Application of Deep Learning Techniques For Drilling Predictions

Ouassim Khebzegga[1]

Abstract— **We apply deep Recurrent Neural Network to predict hydrocarbon well's drilling states (drilling/non-drilling time) based on a set of measurements collected on a drilling rig during operations. We use a series of LSTM models with a softmax output layer to identify a rig state based on a combination of 9 signals (torque, RPM, hook-load, weight on bit,...). Our work aims at automating this classification as it is at the core of drilling performance evaluation and is still largely carried out manually by drilling engineer.**

I. Introduction

Drilling is at the core of hydrocarbon resources production. It is the most capital intensive and riskiest process in the development of an oilfield. Modern wells extend beyond tens of thousand of feet under the surface at extreme pressures and temperatures. In spite of these conditions, modern wells can now be drilled in a few days. Advances in the industry are mostly driven by the very high cost of operations. These can go from a few tens of thousands of dollars per day for onshore factory wells to over a million dollars per day for the most complex offshore wells. The goal of this work is to automate the identification of the well's drilling states (drilling/non-drilling time) using sensors data collected during the drilling process. This identification is at the core of performance analysis performed daily by operators across the US. Over the recent years, engineers have managed to drastically reduce the overall time/cost of drilling a well. They have managed to do so by thorough monitoring of each sequences occurring during the drilling process. The current work uses the data collected by an operator in the US over the past 4 years on a set of 25 wells in two unconventional basins. Various sensors were installed at the surface and down-hole to track the state of the drilling bits with high resolution (up to 100 Hz) and accuracy. Quantities such as torque, rotations per minute, weight on bit, hook-load, pressures and flow rate are primarily used to inform drilling engineers on the state of operations. A trained eye can easily distinguish in what state the rig is based on the raw signal captured by its monitoring system. We present an approach to automate this classification in order to speed up the overall performance diagnostic of a well.

II. Problem Statement

Formally, the well's status prediction goal is a multivariate time series, two-class classification task. The input data is a 130 (or 68) time series representing the measurements of different sensors during the drilling process. We have a total of 25 wells. The goal is to use a subset of the 130 measurements to predict the states of the drilling process, basically we want to know for each time step what is the current state bases on the previously measured data. The LSTM type of RNN's is the most adequate neural network architecture for this type of problems.

[1]PhD Candidate, Department of Energy Resources Engineering at Stanford

III. Data

The dataset is composed of sensors' measurements for 25 wells, The frequency of the measurements is 100Hz, and the drilling process can last for few days. For each well, the data is in the form of a table where the columns represent the features and the rows represent time steps. The table contains millions of rows due to the high frequency of the measurements. A total of 7 out of 25 wells have 130 features and the remaining 18 wells have 68 features. Not all these features represent sensors measurements, some of them are recalculated using the sensors measurements. Due to the use of different service providers for the well drilling (different phases), there is redundancy in some of the sensors measurements. Also, the size of the data is subsequent (2 TB), composed of relatively huge single files, which makes the operations of data loading, handling and saving very challenging, and resulted in the allocation of an important amount of time to the cleaning and processing of the raw data, this part will be presented in the following subsections:

A. Data Cleaning

First, we needed to identify the most relevant features to our task, so we had to define a standard for the importance of each feature and this resulted in the elimination of :

1) **Redundant features:** as we explained previously, some of the measurements represented the same physical property (ex. torque) but were carried out by different drilling service providers which resulted in different names for the same feature.
2) **Calculated features:** these features are calculated taking as input the data from the sensors (genuine features). These calculations were realized by the drilling engineers, and despite the added value we preferred to remove them as we were not sure if these calculated measurements will always be present in future dataset and we wanted our model to generalize well to future data.

Fig.1 shows a classification of the features into: sensors features (genuine), redundant features and calculated features. We can see that the fraction of the sensors features of interest for us is only 60%.

