Frac Hits Detection Using Deep Learning and Fiber Optic

1. Introduction

The advance in horizontal drilling and hydraulic fracturing technologies enable the United States to significantly increase its production of oil and gas. Horizontal drilling is the process of drilling a well from the surface to the kick-off point (just above the target oil/gas reservoir), then deviating the wellbore from vertical plane to horizontal plane. As a result, horizontal drilling allows more of the wellbore to stay in contact with the producing formation as compared to vertical well. Hydraulic fracture, informally known as fracking, is the completion technique, in which water, sand and chemicals are injected under high pressure to the formation. The purpose is to create a new fracture in the rock as well as to increase the size, extent, and connectivity of existing fractures. Hence, it increases the well's productivity in the unconventional reservoirs.

To develop unconventional reservoirs efficiently and economically, operators try to drill their horizontal wells as close to each other as possible. With such well spacing pattern, hydraulic fracture hits among the horizontal wellbores have become norm. Hydraulic fracture hits (frac hits) are induced by fractures from a well that propagate to adjacent well. The impacts of frac his were widely reported in the industry with mixed results. In several cases, frac hits can reduce the productivity of the adjacent wells. However, frac hits can give feedback about completion parameters such as fracture length, width, and height if the project is carefully designed to do so. Optimizing completion parameter is critical to improve hydraulic-fracture efficiency and unconventional production performance.

The objective of this project is to develop a deep learning model to detect frac hits given the distributed acoustic sensor (DAS) as the input. The wireline-fiber was deployed in the monitor well while adjacent wells was fracked. DAS is an emerging fiber-optic-based technology that can measure the strain change along the fiber. DAS signal in low frequency band (also known as slow strain) can be used to monitor the strain perturbation due to the frac hit of treatment wells.

2. Data Acquisition and Cleaning

The fiber-optic data used in this project is confidential and internal within my company. The whole dataset is about 14GB, which consist of 68,333 LAS files. Each LAS file is the slow strain measurement along the wellbore from the surface to the end of the wellbore at every second. All slow strain data was loaded into a Pandas dataframe. Fiber optic cable was setup in the configuration that it is connected to the IDAS interrogator unit within the acquisition trailer and runs from there to the wireline unit nearby and to the wellhead and down to subsurface. Because the zero-depth reference was conventionally setup at the wellhead, the depth of fiber from the wellhead back to acquisition trailer is showed as negative values. Usually, we are only interested in the wellhead to subsurface section; therefore, I filtered the negative depth out of the dataframe. Next, the negative and positive slow strain measurements at each depth are separated into negative and positive slow strain dataframe.

There were 79 hydraulic stages for this project. The start time and stop time of each stage is defined by when the pumps were turned on and off respectively. We usually started recording 5 minutes before the stage starts and continued recording for another hour after the pumps were shut off. However, because the pumping company was able to start the subsequent stage within 10-15 minutes after the previous stages ended, the data was the previous stage was recorded until 5 minutes before the next stage started. The "Pre_frac', 'During frac' and 'Post frac' cumulative slow strain were computed. Furthermore, more engineer features were created at this stage for machine learning model later, such as 'RMS', 'FFT' for both negative and positive slow strain prefrac, during frac and post frac. In addition, 'Total_Strain_Duringfrac' and 'Total_Strain_Afterfrac' were computed as well as the 'Delta_SSPS' and 'Delta_SSNS' during and post frac. Finally, I combined all features for each stage into a dataframe, which were later export out as csv files.

Next step, I need to provide the label for my data to indicate at which depth the frac hits occurs. Normally, field engineers within my company will start label the frac hits as soon as the treatment stage ends. Nevertheless, after I QC the provided labels, I discovered that the process to label the frac hits were subjective. For the same stage treatment, the labels will be different depending on who labels it. This adds more noise to the data. After I spent some time for literature review plus my observations, I decided to use different algorithm to label the frac hits. The algorithm allows me to pick the same frac hit depth regardless who performs it. There were 79 frac hits csv files that corresponding to 79 slow strain csv files. After loading each slow strain and frac hit csv files into separated dataframe, they are merged by depth column.

3. Data Exploration

First, I loaded all the csv files into the dataframe 'data'. Then, I want to take a quick glance into the data.

data.head()

