# Predicting Laboratory Earthquakes from Acoustic Signal

G030 (s1679883, s1860295, s1882834)

## Abstract

Earthquakes are one of the most catastrophic events in nature and have extensively been studied for decades by Earth scientist to try to understand and predict them to prevent, or at least prepare for, disasters. We take part in a Kaggle competition to predict when Laboratory Earthquakes will occur, making use of an acoustic signal generated by a laboratory earthquake. We feed this signal into 3 baseline models of Random Forest, SVM, and XGBoost to compare its performance against models of CNN LSTM, and Transformer neural networks. Our results indicate that a Transformer neural network is capable of predicting the time before an earthquake with a MAE score of 1.532 seconds, outperforming all the other models individually; an ensemble prediction of the two best models yield an assembled MAE score of 1.496.

## 1. Introduction

Predicting earthquakes has long been studied over the decades in an attempt to reduce the effect of this natural catastrophe. Earlier predictions were relying on averaging the occurrence of earthquakes in a given region, and other measures to come with a probability of a future earthquake (WGCEP, 2007) taking place on a time window that spans over years. Today, most cities that are prone to earthquakes have incorporated early alert earthquake systems that rely on seismometers to identify when an earthquake has taken place and alert neighbor and/or vulnerable cities nearby, giving, in the best case, some seconds to react and try to reduce the disaster. Although these systems definitely present an opportunity to reduce human and financial losses, it continues to be a reactive system that may not always provide enough time to respond. Predicting an earthquake occurrence may extend this time window to give more room to proper responses to reduce damage and save lives, every second counts.

We take part on a Kaggle competition that seeks to predict *when*, in seconds, an earthquake will occur using an acoustic signal as unique data. We believe that machine learning models, specifically LSTM and Transformer networks can be applied to receive information about an earthquake and find patterns on the data to predict information of new earthquakes before they occur. We think that previous information in the acoustic signal can lead to better prediction results, which is why we initially proposed an LSTM neural network (Hochreiter & Schmidhuber, 1997), which can leverage old information. After the results seen in Edinquake (2019), we now propose a combination of two ideas, the use of a Transformer network (Vaswani et al., 2017) and a prediction target scheme called *sequence to point* (S2P) (Zhang et al., 2018). Additionally, we compare the performance of this models against a conventional CNN architecture and a novel CNN architecture named SincNet (Ravanelli & Bengio, 2018). We make use of similar baseline models based on previous work, namely Random Forest, regressive SVM, and XGBoost, to have a benchmark we can use against our Transformer model. Conveniently, work done by Mayer (2019) in using RNNs for this challenge provided us with an useful benchmark from which we could build up our previous LSTM model and improve upon it to finally build a Transformer model.

In this report, we will explain in Section 2 the data that will be used for this challenge as well as any feature engineering we perform on it, then in Section 3 we will briefly introduce the algorithms we use for our models. In Section 4 we will present the experiments performed on our models, followed by Section 5 where we briefly introduce related work that was of help for our project. Finally, in Section 6 we will talk about our findings, understandings, and highlights of our work.

## 2. Data set and task

The datapoints are the result of a continuous piece of experiment data sampled at 4MHz by a piezoceramic sensor, where data is sampled in bins of 4095 elements with a 12ms gap between the $4095^{th}$ and the $4096^{th}$ elements.

The training data is set of roughly 630 million datapoints representing an acoustic signal and time to failure, i.e. time to next earthquake (in seconds), resulting in an estimated of 157.5 seconds of training data for our model. The test data is comprised of 2,624 files, each containing 150,000 datapoints representing only an acoustic signal, meaning we have an estimated of 98.4 seconds of test data to evaluate our model on. The goal is to predict, as precisely as possible, the time to failure of the last acoustic signal's element represented in the test files.

