Project2 - RNN - Predicting Laboratory Earthquakes

Emanuel Buchholz

April 16, 2019

Instructors: Javier Bejar

Important Links
Github
Kaggel Challenge

1 Introduction

Predicting the magnitude and timing of rare events is important in many fields and one of the seemingly most unpredictable, but very important rare event are earthquakes. Being able to predict them is an active field of research because even short warning times can significantly improve survival rates in effected populations. Until now it is impossible to predict earthquakes, but researchers managed to predict laboratory earthquakes as seen in Rouet-Leduc et al. (2017). Predicting rare events is a broad field of research, including the prediction of failures of machine parts (predictive maintenance), other natural disasters like landslides and avalanches as well as disasters in complex human systems like an economic crash e.g bursting of a bubble or a Seneca collapse. In general this kind of forecasting applies to every event that takes a stable complex system through a catastrophic failure state to another stable state. Predictive maintenance for example already gets applied to many areas like for example hard drive failure prediction in Anantharaman et al. (2018). Although according to this source (and others) LSTM (Long-short-term memory) is not necessarily the best method and many have identified Random Forrest models as the better alternative. But since this lab report only focuses on Recurrent Neural Networks I will not investigate different machine learning algorithms on the data-set.

1.1 The Data-Set

The data-set used for this lab report is provided by the 'Los Alamos National Laboratory' in the United States. It is published as a challenge on Kaggle and concerns only the prediction of **when** an earthquake occurs. The data-set was created by simulating earthquakes in a laboratory as described by Leeman

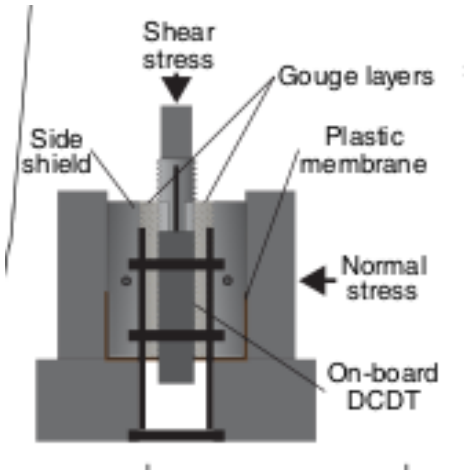

Figure 1: Experimental setup for simulating the laboratory earthquakes

(2016). The researchers used quartz gouge as a material to investigate the frictional failures of slow fault slip earthquakes in an experimental setup as you can see in Figure 1. Slow fault slip earthquakes are a failure behavior in which slip accelerates but does not reach rates sufficient to radiate high-frequency seismic energy like a regular earthquake would. Therefore they are less catastrophic. These slow fault slip phenomena are not yet very well studied in a laboratory environment an most studies have been conducted with real seismic data. Al thought they are not as catastrophic as regular earthquakes, they can still reach magnitudes of M7+ and play a role in stress transfer and triggering of catastrophic earthquakes. Also they have been observed as precursors to regular earthquakes.
The data set contains a training file with continuous measurements of approximately the size of 2Gb with about 700 million samples and 17 failure events. The predictor is the acoustic data measured from the experiment setup, the target value is a linearly decreasing time value indicating the time until the next failure will occur.
You can see a sample plot of this in Figure 2.

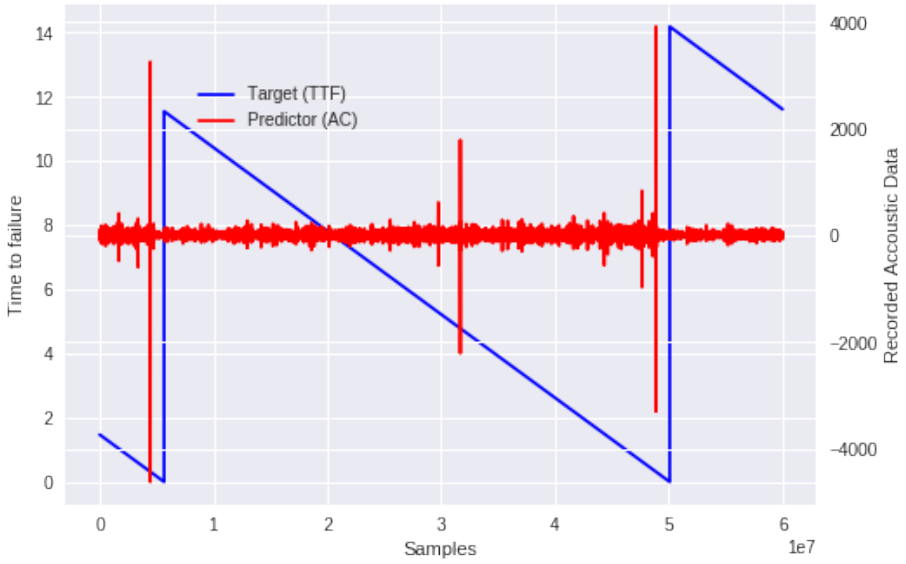

Figure 2: Sample plot of predictor (accoustic data) and target (time to failure (TTF))

