Applications of Deep Learning Methodology in Real-Time Completion Event Recognition

Yuchang Shen
Occidental Petroleum Corporation
The Woodlands, TX 77380
Yuchang_Shen@oxy.com

Dingzhou Cao
Occidental Petroleum Corporation
The Woodlands, TX 77380
Dingzhou_Cao@oxy.com

Kate Ruddy
Occidental Petroleum Corporation
The Woodlands, TX 77380
Kate_Ruddy@oxy.com

Abstract

Historically, the real-time hydraulic fracturing analytics system (Real-Time Completion system, RTC) rely heavily on manual labelled data. The manual tasks, including fracture stage start/end labeling and ball pumpdown/seat event labeling, suffer from human bias and inconsistent errors, and can easily take up to days to finish. In this paper we developed two machine learning models, the CNN model that detect the stage start and end and the U-Net model that identify the ball pumpdown and seat operation, to fill the manual task gaps and pave the way for the automated stage-wise key performance indicators (KPIs) report generator. These tasks are performed based on the slurry rate and wellhead pressure, which enable the real-time automated stage-wise KPI analysis without human bias, and they also lay the foundation for further advanced analysis regarding real time hydraulic fracture operational decision making.

1 Introduction

Multi-stage hydraulic fracturing is the common operation to maximize the reservoir contact and enhance the fluid flow for shale oil and gas wells. During hydraulic fracturing operations, the pumping data (pressures, rates, volumes et al.) is recorded for real-time monitoring and analytics purposes. Our in-house real-time hydraulic fracturing analytics system, the Real-Time Completion (RTC) system, pulls such hydraulic fracturing pumping data from frac van, analyzes with advanced analytics models, generates meaningful analytics results, and finally pushes to a web-based UI for end users to consume, which aids real-time decision making and drives further well completion efficiency. With today's communication technology, as well as the powerful computation hardware, the RTC system with advanced analytics models plays an increasing important role in today's frac job optimization. For example, Paryani et al. Paryani et al. (2018) propose the real-time frac modeling based on the real-time hydraulic fracturing pumping data so as to optimize the frac job in real-time; Ben et al. Ben et al. (2020a) and Ben et al. (2020b) predict the hydraulic fracturing pressure using machine learning methodology and conduct frac job cost optimization in real-time based on the pressure prediction. In this paper, we present two machine learning models that automate the manual tasks in the RTC pipeline, and pave the way to the real-time hydraulic fracture stage-wise KPI auto-generator (Shen et al. (2020)). The two Machine Learning (ML) models detects the stage start/end and identifies the ball pumpdown/seat respectively. Each of these models are built by various techniques including deep learning methodology such as convolutional neural network (CNN) and the U-Net architecture. The purpose is to automate the current RTC pipeline, dispelling the need for manual labeling of stage start and end and eliminating human bias and errors. It fills the manual task gaps in the RTC workflow and lays the foundation for the further advanced analysis (Ben et al. (2020a) and Ben et al. (2020b)), as well as paves the way for a fully automated RTC system.

2 Current state of art

Ramirez et al. (2019) and Lopez et al. (2019) provides an initial study on the stage start/end detection. They proposed to use the logistic regression and SVM to classify each time stamp as inside or outside a stage then detect the stage start/end accordingly and they achieved an accuracy of 90% from training and validation sets (non-blind test). However, accuracy is not a reliable metric in this case since it is describing the performance of labeling each time stamp rather than labeling the start flag and the end flag. We advanced the technique using deep learning and we designed a more reliable metric called flag-wise accuracy to evaluate the performance of the models in this problem. Our model achieved an F1 score of 0.95 and a flag-wise accuracy of 98.5%. To the authors' knowledge, there are no previous study on identifying the ball pumpdown/seat operation.