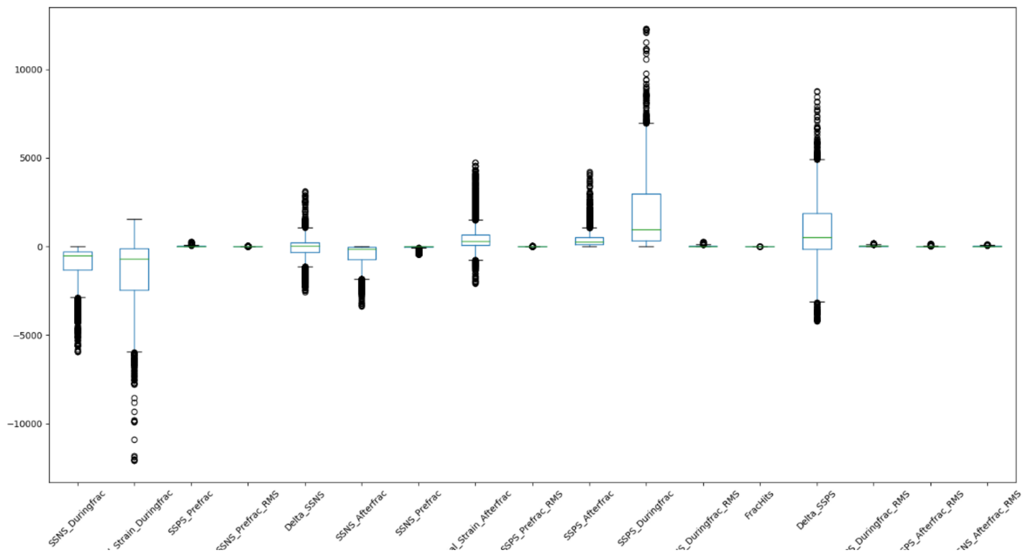
```
           SSNS_Duringfrac  Total_Strain_Duringfrac  SSPS_Prefrac  \
Depth
20258.791 -393.92           80.11                    2.64
20262.146 -392.62           75.02                    2.37
20265.502 -392.14           74.63                    2.24
20268.857 -385.57           73.90                    1.89
20272.215 -386.79           72.50                    2.14

           SSNS_Prefrac_RMS  Delta_SSNS  SSNS_Afterfrac  SSNS_Prefrac  \
Depth
20258.791  0.070640          85.85      -33.04          -1.11
20262.146  0.081158          80.17      -35.21          -1.26
20265.502  0.090646          77.96      -36.47          -1.39
20268.857  0.097074          81.75      -32.66          -1.49
20272.215  0.085538          75.79      -37.55          -1.27

           Total_Strain_Afterfrac  SSPS_Prefrac_RMS  SSPS_Afterfrac  \
Depth
20258.791  360.88                  0.127410          118.89
20262.146  357.41                  0.115686          115.38
20265.502  355.67                  0.111086          114.43
20268.857  352.91                  0.100615          114.41
20272.215  349.24                  0.108873          113.34

           SSPS_Duringfrac  SSNS_Duringfrac_RMS  FracHits  Delta_SSPS  \
Depth
20258.791  38.78            4.104243             0        -355.14
20262.146  40.36            4.108679             0        -352.26
20265.502  39.80            4.086344             0        -352.34
20268.857  40.51            4.048087             0        -345.06
20272.215  40.84            4.054293             0        -345.95

           SSPS_Duringfrac_RMS  SSPS_Afterfrac_RMS  SSNS_Afterfrac_RMS
Depth
20258.791  0.724665             1.776064            0.705337
20262.146  0.758402             1.726569            0.729098
20265.502  0.777089             1.720084            0.746282
20268.857  0.788190             1.713045            0.698112
20272.215  0.798294             1.714032            0.759866
```

Let take a closer look into each column of the dataframe. As we can see from the figure below, there are 6122 observations. Most of the columns are float data, except for column 'FracHits' which is the labels for this project and has integer values. Null values are not observed from this dataframe.

```
In [63]: data.info()
<class 'pandas.core.frame.DataFrame'>
Float64Index: 6122 entries, 20258.791 to 9078.459
Data columns (total 17 columns):
SSNS_Duringfrac            6122 non-null float64
Total_Strain_Duringfrac    6122 non-null float64
SSPS_Prefrac               6122 non-null float64
SSNS_Prefrac_RMS           6122 non-null float64
Delta_SSNS                 6122 non-null float64
SSNS_Afterfrac             6122 non-null float64
SSNS_Prefrac               6122 non-null float64
Total_Strain_Afterfrac     6122 non-null float64
SSPS_Prefrac_RMS           6122 non-null float64
SSPS_Afterfrac             6122 non-null float64
SSPS_Duringfrac            6122 non-null float64
SSNS_Duringfrac_RMS        6122 non-null float64
FracHits                   6122 non-null int32
Delta_SSPS                 6122 non-null float64
SSPS_Duringfrac_RMS        6122 non-null float64
SSPS_Afterfrac_RMS         6122 non-null float64
SSNS_Afterfrac_RMS         6122 non-null float64
dtypes: float64(16), int32(1)
memory usage: 997.0 KB
```