2 State of the Art/Methodology

2.1 Pre-Processing

2.1.1 Data Reduction

Since the data-set is quite big and basically impossible to load as one into memory, the prepossessing will need to address this. I will follow two approaches here. In my first approach I will reduce the size of the data by down-sampling it. Since I will implement a many-to-one approach for the GRU and LSTM networks, also the TTF data size can be reduced massively. In this approach the code implementation loads the data-set in chunks into memory and generates training data from every chunk by first down-sampling the data, removing the earthquake events (since they represent outlier events) and then randomly selecting a specific amount of training data from this down-sampled chunk in memory. Here I have to note that this did not work on the server which is why I did it offline and then loaded the generated data set. Of course this might not produce a data-set that represents the original data-set very well, which is why I will also try to implement this with a data generator and the fit_generator function of Keras.

2.1.2 Data Scaling

I will try three approaches for scaling the data. In the first, no scaling is applied, in the second, a MinMax scaler is applied, and for the third a standard scaler is applied.

2.1.3 The Network

For the network I choose a network architecture with two hidden layers and one linear dense output layer. Both hidden layers contain 100 neurons and the data input is 1500 samples.
I choose to implement the network as LSTM and as GRU (Gated Recurrent Unit) in order to compare which has the better performance.
In the end I will choose whichever scaler or network architecture seems to have the best performance and try to improve the performance of this network specifically with additional design like adding Dropout etc.

2.1.4 Adding Convolutional Layers

As the last experiment I will use the previously optmized network and add convolutional layers for data extraction.
A major problem of downsampling by taking the average is that important structures might disappear. I hope to solve this problem by 'downsampling' with a convolutional network which will essentially learn the how to downsample without loosing important patterns.
For this network I will use the original data by creating a data generator which will read in the *.csv file in chunks. From these chunks I will sample a random training sample which I will feed to the the fit_generator function in Keras.
The convolutional network will contain two convolutional and two max pooling layers which will bring the dimension from 1500x1 down to the 100x1 the GRU network was optimized for.

3 Experimental Results

You can see the results of the experimentation with different scalers and using GRU or LSTM as hidden layers in Figure 3. As you can see the MinMaxScaler does not work, as no learning is happening, although there might be a little bit of learning going on in the LSTM version of the experiment. From this we can see that the MinMaxScaler is not suitable for this kind of tasks, which is the case because the dataset contains many outliers. Or more specifically there are still some spikes left while the majority of the data is around zero.
Not using a scaler is possible for this data, although the convergence is a bit slower than for the network version with the standard scaler. The standard scaler works very well here, because it preserves the standard deviation and does not suppress outliers