3 Methods and key results

3.1 Stage start/end detection

When a human is trying to identify whether a time stamp is inside or outside a stage, he/she not only looks at the data at that specific time, but also look at the previous and subsequent data. We need to look at continuous streams of data to determine the properties of each time stamp. We follow this strategy to structure the problem. First we manually label each time stamp as 0 or 1, indicating whether it is outside or inside a stage, respectively. With these labels we can easily identify the start and the end of a stage by searching for the time when the label changes from 0 to 1 or from 1 to 0. After the labeling, we extract the samples from the data using a fixed-length sliding window (in this case a 50 seconds window). Ramirez et al. (2019) suggested that the slurry rate and the wellhead pressure are among the best features for the stage start/end detection. Other features such as the clean volume and the proppant concentration might be helpful, but since the data quality of these channels are very limited compared to the slurry rate and the wellhead pressure, we chose the latter two as our primary features. In addition to the two features, we add the first and second derivatives of both the slurry rate and the wellhead pressure to give the model additional information. The samples extracted from the data are now matrices with six rows and a fixed number of columns. We label each sample using the time stamp label corresponding to the middle column of the matrix. These sample matrices resemble the data structure of images and will later be fed into a CNN for the image classification task. Since the raw data contains a lot of spikes which will seriously impact the performance, we apply a median filter to get rid of the spikes before sending the data into the model. Figure 1 shows the procedure of extracting the data.

The model we are using is the convolutional neural network and it is implemented using TensorFlow. The network structure consists of an input layer, two convolutional layers, a max pool layer with a dropout rate of 0.3, a dense layer with a dropout rate of 0.3 and an output layer. The convolutional layers will capture the higher level features from the training samples and pass them to the output layer which gives the probability of a sample belonging to either of the two classes according to the softmax function. The first convolutional layer has 8 filters with size 10×6, and the second convolutional layer has 16 filters with the same size. The dense layer has 64 neurons with the ReLU activation. The loss function we are using is the cross entropy function, and the model is trained by the Adam optimizer with a learning rate of 0.001 for 5 epochs.

In order to quantify the performance, we use two different metrics to measure the accuracy. The first one is what we call time-stamp-wise accuracy. The time-stamp-wise accuracy is calculated as the number of time stamps with correct label divided by the number of total time stamps. It measures the accuracy in terms of the duration of a stage. The second metric is the flag-wise accuracy. In order to calculate the flag-wise accuracy, we define a time window called the tolerance window, which has a fixed duration centered at a true flag. Any predicted flags located within the tolerance window are considered accurate. The flag-wise accuracy is calculated as the number of predicted flags within the tolerance window divided by the number of total flags. This metric sheds light on the performance of

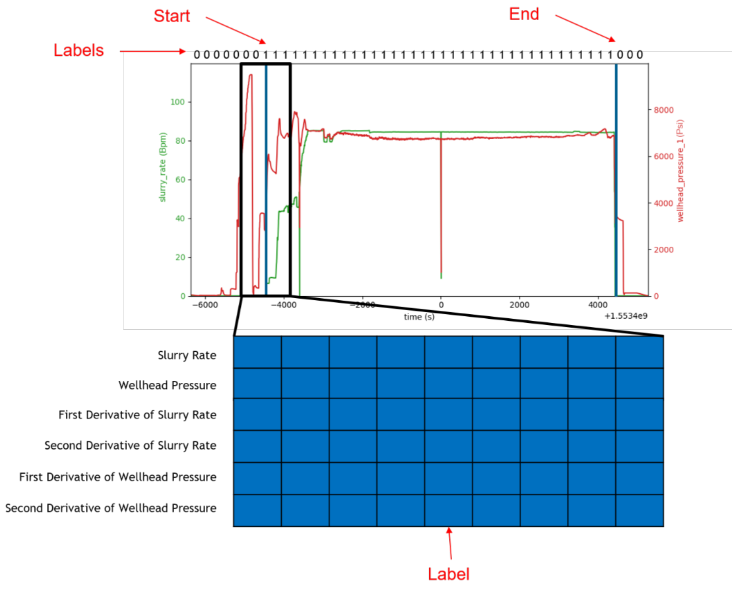

Figure 1: Sample extraction and labeling.