To perform feature engineering, and to train our models as well, we downsample the data into segments of rows, and extract features from those segments, based on work by Inversion (2019). We define a *segment* of data as $n$ consecutive datapoints (i.e., rows) from which we calculate features that represent the given segment. Additionally, we

may or not downsample the data using sliding windows to have more downsampled points from which to train our models. The resulting downsampled data, then, would have a shape of ($\frac{630million-n}{windowSize}$ $x$ $m$), where $n$ is the size of the segment and $m$ is the number of features. For the target data, for each segment of $n$ elements, we take the either last $n^{th}$ element (*sequence to sequence* (S2S)), or the middle element (*sequence to point* (S2P)), resulting in a shape of ($\frac{630million-n}{windowSize}$ $x$ 1). We make use of segments of 150,000 elements with sliding windows of 150,000 and 3,750 elements.

We built three sets of features for our models training, each with a version of the target pointing to the last element of the segment, and the other pointing to the middle element. The first set of features engineering is purely Raw Data (RD), as it consists only of the downsampling of the data to a single acoustic point. The second set we call it Basic Features (BF), comprised of the following eight features for each segment: average, maximum, minimum, standard deviation, quantiles 1, 5, 95, and 99. The third set is an extension of BF, which we call Extended Features (EF), which in addition to the ones of BF, has also the following seven features: median, absolute average, absolute standard deviation, mean absolute deviation, variance, skew, and kurtosis. Additionally, we have normalized our data to prevent our model from getting biased from the datapoints that might skew the data.

To evaluate our models, we will follow the metric used by the competition, which is the Mean Absolute Error (MAE), which translate to the absolute difference between the predicted target of our models and the actual target.

Unless otherwise stated, and for the purpose of improving the performance of our models, we splitted the training data into 90% training and 10% validation.

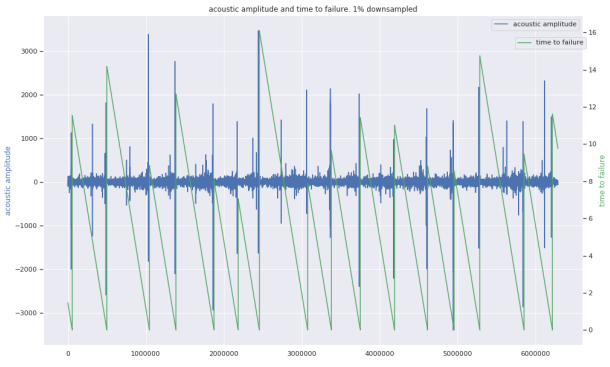


*Figure 1.* Raw data set downsampled to 1%. Acoustic signal and time to failure.

## 3. Methodology

To have a benchmark from which to compare all our result, we created a dummy model trained to predict only the average of the data, using the BF set with S2S targets.

We prepared the following baselines for comparison against our LSTM and Transformer models: a Random Forest, a Support Vector Machine (SVM) with Radial Basis Function (RBF) kernel, and a XGBoost Decision Tree.

Additionally to our LSTM and Transformer models, we implemented a conventional CNN architecture and a SincNet model architecture to evaluate the performance of those against our models.

We performed a series of experiment on each of these models using both S2S and S2P scheme, as it was suggested in the work of Zhang et al. (2018) that an S2P scheme can obtain state-of-the-art results in predicting energy consumption values working with timeseries training data.

### 3.1. Random Forest

Random Forest is an ensemble of multiple Decision Trees that are built to be merged together and generate a more accurate and stable prediction. Additionally, random forest have the advantage of preventing overfitting most of the time, by randomly grouping subsets of features from which a tree in that forest will be grown.

Random forest was used to predict a similar type of laboratory earthquakes (Rouet-Leduc et al., 2017), which is why we decided it would be suitable as a baseline for our work. As we are only interested in having this model as a baseline, we ran this model with default hyperparameter values.

### 3.2. SVM

SVM make use of a hyperplane to calculate boundary lines that minimise the error rate to best classify the data. In regression, however, we make use of the hyperplane and the boundary lines to try to fit the error within a threshold that allows us to have the maximum number of datapoints in it (i.e. the Support Vectors), enabling the model to predict a continues value. In both cases, a kernel is used to map the data to a higher dimension and calculate the best hyperplane.