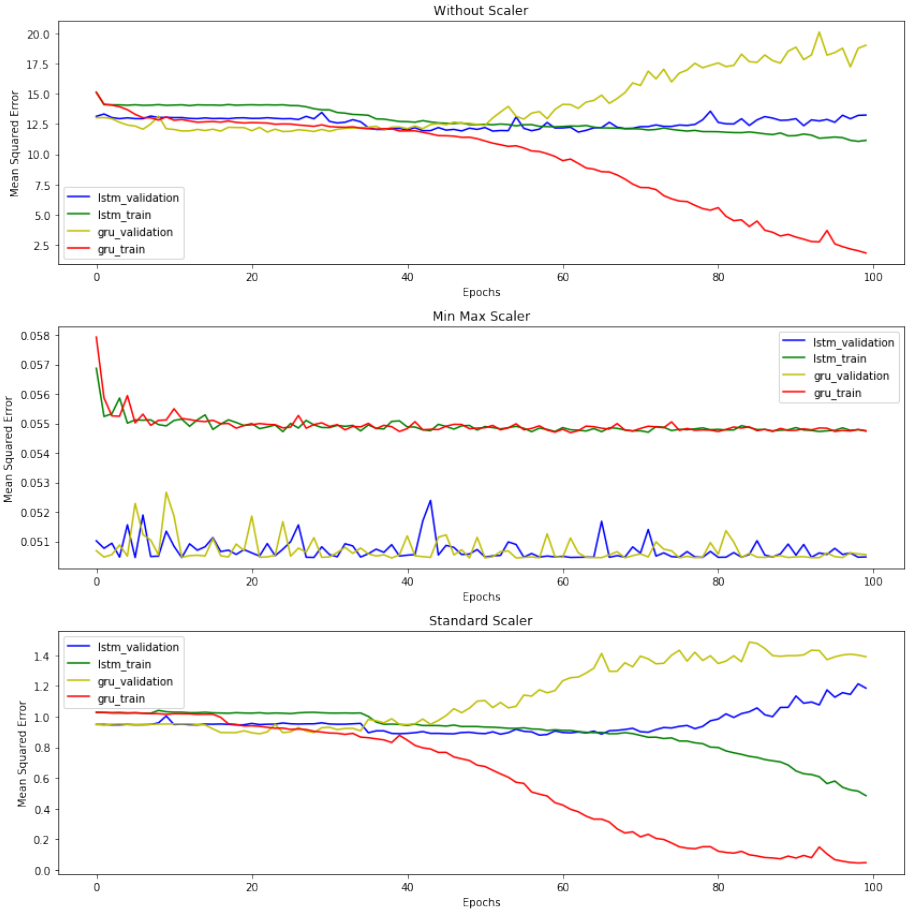

Figure 3: Which Scaler is the best

3.1 Improving the Network

Since I did not notice a better performance of the LSTM networks over the GRU networks in Figure 3, I decided to use Occams razor and keep improving the GRU network since it is simpler and less computationally intensive.

3.2 Using Convolutional Layers

As you can see in Figure 5, the convolutional network still needs design improvements. One of the mayor problems is the high variance of the validation set data. This is most likely due to the random sampling. From this we can see that the generator definetly needs improvement. I choose the random sampling, because the server cluster always ran into job memory limits and I had to run the network on a Google Colab instance. Therefore I could not run all the data

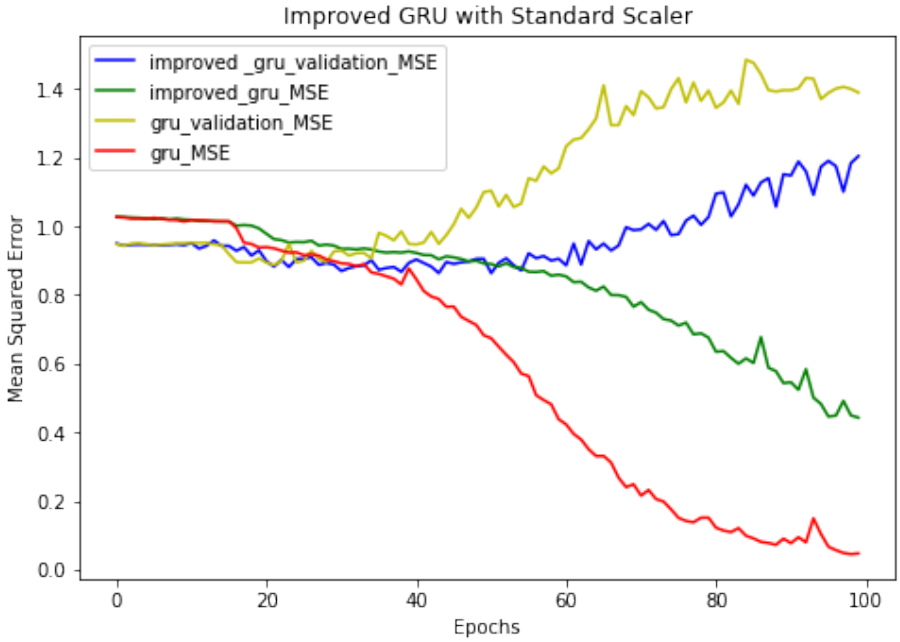

Figure 4: Improvement of the GRU Network with Standard Scaler

through the network every epoch due to computing time restrains. Additionally I think improvements can be made on the architecture of the network, for example changing the dropout values, changing the input size of the network and scaling the batches with a standard scaler for faster convergence. Unfortunately I am not able to implement these due to computing and time restraints. Additionally I think the improvements might not be that significant if we compare it with the MSE of the first plot in Figure 3. Both ac curacies are quite similar.

