Final Project Report
Building An Earthquake Early Warning System

Yajun Peng
Department of Geosciences
yajun@princeton.edu

Wenjie Lei
Department of Geosciences
lei@princeton.edu

Abstract

Large earthquakes can cause an immense amount of casualties and damages. To mitigate hazard in seismically active regions, we build an prototype earthquake early warning (EEW) system using machine learning approaches. We focus on southern California, and use the ground motion data from thousands of earthquakes recorded by hundreds of seismic stations. The goal is to issue fast and reliable reports of earthquake magnitudes to regions susceptible to strong ground shaking before it arrives. We test several different models including lasso regression, ridge regression, Elastic net, random forest and neural network. We show that after feature engineering, these models greatly outperform those using a few empirical measurements employed by currently operating EEW systems. We find that random forest slightly outperforms linear models. However, these models generally underestimate the magnitude of the largest earthquakes ($>$ M6). Possible reasons for this are discussed. Neural network...

1 Introduction

In the past two decades, multiple devastating earthquakes occurred in densely populated regions, including the 2004 M9.2 Sumatra, 2008 M7.9 Wenchuan, 2011 M9.1 Tohoku-Oki, and 2015 M7.8 Nepal earthquakes. The resulting strong ground shaking and tsunami caused deaths of thousands of people, and property damages worth of billions of dollars. Earthquake early warning (EEW) has the potential to provide a few seconds to a few minutes of warning before the strongest ground shaking arrives at a given location. It takes advantage of the difference in propagation speeds between seismic and electromagnetic waves [2, 4]. The alert time can be used to adopt safety measures by individuals, to reduce train derailments, to secure chemical and radioactive materials, and to protect children in schools and patients in hospitals [8]. Given the lack of reliable predictions for earthquake occurrences, EEW remains the primary avenue for short-term hazard mitigation.

There are several active warning systems operating in Mexico, Japan, Taiwan, Turkey and Romania. In addition, implementation and testing of such systems are under development in United states, Europe and Asia [2, 8]. However, currently the adopted methods for EEW generally rely on empirical relations between earthquake magnitudes and several measurements derived from the recorded seismograms. Generally, the standard deviation of the prediction error in magnitude is reported to be around 0.3-0.5 unit (corresponding to a 3-6 times difference in energy release) [3].

In this work, we propose to explore the possibility of utilizing various machine learning techniques to obtain more reliable estimates of earthquake magnitudes. Results using previous EEW methodswould serve as the baseline.

2 Related Work

The measurements often adopted in currently operating EEW systems include the predominant frequency ($tau_p{}^{max}$, tau_c) [1, 7] and the peak amplitude of P waves (P_d, P_v), the seismic waves that arrive first but cause weak shaking [10]. The latter is usually corrected for the effect of geometric spreading (energy decreasing with distance). The distance-corrected P_d tend to underestimate the magnitude of large earthquakes [10]. Although $\tau_p{}^{max}$ and τ_c seem not to suffer from this, they are not very robust and often lead to a large false positive rate [10]. These measurements will be shown in the following sections.

3 Methodology

3.1 Data Processing

Here we focus on the southern California regions, where abundant seismicity and a dense network of seismic stations exist. Figure 1 shows thousands of earthquakes with magnitude larger than 3 since 1990. There are 440 stations in total, but the number of operating stations varies with time. Therefore, we adopt the strategy of treating the recordings of an earthquake at different individual stations as independent data. We note that in practice the predictions from multiple stations can be combined to obtain a more robust estimate.