A simple benchmark with SVM (Inversion, 2019) was published in the challenge's forum to give a reference starting points for participants. We decided to use this as a blue print for our SVM benchmark, although we used our own engineered features to obtain better results. Other than the extra features, we decided to default the hyperparameters as done with the random forest.

### 3.3. XGBoost

Extreme Gradient Boost uses the Gradient Boosting Algorithm (GBM) to train models in an additive way to identify shortcomings in the model by using gradients in the loss function. XGBoost also allows for scalability, making easy to handle datasets with billions of datapoints, which makes it a promising benchmark for our work.

We implemented a simple XGBoost regressor using the XGB python package (Chen & Guestrin, 2016), using default parameters as with the previous models.

### 3.4. LSTM

LSTMs are a type of Recurrent Neural Networks (RNN), with the ability to *remember* long-term dependencies. They do this by using what are known as *gates* that allow the network to *forget*, *learn*, and *filter* data, chaining the information across the network.

Due to their good performance in many problems, they have gained significant importance and increasing popularity. Specifically, they have been used extensively in problems that involve timeseries, where relations in the information could be based on old data, making the ability to remember key to a good prediction.

For such reasons, we aim to implement an LSTM network to work with the acoustic signal, and other features we create, to predict, for any given data point, the time remaining until next earthquake. The architecture of our LSTM network is based on work done by Mayer (2019). This network consists of three parts: a fully connected single hidden layer with 100 units using ReLu activation function, followed by an LSTM network with one hidden layer with 100 units and having 0.25 drop-in and 0.25 drop-out rate, and finally a fully connected hidden layer of 10 units using ReLu as activation function and then pass to a fully connected single output unit for prediction.

### 3.5. Transformer

Building on the success of gated RNNs such as LSTM and GRU to sequence modelling tasks, the transformer (Vaswani et al., 2017) is a model architecture eschewing recurrence and instead relying entirely on an attention mechanism to draw global dependencies between sequential input and output.

The full model consists of an encoder-decoder structure with self-attention mechanism. Here, the encoder maps an input sequence of symbol representations $(x_1, ..., x_n)$ to a sequence of continuous representations $z = (z_1, ..., z_n)$. Given $z$, the decoder then generates an output sequence $(y_1, ..., y_m)$ of symbols one element at a time. At each step the model is auto-regressive (Graves, 2013), consuming the previously generated symbols as additional input when generating the next. The Transformer follows this overall architecture using stacked self-attention and point-wise, fully connected layers for both the encoder and decoder (Figure 2).

For our sequence-to-point prediction task, we use only the encoder, which is a stack of N identical layers. Each layer has two sub-layers. The first is a multi-head self-attention mechanism, and the second is a simple, position-wise fully connected feed-forward network. Residual connections (He et al., 2015) are employed around each of the two sub-layers, followed by layer normalization (Lei Ba et al., 2016). That is, the output of each sub-layer is LayerNorm($x$ + Sublayer($x$)), where Sublayer($x$) is the function implemented by the sub-layer itself. To facilitate these residual connections, all sub-layers in the model, as well as the embedding layers, produce outputs of the same dimension. We refer the reader to Vaswani et al. for full details of the attention mechanisms.