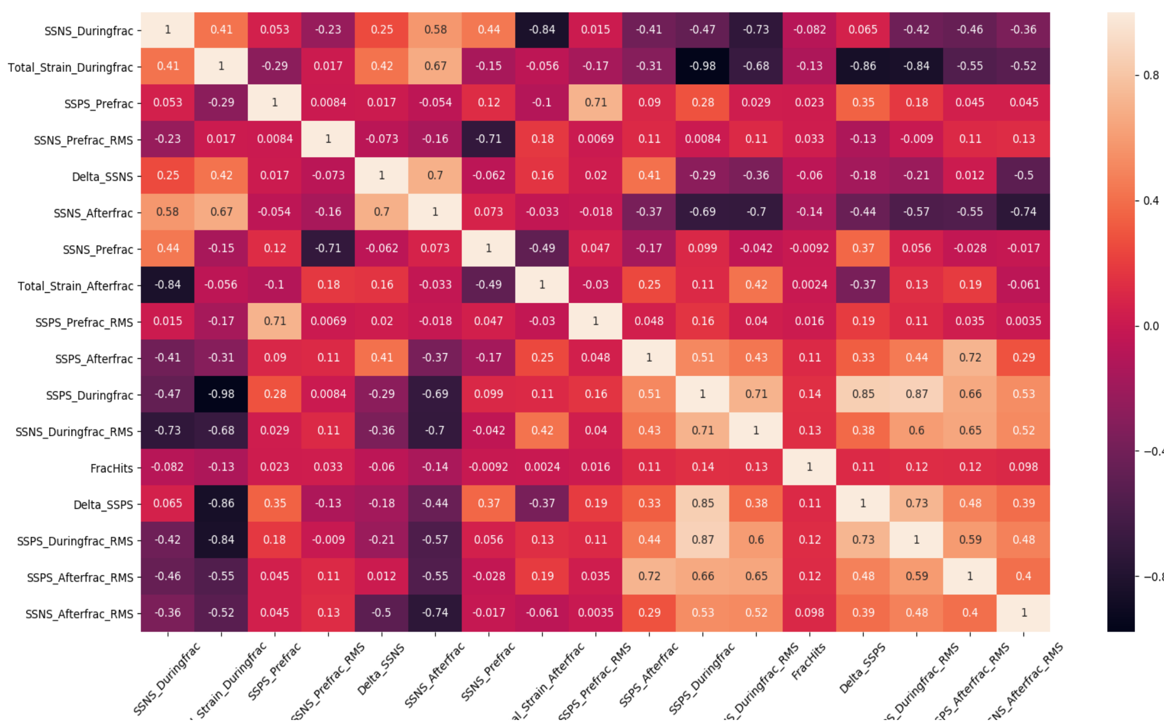
Next, I want to look at descriptive statistic of the data. It's easier to look at the box-plot.

```
       SSNS_Duringfrac  Total_Strain_Duringfrac  SSPS_Prefrac  \
count  6122.000000      6122.000000              6122.000000
mean   -972.211643      -1451.836241             24.091351
std    1050.515400      1812.950636              41.835338
min    -5935.040000     -12088.850000            0.000000
25%    -1317.292500     -2454.120000             0.000000
50%    -517.785000      -702.260000              7.440000
75%    -277.437500      -118.545000              30.055000
max    -0.050000        1552.900000              250.470000

       SSNS_Prefrac_RMS   Delta_SSNS  SSNS_Afterfrac  SSNS_Prefrac  \
count  6122.000000        6122.000000  6122.000000    6122.000000
mean   2.199767           -45.164913   -452.004670    -36.360420
std    5.838411           588.703189   577.942085     76.845282
min    0.000000           -2565.210000 -3369.550000   -433.180000
25%    0.000000           -310.970000  -741.940000    -34.430000
50%    0.160984           51.685000    -159.230000    -3.220000
75%    1.288943           240.640000   -25.215000     0.000000
max    47.693855          3128.140000  0.000000       0.000000

       Total_Strain_Afterfrac  SSPS_Prefrac_RMS  SSPS_Afterfrac  \
count  6122.000000             6122.000000       6122.000000
mean   520.206973              1.066433          406.839757
std    858.577009              2.605551          454.146102
min    -2092.130000            0.000000          0.000000
25%    85.630000               0.000000          129.300000
50%    305.095000              0.380103          263.920000
75%    653.967500              1.122730          501.505000
max    4745.900000             36.798000         4228.370000

       SSPS_Duringfrac  SSNS_Duringfrac_RMS     FracHits   Delta_SSPS  \
count  6122.000000      6122.000000          6122.000000  6122.000000
mean   1858.675998      39.731490            0.009311     886.464355
std    2000.283960      49.524605            0.096049     1771.451903
min    0.000000         0.009129             0.000000     -4205.000000
25%    338.062500       5.679909             0.000000     -134.307500
50%    967.160000       16.199486            0.000000     506.855000
75%    2992.250000      55.832395            0.000000     1883.562500
max    12301.040000     254.994588           1.000000     8778.930000

       SSPS_Duringfrac_RMS  SSPS_Afterfrac_RMS  SSNS_Afterfrac_RMS
count  6122.000000          6122.000000         6122.000000
mean   33.350053            17.411284           15.572593
std    28.624948            23.920166           19.570420
min    0.000000             0.000000            0.000000
25%    7.615124             2.764001            1.223586
50%    27.326357            6.673329            6.033174
75%    52.878354            21.525894           25.292452
max    188.028913           169.310345          128.567894
```



As we can see, there are lots of outliers in each features of my data. It is necessary to investigate on these outliers. However, for this study, I will leave the outliers as they are in my data, as I suspect it might reflect the natural of our measurement data.