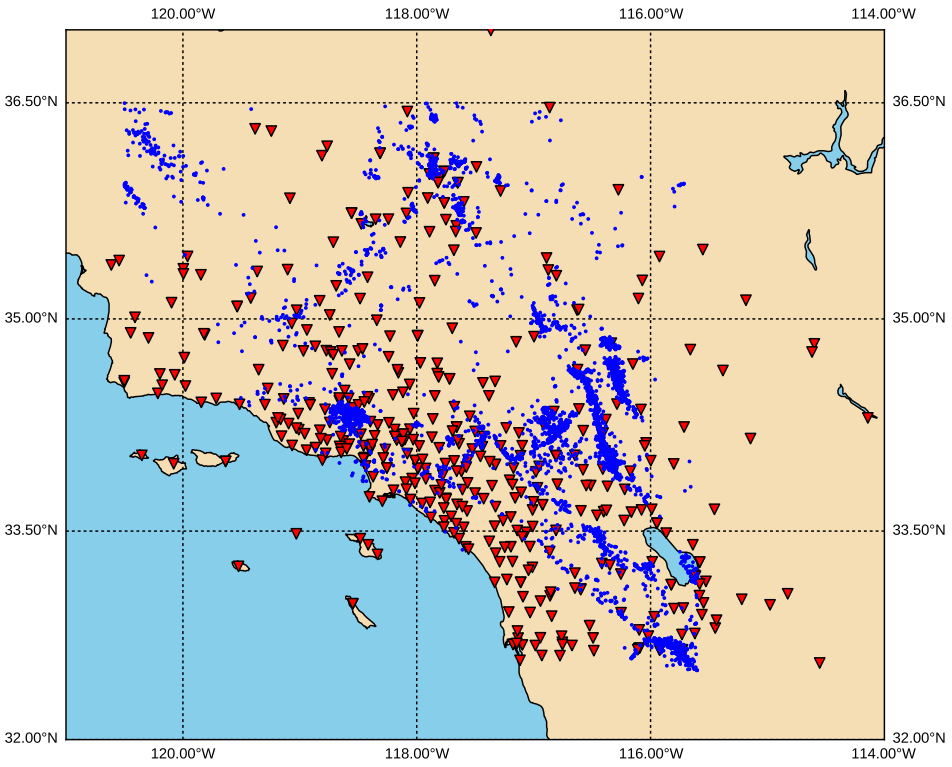

Figure 1: **Stations (red triangles) and earthquakes (blue dots) distribution since 1990 in the study region.** For clarity, only events with magnitude larger than 3.0 are plotted.

Data processing consists of the following steps:

1. Pick the P wave arrival using a standard STA/LTA algorithm [9], which has been widely used in seismology. An example is shown in Figure 2.
2. Remove instrument gain such that the seismogram amplitude is within the correct range.

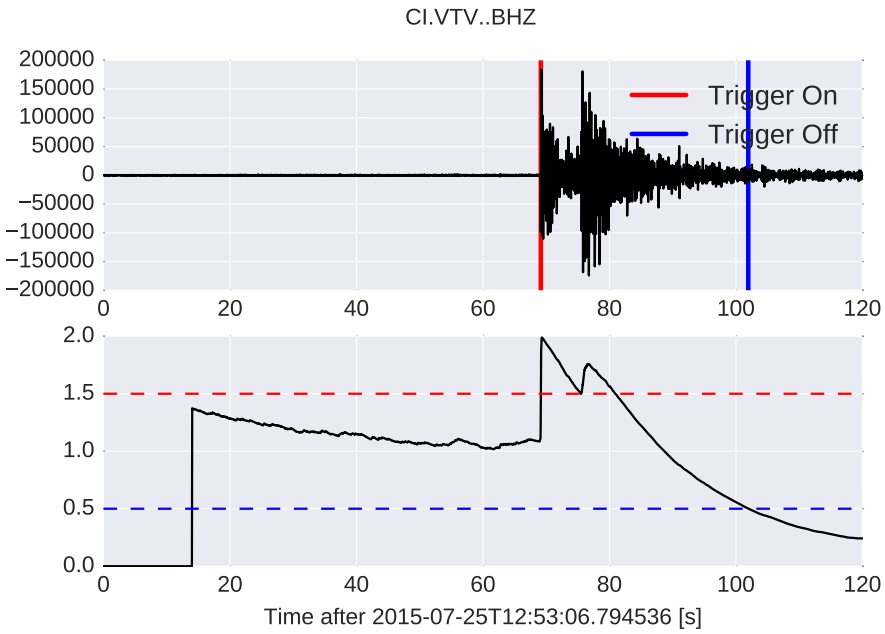

Figure 2: **P wave arrival during a M4.17 earthquake picked by a recursive STA/LTA method.** (Upper) Seismogram at the station VTV (vertical component). A highpass filter (> 1 Hz) is applied. The red line shows the P wave arrival. (Lower) Characteristic function of the recursive STA/LTA. The red dashed line shows the trigger threshold.