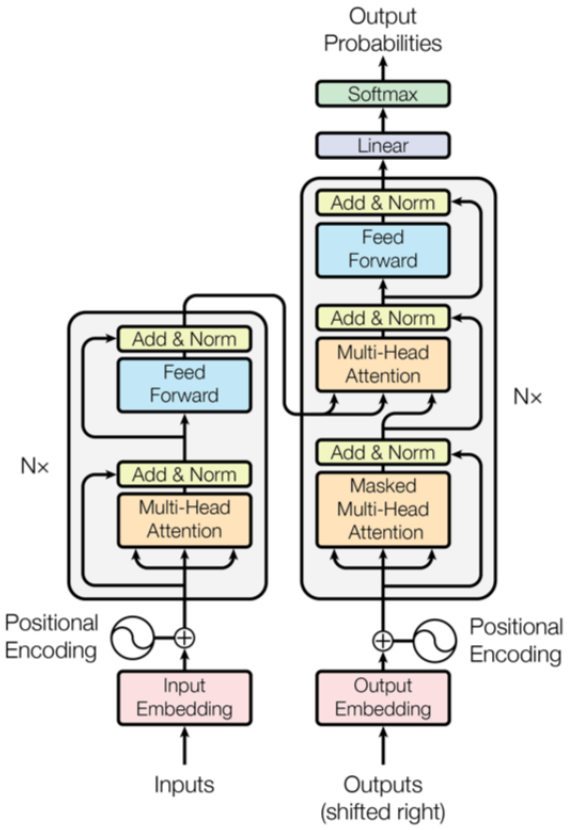


*Figure 2.* Architecture of the Transformer. The encoder is in the left half, and the decoder is in the right.
(Vaswani et al., 2017)

### 3.6. CNN and SincNet

Recent studies have shown deep neural networks that take in raw audio are promising for speech (Palaz et al., 2015; Sainath et al., 2015; Hoshen et al., 2015; Sainath et al., 2015; Tüske et al., 2014), including emotion tasks (Trigeorgis et al., 2016), speaker recognition (Muckenhirn et al., 2018), and speech synthesis (van den Oord et al., 2016; Mehri et al., 2016). Rather than employing standard handcrafted features, the networks learn low-level speech representations from waveforms, potentially allowing them to better capture important narrow-band speaker characteristics such as pitch and formants. Here we follow this line of work that uses raw audio waveform as input and investigate whether this approach to process in the input signals could work well for our regression task as well.

Standard CNN learns all elements of the filter vector. The first layer of a standard CNN applies time-aligned convolutions to the input waveform. Each convolution is defined as follows:

$$y[n] = x[n] * h[n] = \sum_{l=0}^{L-1} x[l].h[n-l] \qquad (1)$$

where $x[n]$ is a chunk of the speech signal, $h[n]$ is the filter of length $L$, and $y[n]$ is the filtered output. All the L elements (taps) of each filter are learned from data. In contrast, SincNet (Figure 3) performs the convolution with a predefined function $g$ that depends on few learn-able parameters $\theta$ only:

$$y[n] = x[n]g[n, \theta] \tag{2}$$

where $g$ is chosen to be the sinc function, inspired by standard filtering in digital signal processing, in order to employ rectangular band-pass filters. Contrary to standard CNNs that often learn filters with undesired and noisy multi-band shapes which do not appeal to human intuition, nor ideal as efficient representation of the input signal, SincNet enforces constraints on the shape of the filters and thus converges faster while learning filters that are more interpretable (Ravanelli & Bengio, 2018).