The next step is to investigate the correlation between each feature in the data.

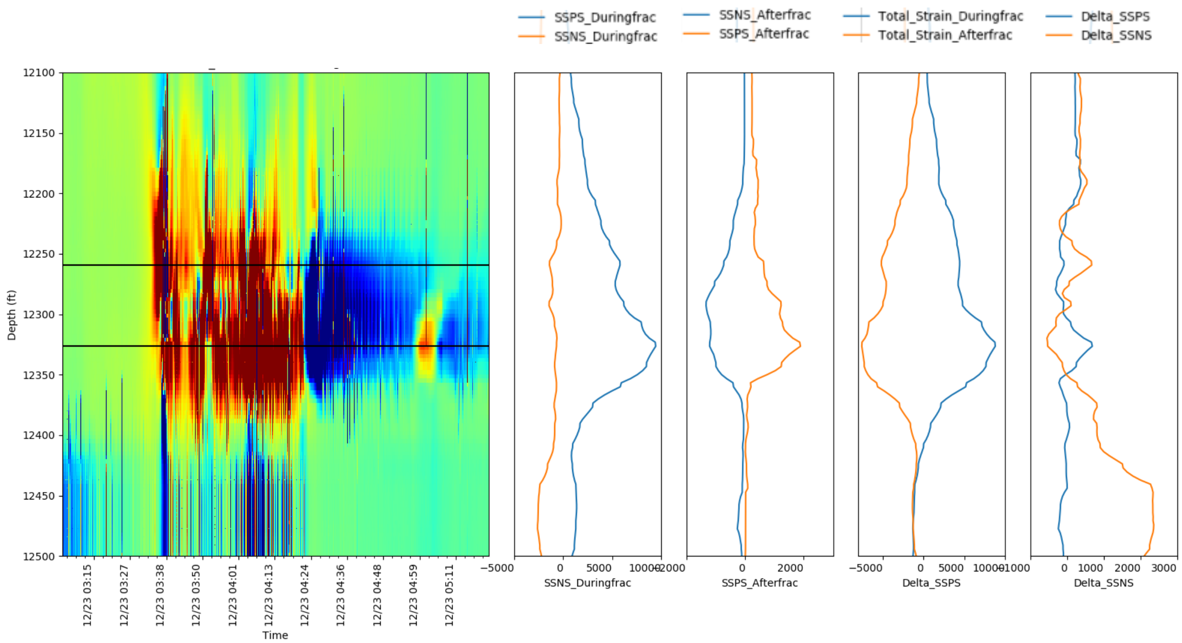


Several important correlations are observed from the heatmap above. These are critical because this is what I was searching for in order to train my model to pick the frac hits.

- 'SSNS_Afterfrac' exhibits negative linear relationship with 'SSPS_Duringfrac'.

- ‘Total_Strain_Duringfrac’ show strong linear relationship with ‘SSNS_Afterfrac’ and inverse relationship with ‘SSPS_Duringfrac’ and ‘Delta_SSPS’

After observing the correlation between features, I’m wondering if these correlation will help to pick the frac hits. Therefore, I plot the slow strain measurement together with the features, to see if I can pick out any correlation.



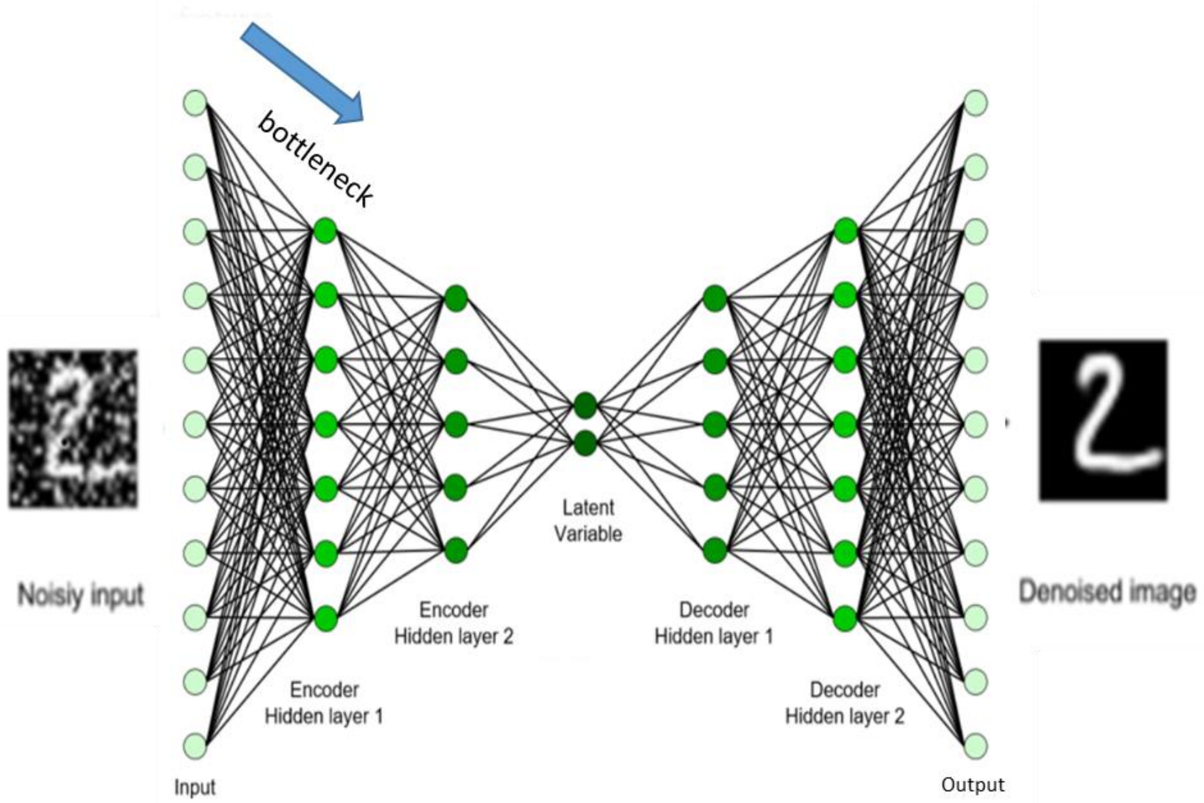
The first plot on the left shows the slow strain measurement along the time and depth. The two solid black lines indicates the depth where the frac hits occurred. The remaining plots on the right show the features used to model the frac hit pick. There appears some peak correlation between the curves where the frac hits happen.

