers.

Lye, K. O., Mishra, S., and Ray, D. (2020). Deep Learning Observables in Computational Fluid Dynamics. Journal of Computational Physics, 410:109339.

Magana-Mora, A., Gharbi, S., Alshaikh, A., Al-Yami, A., et al. (2019). Accupipepred: A Framework for the Accurate and Early Detection of Stuck Pipe for Real-time Drilling Operations. In SPE Middle East Oil and Gas Show and Conference. Society of Petroleum Engineers.

Muqeem, M. A., Weekse, A. E., Al-Hajji, A. A., et al. (2012). Stuck Pipe Best Practices - A Challenging Approach to Reducing Stuck Pipe Costs. In SPE Saudi Arabia Section Technical Symposium and Exhibition. Society of Petroleum Engineers.

Powers, D.. “Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness and Correlation.” Technical report SIE-07-001.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning Representations by Back-propagating Errors. Nature, 323(6088):533–536.

Sak, H., Senior, A., and Beaufays, F. (2014). Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling. In INTERSPEECH.

Salminen, K., Cheatham, C., Smith, M., Valiullin, K., et al. (2017). Stuck-pipe Prediction by Use of Automated Real-time Modeling and Data Analysis. SPE Drilling and Completion, 32(03):184–193.

Shadizadeh, S. R., F. Karimi, and M. Zoveidavianpoor. "Drilling stuck pipe prediction in iranian oil fields: an artificial neural network approach." Iranian Journal of Chemical Engineering 7, no. 4 (2010): 29-41.

Siruvuri, C., Nagarakanti, S., Samuel, R. Stuck Pipe Prediction and Avoidance: A Convolutional Neural Network Approach. Society of Petroleum Engineers (2006). doi:10.2118/98378-MS.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. InAdvances in neural information processing systems, pages 3104–3112.

Tripathi A. M., DuttaBaruah R., Subbiah S., Oil well drilling activities recognition using a hierarchical classifier, Journal of Petroleum Science and Engineering, (pre-proof), 2020.

Ramba V., Selvaraju S., Subbiah S., Palanisamy M., A robust anomaly detection methodology using predicted hookload and neutral point for oil well drilling, Journal of Petroleum Science and Engineering, 2020.

Reid, P.I., Meeten G.H., Clark P., Chambers B.D., Gilmour A., Sanders M.W. (2000). Differential-Sticking Mechanisms and a Simple Wellsite Test for Monitoring and Optimizing Drilling Mud Properties. SPE Drilling and Completion - SPE DRILL COMPLETION. 15. 97-104. 10.2118/64114-PA.

Van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior,A. W., and Kavukcuoglu, K. (2016). Wavenet: A Generative Model for Raw Audio. arXiv preprint arXiv:1609.03499.

Young, T., Hazarika, D., Poria, S., and Cambria, E. (2018). Recent Trends in Deep Learning Based Natural Language processing. IEEE Computational Intelligence magazine, 13(3):55–75, 2018.

Zhu, Q., Wang, Z., Huang, J. (2019, October 21). Stuck Pipe Incidents Prediction Based On Data Analysis. Society of Petroleum Engineers, 2019.