3. Integrate or differentiate the data as appropriate to obtain displacement, velocity and acceleration seismograms for each station.
4. Derive features from the seismograms for training linear and random forest models. Details of these features are discussed in Section 5.2.
5. Neural network...
6. The data are divided into a training set and a test set. We find that earthquakes with magnitude larger than 4.5 constitute only 3% of the total data volume. Since we are particularly interested in testing the predicting power on larger earthquakes, we keep in the test set 30% of the data with magnitude ≥ 4.5, and 10% of those with magnitude < 4.5.

3.2 Methods and Evaluation Metrics

In this project, several different well-established machine learning algorithms will be used. Since this is inherently a regression problem, we use several linear regressors (ridge, lasso, elastic net) and a tree-based method (random forest). To push our project further, we also use neural network in our prediction model. Recurrent Neural Network(RNN) has been successfully used in speech recognition, which is intrinsically similar to our problem in that both utilize the "waveforms" as input information. We will evaluate the performance of these model with mean squared error.

4 Spotlight Regressor: Random Forest

The basic idea of random forests is to build and average a collection of de-correlated trees [5, 6]. It is closely related to bootstrap aggregation or bagging. Bagging averages the prediction from models trained with bootstraped samples in order to reduce its variance. This is particularly suitable for high-variance, low-bias models, such as decision trees. If each tree can be viewed as independent and identically distributed (i.i.d) random variables with variance σ^2, then averaging B such trees would result in the same expectation as the individual trees and a reduced variance σ^2/B. However, it is often likely that these trees are not independent, with pairwise correlation ρ, then the variance

of the average is

$$\rho\sigma^2 + (1-\rho)\sigma^2/B. \tag{1}$$

When bagging is used, increasing B would reduce the contribution from the second term, but the first term remains constant. Therefore, random forest aims to further reduce the prediction variance by decreasing the correlation between different trees [6]. This is achieved by randomly selecting a subset of all the input variables (features) for each terminal node of the tree.

The basic algorithm of random forest is as follows [6]:

For b = 1, ..., B:

1. Draw a sample with replacement from the training set.
2. Train a tree model T_b to the bootstrapped data. For each node of the tree, pick the best variable among the randomly selected subset of the features, and split the node into two daughter nodes.

For a regression problem, the prediction is the average of the predictions from the ensemble of trees (Figure 3):

$$\hat{f}(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x) \tag{2}$$

For classification, the prediction is

$$\hat{C}(x) = majority\ vote\{C_b(x)_{b=1}^{B}\}, \tag{3}$$

where $C_b(x)$ is the class prediction of the bth tree.

Unlike many other machine learning method, quantifying generalization error or tuning hyperparameters for random forest do not require N-fold cross validation. Random forest uses out-of-bag (OOB) samples. For each observation, it constructs the predictor by averaging only those trees corresponding to bootstrap samples where the observation is not included. It can be shown that an OOB error estimate is very similar to the error obtained using N-fold cross validation. Hence, we can fit random forests in one sequence, which greatly reduces the computational expense.

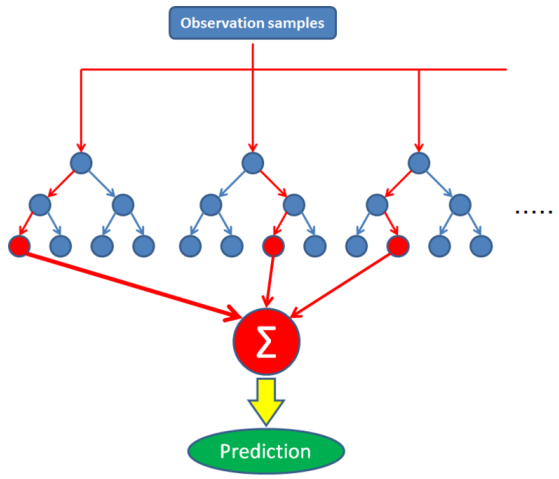

Figure 3: **A schematic illustration for the prediction process for a random forest regressor.**

5 Results

5.1 Empirical Measurements

In this section, we report several popular empirical measurements that have been used in literature. Figure 4 shows the logarithm of distance-corrected P_d ($P_d * r^2$, where r is the distance from the earthquake hypocenter to the station), τ_c and τ_p^{max}.