4. Anomaly Detection with Sparse Autoencoder to detect Frac Hits

Anomaly detection is the task of determining when a behavior or events have deviated from the norm. It has many application on credit card frauds, network intrusions and equipment failure. In this study, a deep learning model spare autoencoder is used to detect frac hits in the monitor well while the treatment well is fracked.

4.1 <u>Autoencoder Background:</u>

Autoencoder is a subset of Artificial Neural Network (ANN). Unlike ANN, Autoencoder aims to reproduce the input to its output. Encoder is the part of network that compresses the input into a latent-space representation. In contrast, decoder is the part of network that reconstruct the input from the latent-space representation.

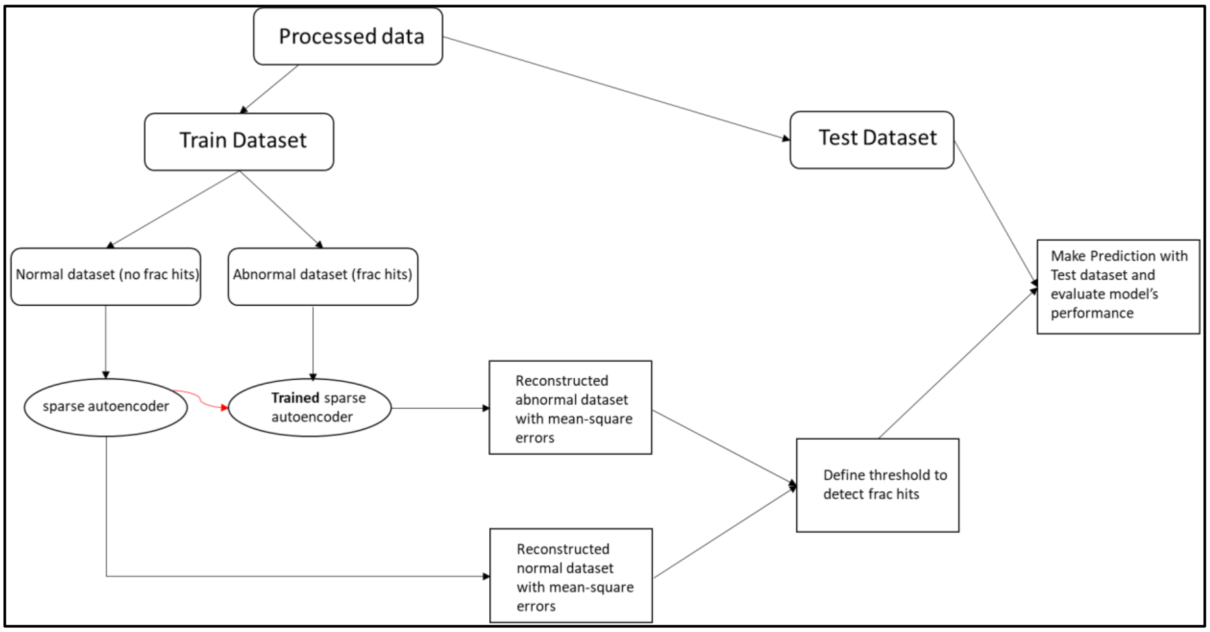


Source: Medium

Autoencoder is designed in fashion that it will reproduce output that is identical to the input. Instead, the learning of the network is constrained in a way that the latent-space representation only contains the most useful properties of the input. As the **fig.xx** shown, the noisy digit '2' is push through the autoencoder network and the denoised digit '2' is shown as the output. As the autoencoder is constrained, the network only learns the salient feature of the input; therefore, the noise in the noisy input is ignored. As a result, a denoised reconstructed input is shown as output.