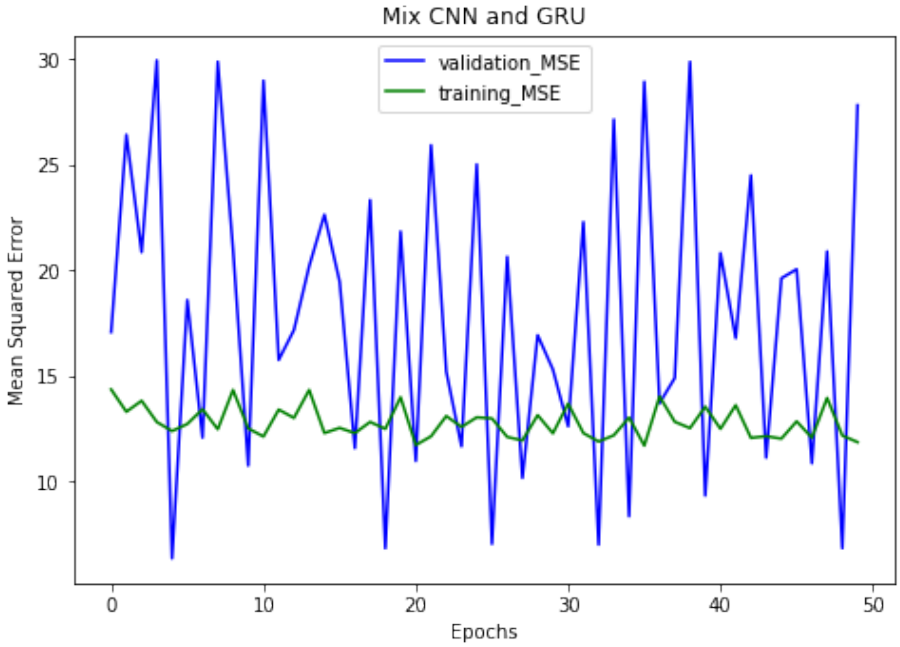

Figure 5: CNN and GRU with data generator and not downsampled

4 Conclusion

As we have seen the laboratory earthquakes are not completely random, since generalized learning is occuring. But most of it is random because the learning improvement before the network goes into over-fitting is pretty small. Since machines are already better at predicting than I would be looking at the data, I am unsure which level of accuracy is possible, but I do have the feeling there are still improvements that could be made, but they are not going to be significantly large. One possible way could be the further improvement on the convolutional approach. Also it might be helpful to use some of my tested network architectures with a data generator to increase the amount of data the network processes.
For even further improvements it might be helpful to look further into the literature.

4.0.1 File Description

In general I conducted every experiment in the beginning with both kinds of hidden units (GRU and LSTM). Then I labled every experiment with a number. In hindsight they are not as systematic as I would wish. Therefore I included a table with the description of all the files.

gru_v1	first try at implementing a network with a data generator
lstm_v1	first try at implementing a network with a data generator
gru_v1_2	downsampled network without scaler
lstm_v1_2	downsampled network without scaler
gru_2_1	downsampled with MinMaxScaler
lstm_2_1	downsampled with MinMaxScaler
gru_2_2	downsampled with Standard Scaler
lstm_2_2	downsampled with Standard Scaler
gru_3_1	improved gru network with dropout
gru_4_1	data generator + improved gru (with dropout)
gru_4_2	data generator + cnn + improved gru (with dropout)
X.csv	downsampled predictors (original can be found on kaggle
Y.csv	downsampled targets
reduce_data	reduces the data from original to X.csv and Y.csv
plots_etc.ipynb	jupyter notebook of the plots
folders contain results, *.sh contain the starting files	

Table 1: File descriptions

References

Anantharaman, P., Qiao, M., and Jadav, D. (2018). Large scale predictive analytics for hard disk remaining useful life estimation. In *2018 IEEE International Congress on Big Data (BigData Congress)*, pages 251–254.

Leeman, Saffer, S. M. (2016). Laboratory observations of slow earthquakes and the spectrum of tectonic fault slip modes. *Nature Communications*, 7.

Rouet-Leduc, B., Hulbert, C., Lubbers, N., Barros, K., Humphreys, C. J., and Johnson, P. A. (2017). Machine learning predicts laboratory earthquakes. *Geophysical Research Letters*, 44(18):9276–9282.