We observe that there appears to be a quasi-linear relation between $P_d * r^2$ and magnitude. We perform a linear fit (dahsed line in Figure 4), the magnitudes of large earthquakes (M $>$ 4.5) are in general underestimated. This indicates that the slope decreases with magnitude. [10] suggested that this might be caused by the fact that the source dimension of large earthquake is finite (not well approximated by a point). Therefore, seismic energy released by large portions of the source area may travel longer than r. In addition, the majority of these large earthquakes occurred before 2000, when the station distribution was very sparse. The stations were far from the earthquakes such that attenuation effect of the earth may be significant, which further reduces the seismic signal amplitude.

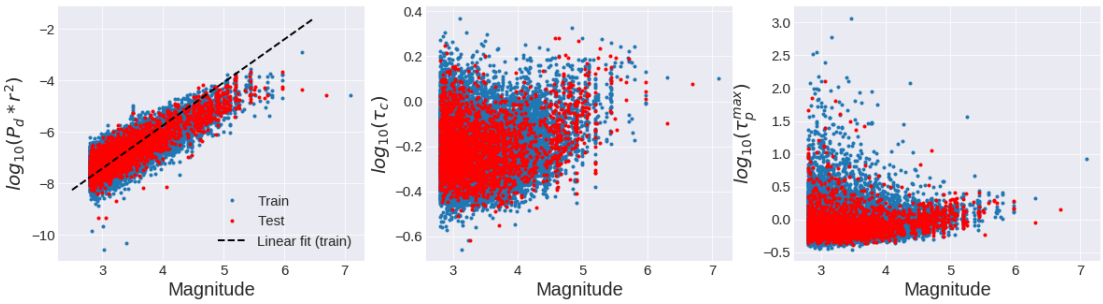

Figure 4: **Three empirical measurements ($P_d * r^2$, τ_c, and τ_p^{max}) as a function of magnitudes.**

We also observe a quasi-linear trend for τ_c and τ_p^{max} for earthquakes larger than M4. However, the significant scatter for the smaller earthquake excludes the possibility of achieving good predicting performance using a simple linear fit. This is consistent with findings in previous studies [10]. Hence, in the following sections, we use only the linear fit for $P_d * r^2$-magnitude as the baseline.

5.2 Linear Models and Random Forest

5.2.1 Feature Engineering

To train machine learning models, we wish to incorporate features other than few popular empirical measurements (Figure 4). Here only broadband velocity seismograms are used. The velocity records are then integrated to displacement and differentiated to acceleration. We also make velocity envelopes. Each of these four types of records has three components (east, north, and vertical). For each component, we use 0.5, 1, 1.5, ... to 3-second time windows since the arrival time picked by the STA/LTA algorithm Figure 2. For each time window, we compute the maximum, the relative time within a window for the maximum, mean, variance, skewness, kurtosis, percentile (0.25, 0.5, 0.75), and norm (L1, L2, L4), τ_c and τ_p^{max}. The distance from the earthquake hypocenter to the station (r) is also included. We have in total 1663 features. We note there may be many redundant features, especially for amplitude information. It is possible that the performance could be improved if we incorporate more frequency-related features. But currently we use only τ_c and τ_p^{max}. For random forest, the amplitude-related features are corrected by the earthquake-station distance. Most of these features are transformed into the logarithmic scale. Exceptions include skewness and kurtosis.

5.2.2 Prediction Performance

We use 5-fold cross validation to tune the hyperparameters for the linear models (lasso, ridge, elastic net), and out-of-bag scores for random forest. Then the models are used to predict the earthquake magnitudes in the test set.