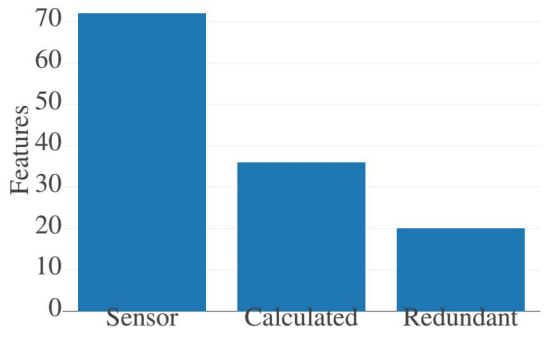

Fig. 1. Distribution of the input features

B. Data Smoothing

The frequency of the measurements is relatively high, which resulted in the time series to be very noisy, so in order for us to prepare them for the training phase we had to apply:

1) **Filtering:** using an averaging sliding window of 100 time step we smoothed the data, this resulted in a relatively smooth data without compromising its variations. The choice of 100 time step was specifically held because it averages over a second which is the smallest time step of interest for us.
2) **Re-sampling:** based on the smoothed time series we re-sampled the data by taking a measure each 100 time steps, this resulted in a subsequent reduction of the dataset size. Fig.2 shows a comparison between the initial and the smoothed data for a *Tri-axial acceleration* feature.
3) **Time to Depth Conversion:** in geoscience subsurface applications, we generally care more about the depth of the measured data rather than the time it was taken, this is because subsurface depth gives important insights into the measured data, many physical properties like the pressure, temperature, and density change gradually with depth. So we converted the time series to depth series.

IV. Literature Review

Application of novel deep learning techniques to the drilling activities in the oil and gas industry is relatively new and there is few works to illustrate it. Some works applied (NLP) techniques to analyze drilling reports [1], they presented a methodology for automatic classification of sentences written in drilling reports into three labels (EVENT, SYMPTOM and ACTION). [2] compared different classifiers to predict the wells states, their focus was on the traditional machine learning algorithms (kNN, SVM, LDA, Random Forest, AdaBoostM2, RUSBoost), and they showed that the RF, AdaBoostM2 and RUSBoost achieved the highest performance on real-time detection of operational drilling states. In this work we will be applying (LSTM) neural network architecture, proposed by [3] to address one of the most prevailing weaknesses in RNNs which is the gradient decay, it has been proven very powerful in the handling of sequence type data (speech, time series ...).

V. Model

In this section we present our first model and the results of the training, dev and test phases, we will start by explaining the splitting strategy of the data:

A. Data Splitting

We changed our approach since the milestone phase, instead of training on each well separately we concatenated all the wells processed data into a single dataset, the reason is that our model performed poorly with the previous approach as it was not seeing the entire dataset at each epoch. The processed dataset has 3422464 samples. Combining the data this way helped mitigate the data imbalance issue, with the largest class been *Drilling*, this is because the ratio $\frac{\#Non-Drilling}{\#Drilling}$ is higher in average (0.4181) than it is for the wells we used previously for training. To create the time series for each sample, we used a sliding window. The *time series widow* is characterized by its width, which refers to the number of previous time steps we include in the definition of the current time step series, for example if $w = 4$, the time series for step t^n is $\{t^{n-3}, t^{n-2}, t^{n-1}, t^n\}$. This parameter w is the second input for the LSTM network in the Keras framework $(m, w, 9)$. We also shuffled the data before splitting it into train/dev/test. The data splitting between the train/dev/test sets was respectively $90\%/5\%/5\%$.

For the input features we selected 9 of the dataset columns representing measurements conducted at the top of the well (Flow in, Hold depth, Bit depth, Pressure, Torque, Hook load, Block, Surface RPM, Surface Weight on bit).

B. Model Architecture

The base model is composed of two LSTM layers, two dropout layers and a softmax output layer, we used a softmax layer instead of a sigmoid to be able to use our model for multiclass classification as well (cf. Section VI-D). Sometimes, we added a Time distributed layer just before the output layer. The first LSTM layer is stateful and composed of 48 neurons, the second has the same number of neurons but it is stateless. the input data is 3D array of dimensions $(m = 3422464, w = 3, 9)$. This base model has been modified to create a variety of models by changing the number of neurons, the optimizer, enabling/disabling dropout and tweaking the optimizer parameters. Fig.3 shows the structure of the base model and Table I gives its detailed configuration:

Below we find a description of each of the models we used for our comparison:

- **Base Model:** LSTM - 2 Layers (48x48) , with dropout and Rmsprop optimizer

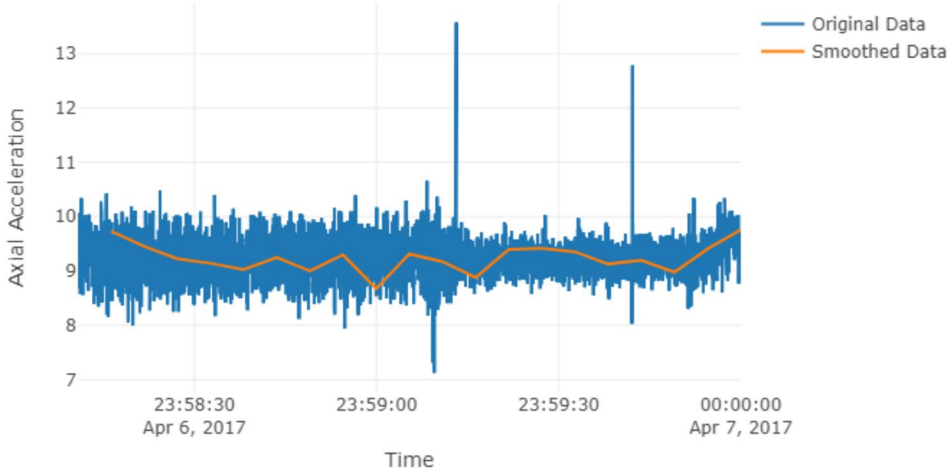

Fig. 2. Comparison between the smoothed and Original data

TABLE I
BASE MODEL

Property	Value
LSTM Network	Base Model
# Nodes layer 1	48
# Nodes layer 2	48
Dropout	0.2
Optimizer	Rmsprop
Model Type	LSTM Stateless
Batch Size	64
Test Split [%]	5
Dev. Split [%]	5
Sliding Window Width	2

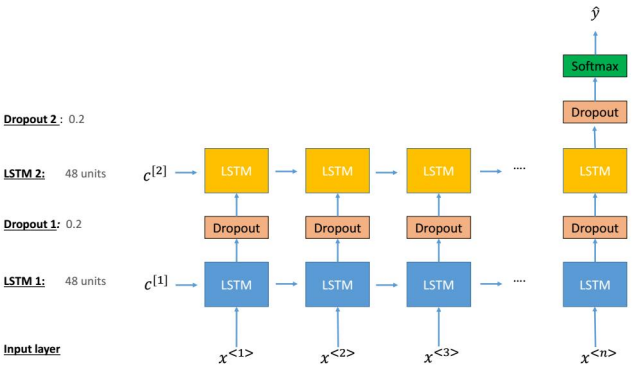

Fig. 3. LSTM Base Model Architecture

- **Model2:** LSTM - 2 Layers (36x12) , with dropout and Rmsprop optimizer (different parameters)
- **Model3:** LSTM - 2 Layers (64x64) , with dropout and Rmsprop optimizer (different parameters)
- **Model4:** LSTM - 2 Layers (48x48) , no-dropout and Adam optimizer.

VI. RESULTS

A. Training

The models were trained for a total of 50 epochs, with a batch size of 64 elements. Both Adam and Rmsprop optimization algorithms were used with the cross entropy loss function. Fig.4 shows the training loss, the accuracy and the F1 score for the train/dev sets. As we can see both the accuracy and the F1 score were high (close to 1) and we will see in the next section that our model generalizes very well to the test set.

If we compare the four models we can see that Model 3 which is the largest has performed better than the others, and Model 4 with the Adam optimizer and without the dropout was the worst performance. We suspect that the absence of the dropout had a negative impact on the models training and we are still investigating that.

Table II shows a detailed picture of the performance of the different models. We see also that in terms of the F1 score, both for the train and dev sets, model 3 outperformed the other models, next we will use this model both on the test set and during the hyperparameters tuning.

B. Test

The test phase was designed in a way to validate our model (model 3) on unseen data from the same distribution as the