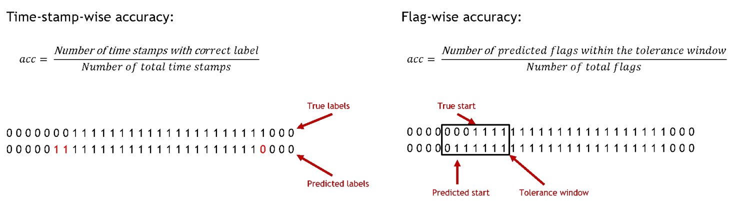

Figure 2: Left: time-stamp-wise accuracy. Right: flag-wise accuracy.

the model in terms of the start and end flag, and we can control the resolution by adjusting the length of the tolerance window. The two metrics are illustrated in Figure 2.

Although flag-wise-accuracy better measures with the start/end identification performance, the training process is designed to correspond to the time-stamp-wise accuracy, in order to keep the dataset from becoming unbalanced. If we use the flag-wise accuracy for the training process, we will have to create labels that directly correspond to the start and end flag, of which there are only hundreds among millions of data points. Conversely, the time-stamp-wise accuracy allows for labeling as a relatively more balanced dataset, since it corresponds to the duration of the stage (60% of data points are within a stage, and 40% are outside a stage). Training on this more balanced dataset yields much better performance.

Figure 3 shows a stage with its true label (blue solid line) and the predicted label (yellow dashed line). The predicted label accurately mark the stage. In order to make full use of the dataset for blind test, we use the five-fold cross-validation technique to evaluate the performance. The dataset is partitioned into five parts. In each trial, the model is trained on four of the five parts then produces predictions on the remaining part. With five trials, we have a full blind test result on the whole dataset, and we use that result to indicate the performance of our model. This performance evaluated by the cross-validation also helps to prevent overfitting. Table 1 shows the flag-wise accuracies with different tolerance window. The model achieves an F1 score of 0.95, and a flag-wise accuracy of 98.5% with a tolerance window of ten seconds.

3.2 Ball pumpdown/seat detection

The ball pumpdown/seat event recognition is a two-step strategy. The first step is to tell if there is a ball pumpdown/seat in a stage, and the second step is to locate the end of the ball pumpdown/seat if

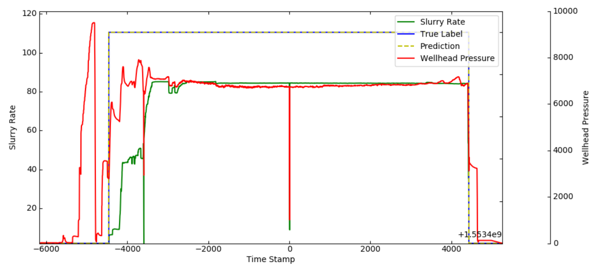

Figure 3: A stage with its true labels and predicted labels. The number at the lower-right corner with the scientific notation indicates the Unix timestamp.

Table 1: Flag-wise accuracies of the blind test results with different tolerance windows.

Tolerance	Flag-wise accuracy	Accurate flags	Total flags
25 seconds	99.7%	646	648
20 seconds	99.3%	644	648
15 seconds	99.0%	642	648
10 seconds	98.5%	639	648
5 seconds	92.3%	598	648
3 seconds	68.1%	441	648

there is one. To achieve the first step, we borrow the idea from Cao et al. (2020) where the authors used the image segmentation method to locate the downlink sequences. Compared to the downlink recognition, the pattern for the ball pumpdown/seat is more obscure and it only counts for a small portion of a whole stage. It is less likely for the image segmentation method to surgically locate the ball pumpdown/seat event, but it is sufficient to tell if a ball pumpdown/seat event exists in a stage. The second step can be achieved by a rule-based selection given the information from the image segmentation. The first step of the ball pumpdown/seat recognition is formulated as an image segmentation problem, and we will need to reconstruct the time series data into images. Different from the stage start/end detection, we are not feeding the values of the time series data into the deep learning model. Instead, we are feeding the time series plot as images into the model. For each plot image, we assign a corresponding mask to indicate the location of the ball pumpdown/seat pattern. Since the ball pumpdown/seat event always happens at the beginning of a stage, we take the data within the first hour of each stage as the samples. Figure 4 shows an example of a ball pumpdown/seat event and the red vertical line indicates where the ball pumpdown/seat ends. Since in the first step we only need to tell if there is a ball pumpdown/seat or not, there is no need to mask and recognize the whole pattern. Noticed that there is a clear ‘Z’ pattern (brown rectangle) in the slurry rate curve, and the wellhead pressure signal is unstable compared to the slurry rate, we use the slurry rate as our only input and we aim at looking for the ‘Z’ pattern.