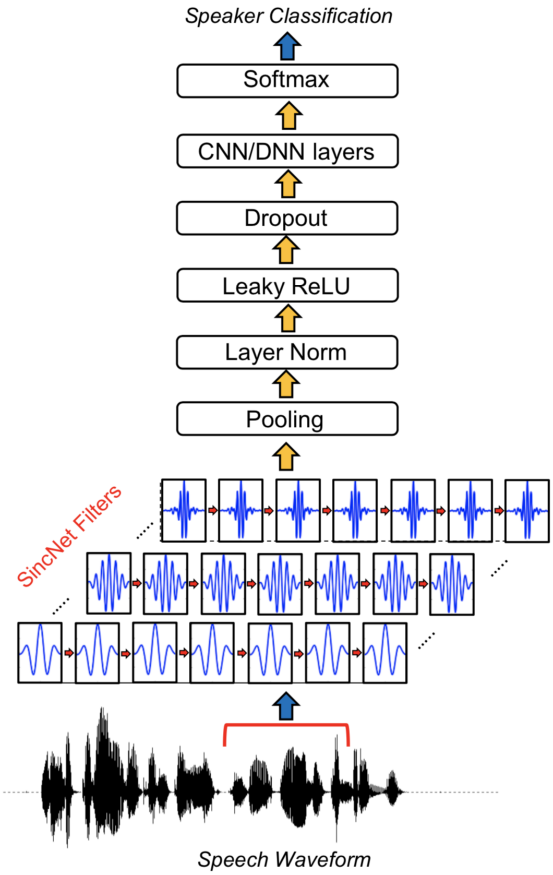


*Figure 3.* Architecture of SincNet. (Ravanelli & Bengio, 2018)

## 4. Experiments

We conducted 2 experiments involving the Random Forest, SVM, and XGBoost. These experiments were run with the sliding version of the EF (SEF) data, using both the S2S and S2P targets, as we had already baseline results using the BF set from the work done in Edinquake (2019). We ran these models on a Jupyter notebook directly in the Kaggle environment as it allows us to make use of a virtual session that provides computing resources enough for these models. These experiments were all executed using the default hyperparameters of the *scipy* python package and performances was only calculated on the Train and Test Set, no Validation Set was used for these models.

Table 1 shows the results on the Train Set and Test Set of our baseline models, and we can see that our baseline models are good enough to completely outperform the dummy baseline.

| Model | Dataset | Target | Train Score | Test Score |
|---|---|---|---|---|
| Dummy | SEF | S2P | – | 3.68 |
| RF | SEF | S2P | 0.9 | 1.68 |
| SVM | SEF | S2P | 2.041 | 1.536 |
| XGB | SEF | S2P | 2.044 | 1.558 |

*Table 1.* Baseline results on Train and Test set. Scores represent MAE in seconds.

From our baselines experiments, we noticed that performances tends to improve when training is done with the EF set, and even more with the SEF set, however the training time takes longer, so in order to find the best model without running many unnecessary experiments, we did an initial comparison of our LSTM and Transformer models in the BF set, using both the S2S and S2P targets. Then we would take the best model on the validation set to tune its hyperparameters, again, in the BF set, and finally try the best version in the SEF set with S2S and S2P targets.

Table 2 show the performance on the Validation set of our LSTM, Transformer, CNN, and SincNet models, from which we select the best model for hyperparameter tuning and evaluation on the Test Set.

| Model | Dataset | Target | Val Score |
|---|---|---|---|
| Transf | BF | S2S | 2.26 |
| Transf | BF | S2P | 2.19 |
| LSTM | BF | S2S | 2.34 |
| LSTM | BF | S2P | 2.27 |
| LSTM | RD | S2P | 2.75 |
| CNN | RD | S2S | 2.57 |
| CNN | RD | S2P | 2.49 |
| SincNet | RD | S2S | 2.71 |
| SincNet | RD | S2P | 2.64 |

*Table 2.* Models results on Validation Set. Scores represent MAE in seconds.

The best model from this stage was the Transformer using BF and S2P targets following the architecture described in 3.5. We proceed to tune hyperparameters of this model, mainly we changed the optimizer to be AMSGrad instead of Adam, as it is supposedly an improvement over Adam with Weight Decay, and the results were not disappointing. We also changed the transformer heads to be 16, and used batch normalization instead of layer normalization. With this new

fixed hyperparameters, we conducted more experiments changing the number of layers and the batch size, with the results available in Table 3.

| Model | Layers | BatchSize | Val Score |
|---|---|---|---|
| Transf | 4 | 64 | 2.15 |
| Transf | 5 | 64 | 2.59 |
| Transf | 4 | 128 | 2.23 |
| Transf | 4 | 32 | 2.11 |
| Transf | 4 | 16 | 2.10 |
| Transf | 4 | 8 | 2.06 |
| Transf | 4 | 4 | 2.09 |

*Table 3.* Transformer models results on Validation Set after tuning. Scores represent MAE in seconds.