The evaluation metrics for the regressors are shown in Table 1. All the machine learning methods outperform the baseline, with random forest being slight better than the linear models. This may suggest a weak non-linear relation between the features and the magnitudes. As expected, the lasso regression and elastic-net regression produce the most sparse models, with only 347 and 349 features with non-zero coefficients (linear models) or importance (random forest).

Regressor	MSE	MSE ($<$ M4.5)	MSE ($\geq$ M4.5)	Error Mean ($\geq$ M4.5)	Non-zero features
Baseline	0.0899	0.0695	0.2620	-0.4250	1
RF	0.0451	0.0388	0.0981	-0.1536	1663
Lasso	0.0565	0.0477	0.1309	-0.2470	347
Ridge	0.0560	0.0477	0.1260	-0.2421	1663
EN	0.0565	0.0476	0.1310	-0.2470	349

Table 1: **Results from the adopted regressors on the test set.** The first three columns show mean squared error (MSE) for the whole test set, the small earthquakes ($<$ M4.5) and the large earthquakes ($\geq$ M4.5). The mean error for large earthquakes and the number of features with non-zero coefficients (linear models) or importance (random forest) are also reported.

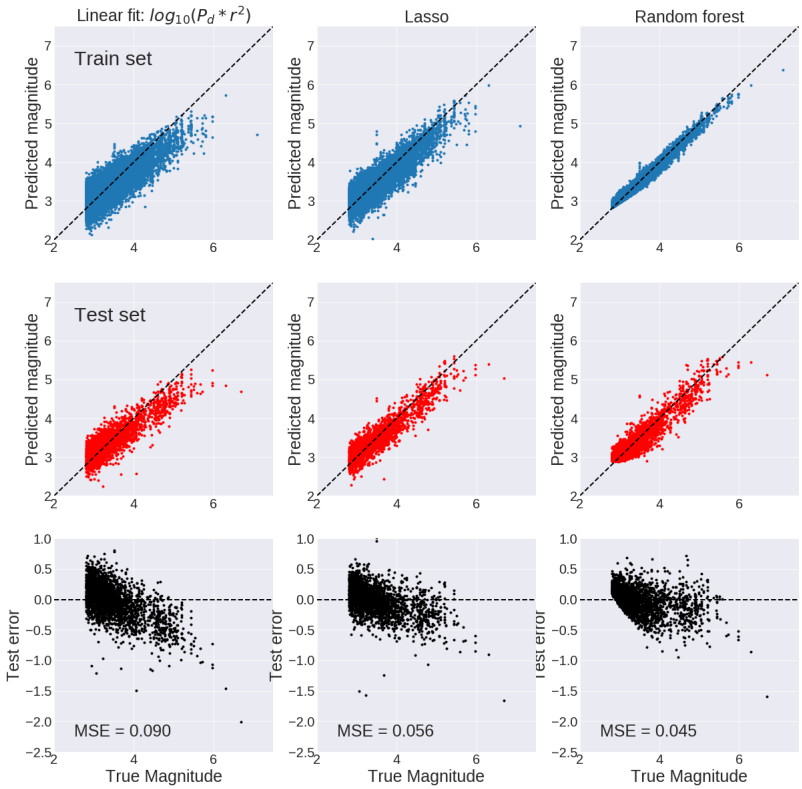

Figure 5: **Predictions by the baseline model, lasso regression, and random forest.** The three rows show the train set, the test set, and the test error respectively. The dashed lines show zero error.

Figure 5 shows the predicted magnitudes and error compared to the true magnitudes. As mentioned in Section 5.1, the baseline model systematically underestimates the magnitudes of large earth-

quakes. For lasso and random forest models, this effect is less severe. However, it seems that even using the latter two models, we still underestimate the magnitudes of the large earthquakes (geq M4.5), as shown by the mean error (Table 1). The MSE for the small earthquakes is several times less than that of the large earthquakes. At least one of the reasons is that the number of samples within this magnitude range is too small. Another reason could be that there are not many features which are not sensitive to the finite fault effect for large earthquakes, such as reliable frequency-related features.