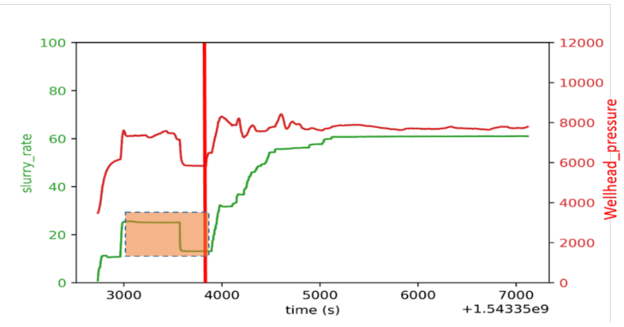

Figure 4: The ball pumpdown/seat at the beginning of a stage. The number at the lower-right corner with the scientific notation indicates the Unix timestamp.

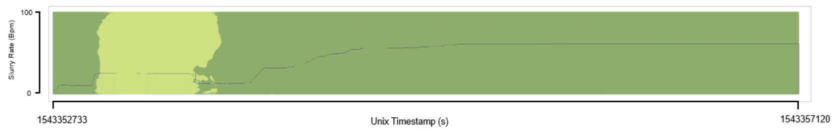

Figure 5: An example of the prediction mask.

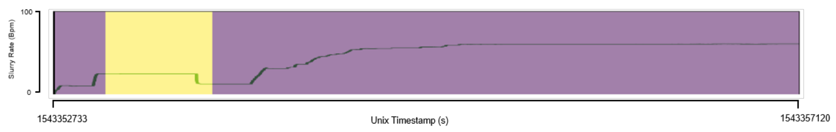

Figure 6: Mask from Figure 5 after cleaning the edge.

We follow the idea of Cao et al. (2020) and use a DL model with the U-Net architecture Ronneberger et al. (2015) to perform the image segmentation task. Since our training dataset consists of 72 samples, which is not enough to train the whole model, we apply the transfer learning technique and we train our model based on the pre-trained U-Net model with skip-connections (ResNet-34). All these 72 samples have a ball pumpdown/seat in them. After training the model, we introduce an extra set of testing dataset. Among the 107 samples in the extra testing dataset, there are 9 samples that have a ball pumpdown/seat phase, and the rest of the samples do not. With that we have two parts in our evaluation. The first part is the blind test of the training dataset, and the second part comes from the extra testing dataset. Result shows that this model alone achieves very good true positive rate (79 out of 81) in determining whether or not there is a ball pumpdown/seat. But it also has a high false positive rate (8 out of 98). Figure 5 shows an example of the prediction mask. In order to lower the false positive rate, we borrow the 2nd opinion mechanism from Cao et al. (2020) and introduce a second model to vote on the decision.

The second model is exactly the same as the first one except that it takes in the wellhead pressure signal along with the slurry rate through an additional channel. The input image is now an RGB image, where the R channel is the wellhead pressure, the G channel is the slurry rate, and the B channel is left blank. With the opinion from the second model, the two models together achieve an F1 score of 0.97, with a true positive rate of 0.96 (78 out of 81) and a false positive rate of 0.03 (3 out of 98), which is a better performance compared to a single model.

After determining the existence of the ball pumpdown/seat, the second step is to locate the end of the event. The masks given by the U-Net model provides a rough information of where the ball pumpdown/seat event happens, and we pinpoint the end of the event based on that information. In order to get the location information from the mask, we first project the prediction mask to 1D in correspondence with the time axis. For each column from a mask, if the fraction of pixels that are marked as positive exceeds a certain threshold, we consider the time stamp that correspond to the column as positive. The threshold for our tests is 95%. Figure 6 shows the mask from Figure 5 after cleaning the edge.