Finally, to choose a model of Transformer to be use in the Test Set, we ran the best architecture from the results on Table 3 using the BF, EF, SBF, and SEF sets. Additionally, for the case of the run with BF and EF, we ran the experiments with both S2S and S2P targets. These allowed us to confirm that the S2P targets make our model yield better predictions, as can be seen in Table 4.

| Model | Dataset | Targets | Val Score |
|---|---|---|---|
| Transf | BF | S2S | 2.07 |
| Transf | BF | S2P | 2.06 |
| Transf | EF | S2S | 2.04 |
| Transf | EF | S2P | 2.03 |
| Transf | SBF | S2P | 1.99 |
| Transf | SEF | S2P | 1.92 |

*Table 4.* Models results on Validation Set after tuning. Scores represent MAE in seconds.

The best Transformer model, then, was the one with that was trained with AMSGrad optimizer, Batch Size of 8, 16 Transformer Heads, 4 layers, and batch normalization, maintaining the other architecture characteristics described earlier.

The best LSTM model was the one with the following architecture and hyperparameters: one fully connected hidden layer with 100 ReLu units, followed by an LSTM network with one hidden layer of 100 units and finally a fully connected hidden layer with 10 ReLu units which are fully connected to a single output units. This model is trained with AMSGrad optimizer, learning rate of $10^{-4}$ and batch size of 16. Table 5 shows how this model perform with different data sets. This model perform the best with SEF S2P data set when evaluate on validation set.

As we have only a limited amount of submissions per day, we did not evaluated all of the models on it. Table 6 show the evaluation of the best models we find, on the Train Set and Validation Set, performing on the Test Set provided by Kaggle. From there, we can see the best model is the Transformer model with a MAE score of 1.532. Additionally, we can see an even better performer represented by

| Model | Dataset | Targets | Val Score |
|---|---|---|---|
| LSTM | BF | S2P | 2.19 |
| LSTM | EF | S2P | 2.16 |
| LSTM | SBF | S2P | 2.06 |
| LSTM | SEF | S2P | 1.99 |

*Table 5.* LSTM model results on different data set evaluate on Validation set. Scores represent MAE in seconds.

the *Ensembled Model* with a MAE score of 1.496, which represents the assembled predictions of the best SVM and Transformer models.

| Model | Dataset | Target | Test Score |
|---|---|---|---|
| Dummy | SEF | S2P | 3.68 |
| RF | SEF | S2P | 1.68 |
| SVM | SEF | S2P | 1.536 |
| XGB | SEF | S2P | 1.558 |
| LSTM | SEF | S2P | 1.538 |
| Transf | SEF | S2P | 1.532 |
| Ensemble | SEF | S2P | 1.496 |

*Table 6.* Dummy and Best models results on Test Set. Scores represent MAE in seconds. Ensemble model is the assembled predictions of best SVM and Transformer model.

## 5. Related Work

Recently, Asim et al. (2018; 2017) worked with Machine Learning algorithms such as Random Forest, Recurrent Neural Networks (RNN), and regressive SVM to attempt to predict time to earthquakes occurrence using a wide range of seismic timeseries. In a similar work, but focusing in *where* rather than *when*, M. R. DeVries et al. (2018) apply Machine Learning algorithms to predict location of aftershocks that follow earthquakes, as these too represent a huge threat.

More related to this competition, a series of work has been done by the Los Angeles National Laboratory (LANL) in predicting occurrence and intensity of laboratory earthquakes by means of Machine Learning algorithms using solely the acoustic signal as an input, and then performing feature engineering over it to feed the models with. In Rouet-Leduc et al. (2017), the authors state that their work is first of its kind in using acoustic signal of earthquake experiments to attempt prediction of when an earthquake will occur, they use Random Forest for this purpose by using the acoustic signal of a laboratory earthquake as the single input from which other features were derived for the model to predict an accurate time result. In Rouet-Leduc et al. (2018), they make use of the same acoustic signal as input to feed an XGBoost Decision Tree to predict the fault frictional state, which could derive in estimating the magnitude of such earthquake. In yet another related work, Hulbert et al. (2019) make use of gradient boosted trees to

predict the time to an earthquake, magnitude, and duration, using the acoustic signal of the earthquake as single input.