Undercomplete and overcomplete autoencoders are two different ways to constrain the autoencoder. In the undercomplete case, the encoder hidden layers are designed to have smaller dimension than input x. In contrast, the overcomplete autoencoder, also known as sparse autoencoder, allows encoder hidden layers to have higher dimension than the input x. So, it put constraints on the network by inducing regularization on autoencoder. In either fashion, a bottleneck is introduced in the network that force the autoencoder to learn the salient feature of the input. In the testing phase of this project, the sparse autoencoder exhibits better performance than autoencoder because it has more room to design for the number of nodes in hidden layers than autoencoder.

4.2 Methodology

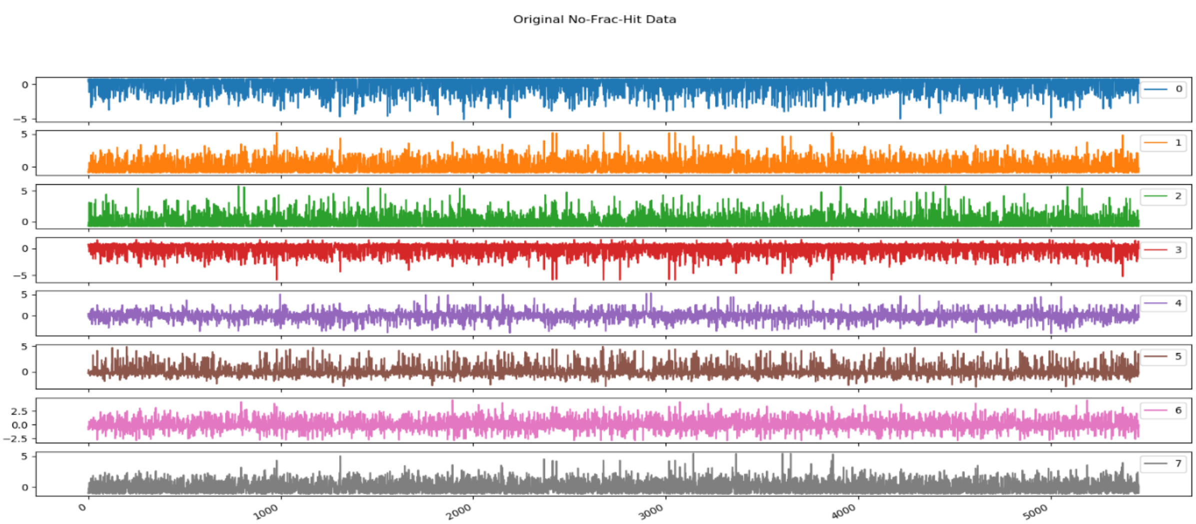


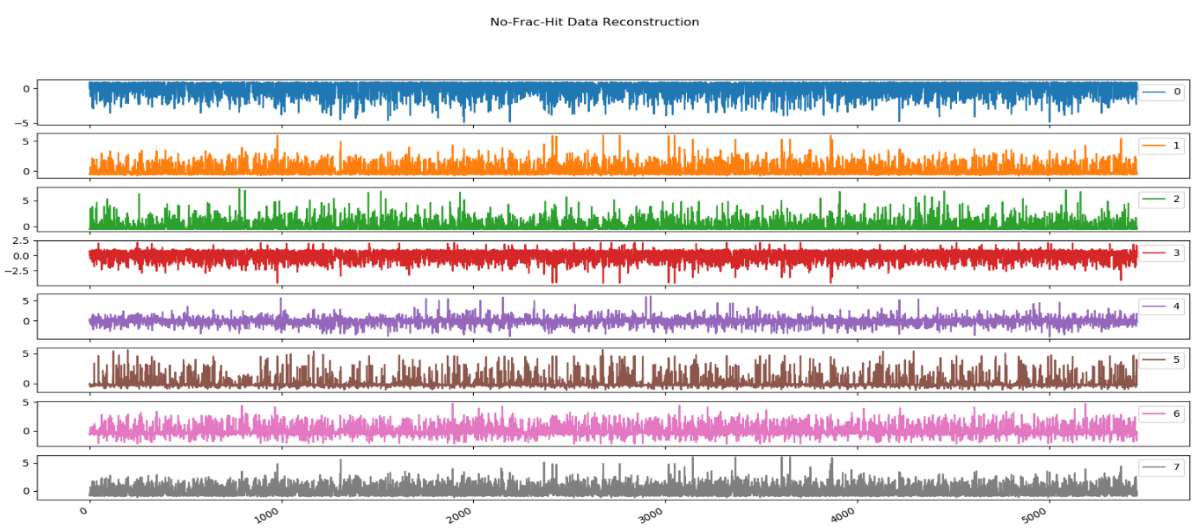
After data wrangling process, dataset is split into train and test datasets in 90%:10% fashion. The train dataset is used to train and fine tune model. Whereas the test dataset is hold out to the side and only be used to evaluate the model that has been trained with training data. The train dataset is further divided into normal behavior and abnormal behavior dataset. In this project, the normal behavior is defined as no frac hit was observed from the monitor well while the treatment well was treated. Abnormal behavior is when frac hits are observed. Next, the non-frac hits data is used to train the spare autoencoder. Once the model is trained, the mean-squared-error (MSE) is computed to quantify how much difference between the input data and reconstructed input for non-frac hit data. Similarly, the frac hit data is push through the trained model and the MSE is computed. The idea is if the spare encoder is trained with non-frac hit data, it will learn the salient features of the input data. Therefore, the MSE between the original and reconstructed input data for non-frac hit data should be relatively small. The trained model, which capable of reconstruct the input data well, will do terrible job at reconstruct the frac hit data as original input. Hence, the MSE between original input and reconstructed input for frac hit data is expected to be higher than the MSE for non-frac hits data. If the MSE for both dataset is plotted, two group of data should be distinct enough. A threshold is set, so if the MSE is above this threshold will be consider abnormal (frac hit).