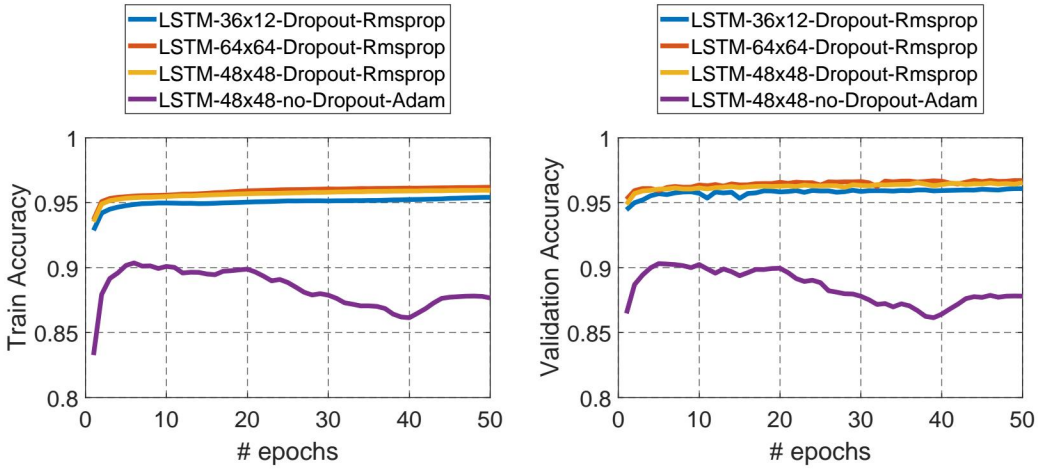

Fig. 4. Training/Dev results

TABLE II
METRICS COMPARISON

	Train			Dev		
Model	**Loss**	**Acc.**	**F1**	**Loss**	**Acc.**	**F1**
Base Model	0.0929	0.9651	0.965	0.0781	0.97	0.9701
Model 2	0.1092	0.9585	0.9586	0.0946	0.9641	0.9641
Model 3	0.0886	0.9666	0.966	0.0748	0.9712	0.9713
Model 4	0.2711	0.8957	0.8959	0.2659	0.8978	0.8965

TABLE III
BEST MODEL-TEST METRICS

	Test		
Model	**Loss**	**Acc.**	**F1**
Model 3	0.0781	0.9701	0.97

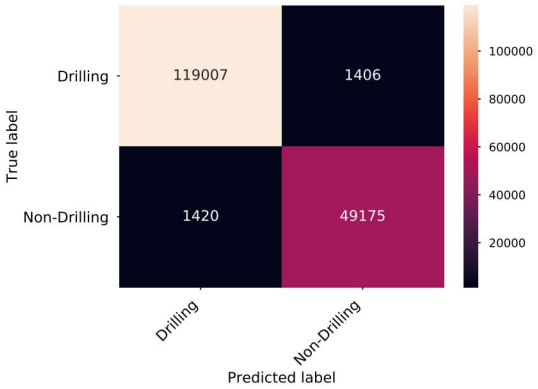

Fig. 5. Confusion Matrix for 2 Class Classification - Test Results

dev/test sets. The ratio *non-Drilling/Drilling time* for the test set is 0.4202, a value that is very close to what we got for the entire dataset which is 0.4181. The accuracy was calculated using Eq.1

$$Accu_{test} = \frac{1}{m}\sum_{i}^{m} 1\{\hat{y} = y\} \tag{1}$$

The results obtained are shown in table III, as we observe our model has a very good accuracy with the test set as well, Fig. 5 shows the confusion matrix for the test case and we see that the model identifies the two classes *Drilling/non-Drilling* very well with less than 2% of the test samples misclassified, despite that our data is relatively imbalanced.

C. Parameters Tuning

A significant part of the tuning has already been presented in the comparison of the models, as each model includes a unique set of parameters (# of neurons per layer, optimizer parameters, dropout inclusion). In this section we will focus on the results we obtained for other parameters, mainly the *time series window width* and the *batch size*. The results are presented in Fig.6 and 7 we get very similar accuracy results that are close to our model 3 performance of 0.9701. We see also that increasing the *batch size* has always improved the accuracy, while the *time series window* gets better results in the case of $w = 6$ compared to $w = 3$ then the accuracy drops when we increase w to $w = 10$, this is probably due to the time length dependence between the successive measurements, a window of $w = 10$ means that the sensor measure at the current time step is impacted by a time frame of 10 seconds before it, this correlation will definitely decrease as we increase the time frame length, we plan to test on longer time frames when we improve our model input data streaming, as we encountered some challenges relaters to memory limitations (increasing w by one value increases by $(m \times 9)$ the size of our input data which imposed some limitation in terms of memory).

D. Other Results

Aside from classifying the samples into the two classes *Drilling/ Non-Drilling*, we also looked at their sub-classes to see if we can use the same model to predict them. Fig.8 shows the subdivision of the *Non-Drilling* parent class into its sub-classes *Pull out/ Descent/ Surface*. The accuracy of the classification was as good as in the case of tow classes,