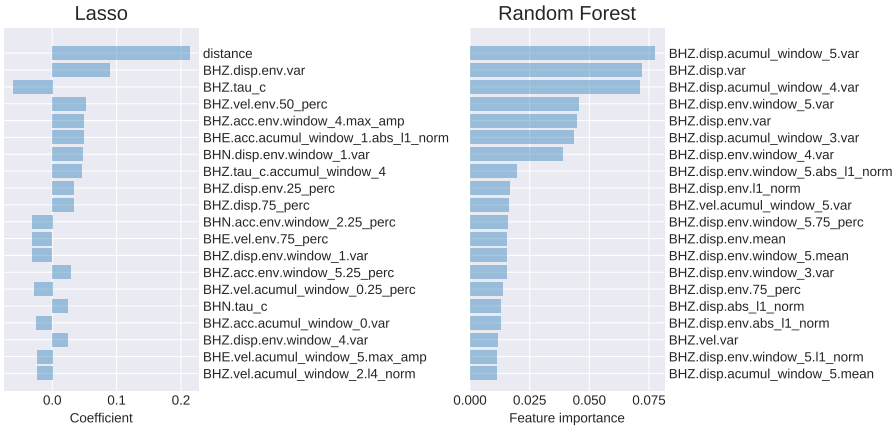

Figure 6: **Top 20 most important features for lasso regression and random forest.** In random forest the importance of distance is the former, although it is also kept as a feature.

5.2.3 Important features

Figure 6 shows the most important features for lasso and random forest. It is expected that distance would be very important for lasso but not for random forest because the amplitude-related features for the former are not corrected for geometrical spreading (distance). It seems that all but one for random forest are derived from the vertical component of the displacement time series (labeled as BHZ.disp), while features for lasso show a greater variety. This seems consistent with that the signal-to-noise ratio for P wave is generally higher for the vertical component, but there could be some non-linearity that may not be fully explained by a linear model such as lasso. In addition, frequency-related features are more important for lasso (e.g. τ_c).

6 Neural Networks and LSTM

Neural networks are famous for its function of "automatic feature extraction". So for the input layer, instead of using empirical features, we tried used several "raw" measurements, including the time series of displacement, velocity, acceleration and cumulative acceleration of the ground motion. As we have mentioned in the linear models, we cropped 3 seconds window starting from the onset, which give us total 60 data points, given the sampling rate of instrument is 20Hz on one component. Since in total we have three components data, plus the distance information, the input layer size is $3 * 60 + 1 = 181$.

Again, considering the symmetrical feature of input, i.e., given the sign reverted ground motion, it ends up with same magnitude of earthquakes, we did one more step, using the envelope transition, rather than the raw waveform, as the features of input layer. To avoid over-fitting, we include the $l2$ regularization and a dropout layer.

6.1 Hyper-parameter tuning using validation dataset

The **hyper-parameter** in Neural networks includes the **number of hidden layers**, the **number of neurons in each hidden layers**, the learning rate, weight decay($l2$ regularization coefficient),

dropout ratio, the activation function and so on. Due to the high cost of training a neural network, here we only focusd on tunning two parameters, the number of hidden layers and the number of neurons in each hidden layers, which affect the representation capability of neural networks the most. For the layout of neural networks, we keep the number of neurons the same in each hidden layer.

For the choice of activation function, we tried using ReLU and Tanh. It turns out that the **ReLU** gives much better performance, from the point view of convergence rate and final loss values. As for other parameters, we set the weight decay to 0.005, the drop ratio to 10%.

Neurons in each layer	10	20	50	100	180	300
hidden layer = 1	0.1098	0.1476	**0.1130**	0.1046	0.1092	0.1159
hidden layer = 2	0.0895	**0.0872**	0.0915	0.0978	0.0899	0.0873
hidden layer = 3	0.0994	0.0950	0.1479	**0.0861**	0.0906	0.0942
hidden layer = 5	0.0883	0.1128	0.0865	0.0947	**0.0862**	0.0990
hidden layer = 10	0.0944	0.0948	0.0935	0.0912	**0.0878**	0.2084

Table 2: **MSE values on the validation data using different number of hidden layers and neurons in each hidden layer, using 25 epoch for training.** The best performance is given by the model (hidden_layers=5, neurons=180), which is underlined in the table.