4.3 Sparse Autoencoder

Keras is used in this project to train the deep learning model. First, all the necessary libraries are imported to the Python. Two utilities functions are used to format the data and compute metrics for the model. The preprocessed csv files from the wrangling process are loaded into a Pandas dataframe. The unwanted features will be dropped out from this dataframe. Next the train data is separated to the non-frac hit and frac hit datasets. Both datasets are normalized with 'StandardScaler' so the data is transformed in a way that the distribution will have a mean value 0 and standard deviation of 1.

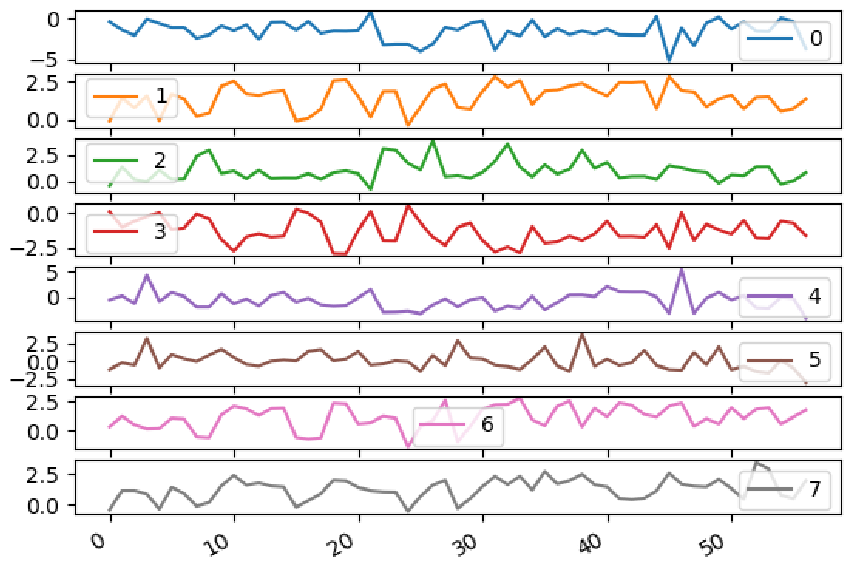
The next part is to design the spare autoencoder network. There are 8 neuron nodes for input layer. So, I used 12 nodes for the encoder hidden layers with Lasso regularization (L1) with the strength of 0.2. Leaky Rectified Linear Unit (Leaky_ReLU) is chosen as activation function because its range goes from negative infinitive to positive infinity which is suited for the features' values ranges. Decoder layer has 8 nodes just like the input layer. Adaptive Moment Estimation (Adam) is used as optimizer because it the most popular and widely used optimizer in deep learning. I chose accuracy as the metric and mean-square-error to compute the loss.



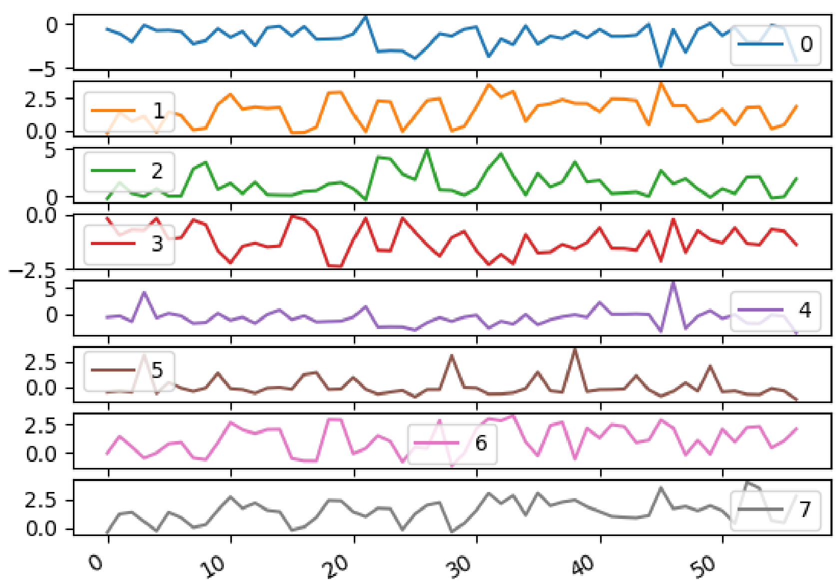


Above are the original input data vs. reconstructed input after the model is train. As illustrated, the reconstructed input looks very similar to the original input.