Next we scan the signals from the midpoint of the mask using a set of numerical rules suggested by the engineers. If such a time stamp exists, we mark it as the end of the ball pumpdown/seat. The blue vertical line in Figure 7 shows the predicted ball pumpdown/seat ending point for the sample in Figure 4. Our two-step strategy achieves 94% of accuracy (173 out of 179 samples correct).

4 Conclusion

The work in this expanded abstract automates the manual tasks of labeling the start and end of a stage and the end of the ball pumpdown/seat event with high accuracies. It fills the manual task gaps in the RTC workflow and lays the foundation for the further advanced analysis, as well as paves the way for a fully automated RTC system.

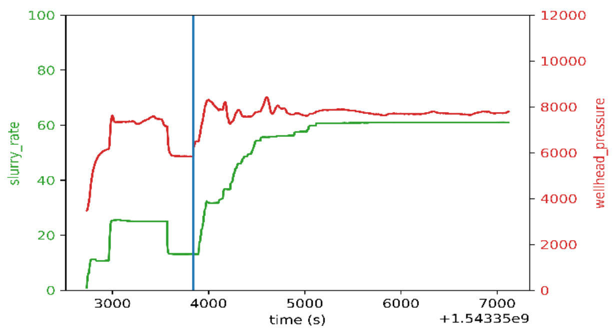

Figure 7: The final ball pumpdown/seat ending point prediction for Figure 4.

References

Ben, Y., Perrotte, P., Mistry, B., Ezzatabadipour, M., Ali, I., Sankaran, S., Harlin, C. and Cao, D. *Real Time Hydraulic Fracturing Pressure Prediction with Machine Learning.* SPE Hydraulic Fracturing Technology Conference and Exhibition, 4-6 February, The Woodlands, TX, USA. SPE-199699-MS. 2020a.

Ben, Y., Sankaran S., Harlin C., Perrotte, P. *Real Time Completion Cost Optimization Using Model Predictive Control.* SPE Hydraulic Fracturing Technology Conference and Exhibition, 4-6 February, The Woodlands, TX, USA. SPE-199688-MS. 2020b.

Cao, D., Hender, D., Ariabod, S., James, C., Ben, Y. and Lee Micheal. *The Development an Application of Real-Time Deep Learning models to Drive Directional Drilling Efficiency.* SPE/IADC International Drilling Conference and Exhibition, 3-5 March, Galveston, TX, USA. SPE-199584-MS. 2020.

Lopez, J. and Ramirez, *A. Machine Learning Helps Pinpoint Events from Fracturing Data. Data Science and Digital Engineering in Upstream Oil and Gas*. Retrieved from `https://www.spe.org/en/dsde/dsde-article-detail-page/?art=5689`. 2019.

Paryani, M., Sia, D., Mistry, B., Fairchild, D. and Ouenes, A. *Real-Time Completion Optimization of Fracture Treatment Using Commonly Available Surface Drilling and Fracing Data*. SPE Canada Unconventional Resources Conference, 13-14 March, Calgary, Alberta, Canada. 2018.

Ramirez, A. and Iriarte, J. *Event Recognition on Time Series Frac Data Using Machine Learning*. SPE Western Regional Meeting, 23-26 April, San Jose, CA, USA. 2019.

Ronneberger O, Fischer P, Brox T. *U-net: Convolutional networks for biomedical image segmentation.* International Conference on Medical image computing and computer-assisted intervention 2015 Oct 5 (pp. 234-241). Springer, Cham. 2015.

Shen, Y., Cao, D., Ruddy, K. and Teixeira De Moraes, L. *Deep Learning Based Hydraulic Fracture Event Recognition Enables Real-Time Automated Stage-Wise Analysis* SPE Hydraulic Fracturing Technology Conference and Exhibition, 4-6 February, The Woodlands, TX, USA. SPE-199738-MS. 2020.