In order to tune the hyper-paramters, we split the train data one step more, keeping 80% as train data and 20% as validation data. Table2 shows our validation results, using the displacement as the input feature. From the table, we can see that there is really no clear patterns as to set the number of layers and neurons, indicating the complexity and non-linearity of neuron networks. One observation, though, coming from the table, is that setting the number of neurons in each hidden layer to 300, might there is no optimal models coming from that setting. Since our input layer is in dimension of 181, 300 might leads to over-fitting our model. On the other hand, setting the number of neurons to 10 is might also leads to under-fit.

6.2 Long Short-Term Memory(LSTM)

We also tried LSTM here since our input feature is basically time series. However, the MSE of validation and test data, stays around 0.180 and won't reduce any more. We got one model, using the acceleration as input time series, reaches the MSE value as 0.140. However, the MSE value is still way beyond our baseline mode. One difficulty we found in LSTM is there a very important feature, distance, very difficulty to incorporate into the LSTM framework since it doesn't have any temporal relationship with the time series. Unlike the task of speech recognition, in which the shape of waveform is important but the absolute amplitude is not, distance will play a vital role in prediction the magnitude since the amplitude of waveform will decay in exponential order with distance.

6.3 Performance Summary

Input features	MSE	MSE($<$M4.5)	Mean Error($<$M4.5)	MSE($\geq$M4.5)	Mean Error($\geq$M4.5)
displacement	**0. 088**	0.1476	**0.1130**	0.1046	0.1092
velocity	0.104	**0.0872**	0.0915	0.0978	0.0899
acceleration	0.102	0.0950	0.1479	**0.0861**	0.11
cumul acc	0.099	0.0865	0.0947	**0.0862**	0.0990

Table 3: **The MSE values of Neural networks using different input features.** The best performance is given by the model (hidden_layers=5, neurons=180), which is underlined in the table.

From table3, we can see that the overall performance of Neural networks beats the baseline model but still not as good as linear models or random forest.

7 Discussions and Conclusions

Blabla

References

[1] R. M. Allen and H. Kanamori. The potential for earthquake early warning in southern california. *Science*, 300(5620):786–789, 2003.

[2] R.M. Allen, P. Gasparini, O. Kamigaichi, and M. Bose. The status of earthquake early warning around the world: An introductory overview. *Seismo. Res. Lett.*, 80(5):682–693, 2009.

[3] M. Bose, R. Allen, H. Brown, G. Cua, M. Fischer, E. Hauksson, T. Heaton, M. Hellweg, M. Liukis, D. Neuhauser, P. Maechling, and CISN EEW Group. *CISN ShakeAlert: An earthquake early warning demonstration system for California*. Early Warning for Geological DisastersScientific Methods and Current Practice, 2013.

[4] M. Bose, E. Hauksson, K. Solanki, H. Kanamori, Y.-M. Wu, and T. H. Heaton. The status of earthquake early warning around the world: An introductory overview. *Bull. Seismol. Soc. Am.*, 99:897905, 2009.

[5] L. Breiman. Random forests. *Machine learning*, 45(1):5–32.

[6] T. Hastie, R. Tibshirani, and J. Friedman. The elements of statistical learning. *NY Springer*, 2008.

[7] H. Kanamori. Real-time seismology and earthquake damage mitigation. *Annu. Rev. Earth Planet. Sci.*, 33:195–214, 2005.

[8] J.A. Strauss and R. M. Allen. Benefits and costs of earthquake early warning. *Seismo. Res. Lett.*, 87:765–772, 2016.

[9] M. Withers, R. Aster, C. Young, J. Beiriger, M. Harris, S. Moore, and J. Trujillo. The status of earthquake early warning around the world: An introductory overview. *Bull. Seismol. Soc. Am.*, 88(1):95–106, 1998.

[10] G. Wurman, R. M. Allen, and P. Lombard. Toward earthquake early warning in northern california. *Journal of Geophysical Research: Solid Earth*, 112(B8), 2007.