## 6. Conclusions

We found that Machine Learning models can be used to predict laboratory earthquakes using an acoustic signal of the quake as input, performing feature engineering on it to allow the models to infer relations and provide a prediction from it. Complex models, like LSTMs and Transformers, result in better performance although it may take longer to train them compared to a simpler model like XGBoost that can be trained in a fraction of the time, although with 10% of the performance being compromised.

Feature engineering is key to improve the performance of all our models both to save computational resources and to obtain better predictions. It is also one of the most complex tasks as there are many features one could obtain from the data, but not all of them will be of use for the model to learn a relationship from the data.

An appropriate downsampling of the data is also of fundamental importance for our models to work faster and more precisely. Sliding windows allow our models to smoothly change its attention throughout the data to not skip potential relationships across segments. Target selection was surprisingly helpful as it managed to improve results in all of our models.

Assembling predictions can benefit the predictions' accuracy as it removes potential biases that each model may have, yielding a more objective and precise prediction.

Although results are not directly comparable to the existent literature, mainly because of the data being used and the different metrics for evaluation, we can compare our results to other participants in the competition, where we currently stand in the top 30%.

## References


Asim, Khawaja, MartÃŋnez-ÃĄlvarez, Francisco, Basit, Abdul, and Iqbal, Talat. Earthquake magnitude prediction in hindukush region using machine learning techniques. *Natural Hazards*, 85:471–486, 01 2017. doi: 10.1007/s11069-016-2579-3.

Asim, Khawaja M., Idris, Adnan, Iqbal, Talat, and MartÃŋnez-ÃĄlvarez, Francisco. Earthquake prediction model using support vector regressor and hybrid neural networks. *PLOS ONE*, 13(7):1–22, 07 2018. doi: 10.1371/journal.pone.0199004. URL https://doi.org/10.1371/journal.pone.0199004.

Chen, Tianqi and Guestrin, Carlos. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pp. 785–794, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4232-2. doi: 10.1145/2939672.2939785. URL http://doi.acm.org/10.1145/2939672.2939785.

Edinquake, Solana, Chiehmin Pang. Predicting laboratory earthquakes from acoustic signal: Interim report, February 2019.

Graves, Alex. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013. URL http://arxiv.org/abs/1308.0850.

He, Kaiming, Zhang, Xiangyu, Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

Hochreiter, Sepp and Schmidhuber, Jürgen. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL http://dx.doi.org/10.1162/neco.1997.9.8.1735.

Hoshen, Y., Weiss, R. J., and Wilson, K. W. Speech acoustic modeling from raw multichannel waveforms. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4624–4628, April 2015. doi: 10.1109/ICASSP.2015.7178847.

Hulbert, Claudia, Rouet-Leduc, Bertrand, Johnson, Paul A., Ren, Christopher X., RiviÃĺre, Jacques, Bolton, David C., and Marone, Chris. Similarity of fast and slow earthquakes illuminated by machine learning. *Nature Geoscience*, 12(1):69–74, 1 2019. ISSN 1752-0894. doi: 10.1038/s41561-018-0272-8. URL https://doi.org/10.1038/s41561-018-0272-8.

Inversion. Basic feature benchmark, Jan 2019. URL https://www.kaggle.com/inversion/basic-feature-benchmark.

Lei Ba, Jimmy, Kiros, Jamie Ryan, and Hinton, Geoffrey E. Layer Normalization. *arXiv e-prints*, art. arXiv:1607.06450, Jul 2016.

M. R. DeVries, Phoebe, ViÃl'gas, Fernanda, Wattenberg, Martin, and Meade, Brendan. Deep learning of aftershock patterns following large earthquakes. *Nature*, 560, 08 2018. doi: 10.1038/s41586-018-0438-y.