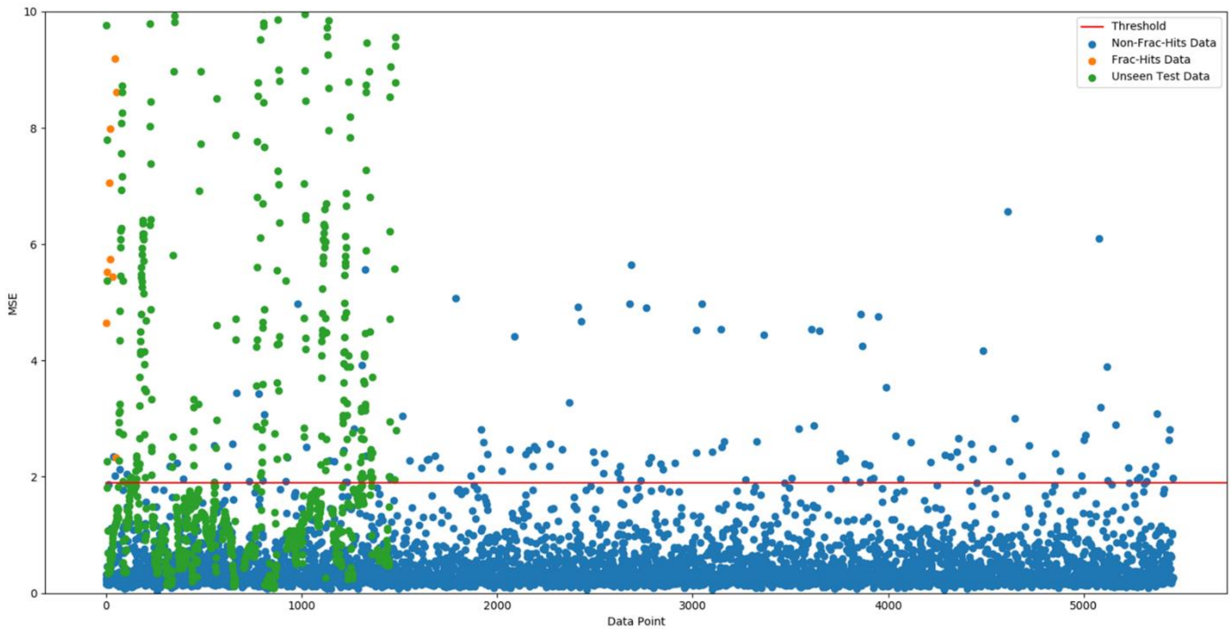
Original Frac-Hit Data



Frac-Hit Data Reconstruction



In contrast, we can see the difference in magnitude between original frac hits data to the reconstructed frac hits data. It is difficult to see how much difference between them, so a scatter plot for the mean-squared-error for non-frac hits and frac hits data will reveal the trend better.



The blue and orange dots represent the non-frac hit and frac hit data respectively. As illustrated, the mean-square-error results for the frac hits data are generally higher than the non-frac hit data. Several non-frac hit data was so big that it is out of the scale in this picture. Another interesting observation is that there are several non-frac hit data that can reach the same level of mse results. Those points should be investigated further but out of scope for this project.

The green dots represent the unseen test data. It is a mixture of frac-hit and non-frac hit data. A threshold is needed to set in order to detect frac hit. It's decided to set at 2 as indicated in the plot. A confused matrix is used to summarize the prediction for the unseen test data.

		Predicted	
		No Hit	Frac Hit
TRUE	No Hit	958	523
	Frac Hit	2	11

From the matrix, the model can correctly detect 11 frac hits out of 13, which yield 85% accuracy. This is very good result. However, it also misclassifies 523 non-frac hits as frac hit, which yield 65% accuracy. There is the tradeoff between the true positive and false negative. The threshold can be adjusted which will vary the result for true positive and false negative.

5. Conclusion

- A sparse autoencoder was developed to detect the frac hits with 85% accuracy for true positive and 65% accuracy for false negative.
- Need to investigate why the mse results for some of the non-frac hit data is as high as frac hit data.

6. Future Work

- The model is capable to detect the depth where the frac hit happens. However, it does not have the capacity to specify the time when it happened.
- LSTM autoencoder should be considered next to detect when the frac hit happens.