Predicting Laboratory Earthquakes using Deep Learning

Chris Daw Max Ismailov Chris White

Computer Science Department, Western Washington University

Overview

Motivation: Accurate earthquake prediction could save billions of dollars in damages and save countless lives. Deep learning models can likely detect patterns in seismic acoustic data unrecognizable to humans that predict the time until the next earthquake.

Goal: Develop a deep learning model to learn accurately predict the time until the next laboratory earthquake given time series acoustic data.

Approach: Train a Long Short-term Memory Recurrent Neural Network (**LSTM**) to extract features and form an embedding that can be passed to a Convolutional Neural Net (**CNN**) to determine the time until failure of the system (**TTF**) which marks a laboratory earthquake.

Background

- Los Alamos National Laboratory (LANL) has developed a lab system to produce acoustic data similar to a seismograph data for real earthquakes
- Their previous work in earthquake prediction employed machine learning over manually extracted statistical features such as mean, variance, kurtosis, and skewness
- LANL is hosting a competition on Kaggle
- The training dataset LANL provides for the competition consists of 625M tuples consisting of an integer acoustic data value and its corresponding TTF.

Model

LSTM: The LSTM component of our model is a 3 layer stacked LSTM, that takes as input both the acoustic data at a certain time step, the hidden state of itself at a previous time step, and the output of previous LSTM cells in the stack as its input.

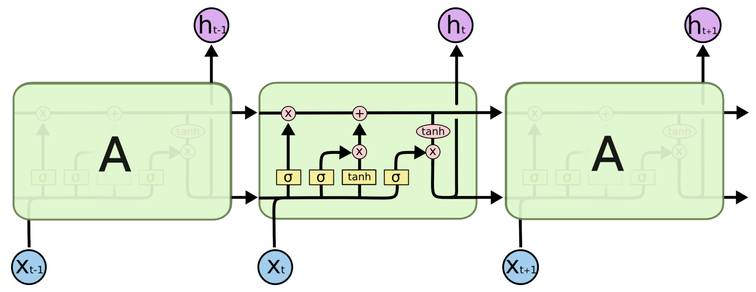

Figure: Diagram of an LSTM model with 3 connected cells

CNN: We use 1-dimensional convolutions meaning our kernel of size 3 will have dimensions 1 x 3 and it only sweeps along in one direction. We also use a small dilation of 1 in the kernel to capture some time dependent trends. The CNN takes in the 1 channel input and increasingly deepens it to 300 channels through convolutional layers, decreases dimensionality further through max pooling, and finally passes through three fully connected layers to produce a single output for each of the 150,000 data-point-long minibatches.

Experimental Setup

The data we recieved from LANL for the Kaggle competition consisted of 625 million lines of data. Originally the data took too long to load so we tried thinning the data by taking every 1,000 data points. However, we did not want to risk inaccuracies so we took made the data smaller by taking the first 150 million lines and saved it as a PyTorch tensor. This loads in a matter of seconds.

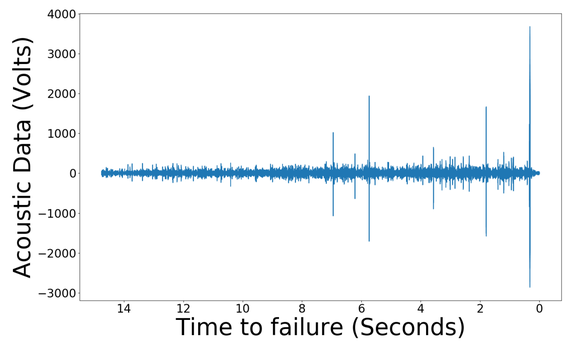

Figure: Plot of a single earthquake event

Since we have more data than we can handle with our available hardware, we do not have to worry about losing data for our dev or test sets. In fact, LANL provides separate chunks of data for testing. In total, they are split up proportionally like so:

train	dev	test
150 million	4.8 million	15 million
88.4%	2.8%	8.8%

Results and Analysis

For a baseline, we measure the Mean Absolute Error (**MAE**) from the average of the entire training data set which gives us **3.048 seconds**.

The LSTM to CNN model converges to a MAE of **3.1 seconds** on average for the training set. Which unfortunately is very close to the MAE of simply guessing the average of the training data. The LSTM layer was likely unable to embed the acoustic data in a meaningful way.

Our CNN model alone performed the best. Its best performance produced a MAE of **0.263 seconds** on our training set and averages **2.5 seconds** on the dev set. We did not have time to run and submit the test set to Kaggle.

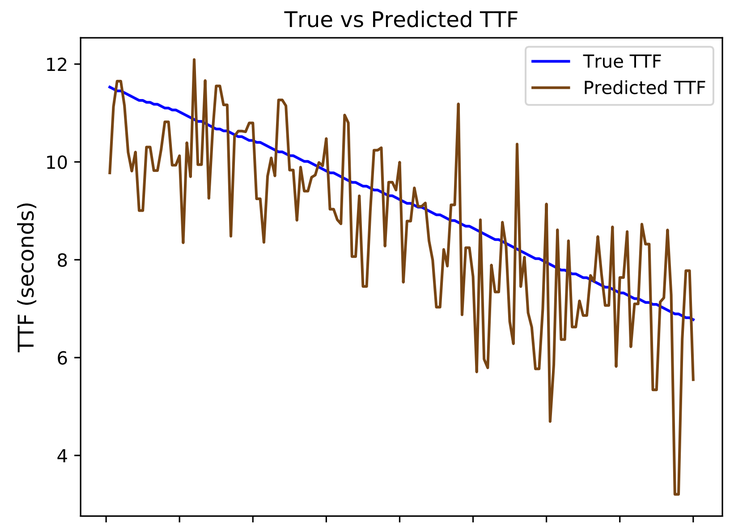

Figure: First 5 seconds of an earthquake event showing the true TTF compared to what our CNN model predicted.

For reference, the lowest MAE on the Kaggle leaderboards for the test sets is 1.238 seconds.

Conclusions & Future Work

Ultimately our model did not generalize past the training set as well as we had hoped. This is likely due to the fact that our data had a very large amount of noise present that our model was simply unable to find patterns in. In addition, we did not have the memory space for our entire dataset so we limited training to a fifth of what was provided to us.

Moving forward, performing more data regularization would likely pay dividends in having our model generalize better; performing a hilbert transform to create a signal envelope could regularize in the ideal fashion and smooth out the erratic behavior of the acoustic signal.

Extracting some features for our model to work with may be valuable as well, even though generally a deeper model should be able to learn to extract features on its own.