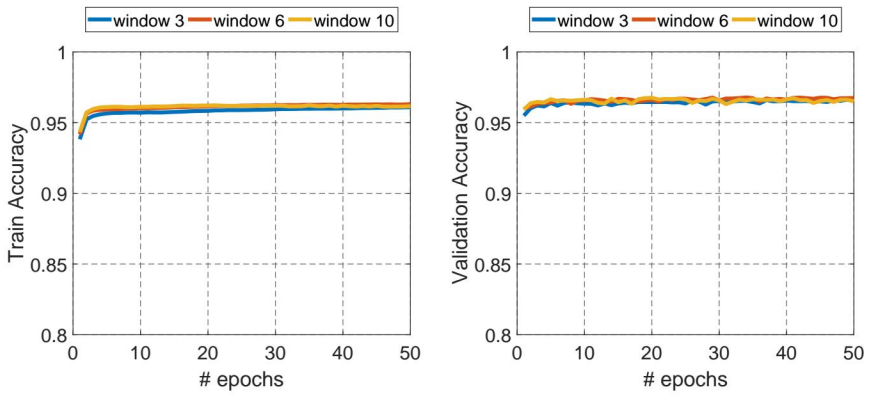

Fig. 6. Hyper parameters Tuning Using different window width

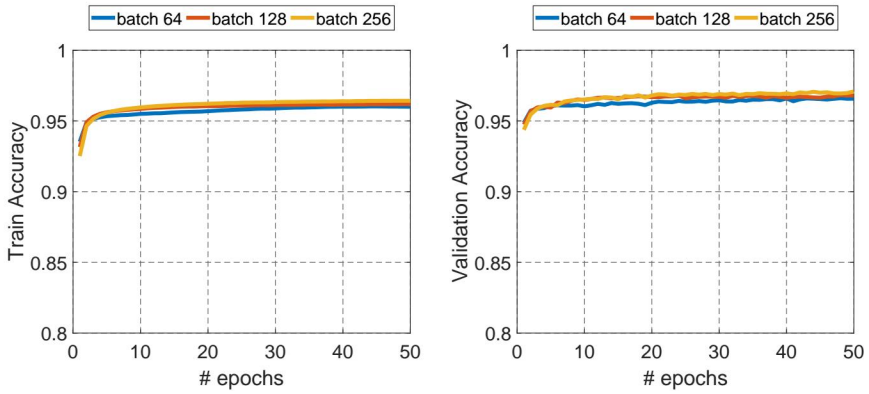

Fig. 7. Hyper parameters Tuning Using different batch size

but when we looked into the the predicted classes it was evident that the imbalance in the input data had led to the mixing of some classes. Fig.9 shows the confusion matrix for the 4 classes *Drilling/Pull out/ Descent/ Surface* and we can see clearly that the two classes *Pull out/ Descent* are not well separated, further exploration should be done to separate them using weighted classes for example.

Fig. 8. Hierarchy of Predicted Properties

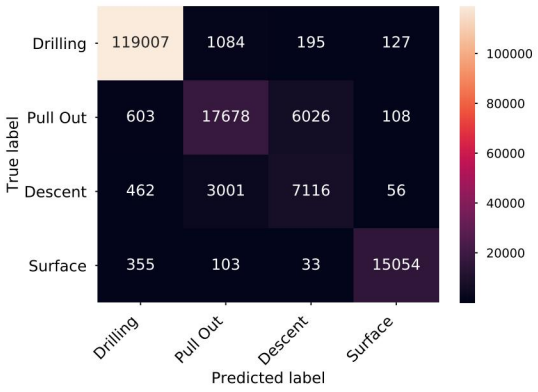

Fig. 9. Confusion Matrix for 4 Class Classification - Test Results

VII. Conclusion and Future Work

LSTM networks were able to predict the overall drilling state based on the interpretation of raw real-time data. This introductory work sets the path for a more refined analysis following a similar approach.

- Human level accuracy for the prediction of Drilling/ non-Drilling states
- Models with large window and batch sizes give the best results despite requiring the longest time to train. There is a trade off in terms of Training time/ Accuracy.
- Accuracy is not the best metric to measure the performance of the Multi-class model (Drilling/ Pull out/ Decent/ Surface) as the data becomes imbalanced. Further exploration should lead to improve the classification accuracy of minority classes (Pull out/ Trip In)
- The results of this first classification can be used as an input to the more challenging 12 sub-classes exercise.

REFERENCES

[1] J. Hoffimann al, Sequence Mining and Pattern Analysis in Drilling, arXiv:1712.01476v1 [cs.CL] 5 Dec 2017 Reports with Deep Natural Language Processing

[2] G.V. Veres al. Data Analytics for Drilling Operational States Classifications, ESANN 2015 proceedings. Bruges (Belgium), 22-24 April 2015

[3] S. Hochreiter, Long Short-Term Memory, Neural Computation 9(8):17351780, 1997.