Mayer, Michael. Basic feature benchmark, Jan 2019. URL https://www.kaggle.com/mayer79/rnn-starter-for-huge-time-series.

Mehri, Soroush, Kumar, Kundan, Gulrajani, Ishaan, Kumar, Rithesh, Jain, Shubham, Sotelo, JosÃČÂl', Courville, Aaron, and Bengio, Y. Samplernn: An unconditional end-to-end neural audio generation model. 12 2016.

Muckenhirn, Hannah, Magimai-Doss, Mathew, and Marcel, Sébastien. Towards directly modeling raw speech signal for speaker verification using cnns. *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4884–4888, 2018.

Palaz, Dimitri, Magimai.-Doss, Mathew, and Collobert, Ronan. Analysis of cnn-based speech recognition system using raw speech as input. In *Proceedings of Interspeech*, pp. 11–15. ISCA, ISCA, September 2015.

Ravanelli, Mirco and Bengio, Y. Speech and speaker recognition from raw waveform with sincnet, 12 2018.

Rouet-Leduc, Bertrand, Hulbert, Claudia, Lubbers, Nicholas, Barros, Kipton, Humphreys, Colin J., and Johnson, Paul A. Machine learning predicts laboratory earthquakes. *Geophysical Research Letters*, 44 (18):9276–9282, 2017. doi: 10.1002/2017GL074677. URL https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/2017GL074677.

Rouet-Leduc, Bertrand, Hulbert, Claudia, Bolton, David C., Ren, Christopher X., Riviere, Jacques, Marone, Chris, Guyer, Robert A., and Johnson, Paul A. Estimating fault friction from seismic signals in the laboratory. *Geophysical Research Letters*, 45(3):1321–1329, feb 2018. doi: 10.1002/2017gl076708. URL https://doi.org/10.1002/2017gl076708.

Sainath, T. N., Weiss, R. J., Wilson, K. W., Narayanan, A., and and, M. Bacchiani. Speaker location and microphone spacing invariant acoustic modeling from raw multichannel waveforms. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pp. 30–36, Dec 2015. doi: 10.1109/ASRU.2015.7404770.

Sainath, Tara N., Weiss, Ron J., Senior, Andrew W., Wilson, Kevin W., and Vinyals, Oriol. Learning the speech front-end with raw waveform cldnns. In *INTERSPEECH*, 2015.

Trigeorgis, George, Ringeval, Fabien, Brueckner, Raymond, Marchi, Erik, Nicolaou, Mihalis, Schuller, BjÃČÂűrn, and Zafeiriou, Stefanos. Adieu features? end-to-end speech emotion recognition using a deep convolutional recurrent network. 03 2016. doi: 10.13140/RG.2.1.3842.7283.

Tüske, Zoltán, Golik, Pavel, Schlüter, Ralf, and Ney, Hermann. Acoustic modeling with deep neural networks using raw time signal for lvcsr. In *INTERSPEECH*, 2014.

van den Oord, Aäron, Dieleman, Sander, Zen, Heiga, Simonyan, Karen, Vinyals, Oriol, Graves, Alex, Kalchbrenner, Nal, Senior, Andrew W., and Kavukcuoglu, Koray. Wavenet: A generative model for raw audio. In *SSW*, 2016.

Vaswani, Ashish, Shazeer, Noam, Parmar, Niki, Uszkoreit, Jakob, Jones, Llion, Gomez, Aidan N., Kaiser, Lukasz, and Polosukhin, Illia. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.

WGCEP. 2007 working group on california earthquake probabilities, Jan 2007. URL https://www.sciencebase.gov/catalog/item/4f4e4abce4b07f02db672e8e.

Zhang, Chaoyun, Zhong, Mingjun, Wang, Zongzuo, Goddard, Nigel H., and Sutton, Charles A. Sequence-to-point learning with neural networks for non-intrusive load monitoring. In *AAAI*, 2018.