Short-term Earthquake Prediction via Recurrent Neural Network Models

Comparison among vanilla RNN, LSTM and Bi-LSTM

Xiangyu Du*
Responsible Professor: Elvin Isufi†
Supervisors: Mohammad Sabbaqi, Maosheng Yang‡
EEMCS, Delft University of Technology, The Netherlands

January 25, 2022

Abstract

Earthquake prediction has raised many concerns nowadays, due to the massive loss caused by earthquakes, as well as the significance of accurate forecasting. Lots of trials have been investigated and experimented but few achieved satisfying results on short-term prediction (i.e., usually those earthquakes that will happen in three months). It is cardinal to detect strikes within a few minutes or hours in advance. In this paper, given thirty seconds of waveform signal before earthquakes happen, we compare the performances of three different recurrent neural networks, namely vanilla recurrent neural network (RNN), long short-term memory (LSTM) and bidirectional LSTM, on earthquake prediction. We choose recurrent neural networks because their inner structures take advantage of learning the temporal dependencies from time series sequence. Results show that LSTM has better performance predicting on unseen data than the other two networks.

Keywords: Earthquake Prediction, Time Series Model, Recurrent Neural Network, Long Short-term Memory, Bidirectional Long Short-term Memory, Over-fitting, Grid Search

1 Introduction

Earthquake is recognized as one of the most devastating natural disasters on earth, which usually causes massive loss in human world. Thus, it is essential to predict when and where they will take place, although it is a challenging task as their intrinsic nature is random [1].

According to [2], widely used earthquake prediction approaches nowadays can be divided into four categories. The first two approaches were popular with inadequate amount of earthquake data, i.e., 1) mathematical tools like FDL [3] and 2) precursor-based methods which retrieve geographical features such as seismic anomalies [4], cloud image [5] and animal behaviour [6]. Then, machine learning methods came to stage because of more and more

*x.du-1@student.tudelft.nl
†e.isufi-1@tudelft.nl
‡m.sabbaqi@tudelft.nl, m.yang-2@tudelft.nl

earthquakes happening, leading to larger datasets. One type of them called artificial neural network was utilized by Maria Moustra et al. [7] to predict earthquake in 2011. The model achieved accuracy of almost 70 percent, which was not that decent at the moment. From the research it was concluded that less number of attributes in the dataset as well as the class imbalance leaded to the unsatisfying result.

Finally, deep learning approaches have been applied to predict earthquake recently. Mohsen Yousefzadeh et al. [8] evaluated deep neural network (DNN) model that classified magnitudes of earthquakes happening in the next seven days using a newly introduced parameter Fault Density (based on the concept of spatial effect). Result showed that DNN outperformed other machine learning models with test accuracy around 79% on magnitudes higher than 8. Combining characteristics of convolutional neural networks (CNN) and recurrent neural networks, Luana Ruiz et al. [9] introduced a network called Graph Convolutional Recurrent Neural Network (GCRNN) to identify the epicenters of earthquakes happened in two months. GCRNN made use of convolutional filter to keep the number of trained parameters independent of the time sequences taken as input. With considerably less parameters taken into consideration, GCRNN achieved accuracy around 33% for both 30-seconds wave and 60-seconds wave. Furthermore, a variant of recurrent neural network called long short-term memory (LSTM) was applied by Q.Wang et al. [2] to learn the temporal-spatial dependencies of earthquake signal, and thus to predict earthquake. The accuracy was around 75%, although they took the earthquake data within one year as input. In addition to that, Parisa Kavianpour et al. [10] proposed a CNN-LSTM combined network to also learn spatial-temporal dependencies of earthquake signal. They evaluated the model performance in 9 areas of mainland of China with number of all earthquakes happening in one month between 1966 and 2021 as input. They also took other earthquake features into consideration such as latitude and longitude. They compared the performances of several models such as MLP and SVM and CNN-LSTM achieved the highest score.

Most of the experiments conducted by scientists and researchers in this field considered to predict earthquakes in a long run (earthquakes that will happen several months or even several years later). However, our task addressed in this paper expects to identify the earthquake that will happen in 30 seconds. Regardless of probably inaccurate result of long-term prediction (not accurate to a single day), 30 seconds gives people time to react and thus, avoid unimaginable outcome. The research question is *How do individual time series model compare with each other (vanilla RNN, LSTM, Bi-LSTM)?* Specifically, we will evaluate the performance of each model on earthquake prediction using various evaluation metrics and make comparisons among them.

2 Methodology

In this section, we first introduce the three deep neural networks and motivate why they are suitable for earthquake prediction task. Then metrics used to evaluate the performances of the networks will be explained in detail.

2.1 Deep Learning Approach

We introduce the three types of recurrent neural networks and present their strengths respectively for easy comparison.

2.1.1 Recurrent Neural Network - vanilla RNN

Based on David Rumelhart's work in 1986 [12], recurrent neural network (RNN) was invented to be capable of learning temporal dynamic behaviour. It reads data in a sequential manner (one at a time) and its internal architecture ensures that the prior input is considered when computing the output of the current input [13]. To be specific, at each step, calculations are done in an RNN cell to produce the output which is also known as hidden state. The output is then transmitted to the next cell and combined with the next input in the sequence to generate the next output, as shown in Figure 1. In short, RNN is able to 'remember' the information retrieved from the previous step, and thus learn the sequential features to predict on unseen data. And that explains why RNN is suitable for earthquake prediction task. With such chain type architecture, RNN can deal with problems such as handwriting recognition which converts the sequential information generated when writing on the handwriting device into text [15] and speech recognition [16].

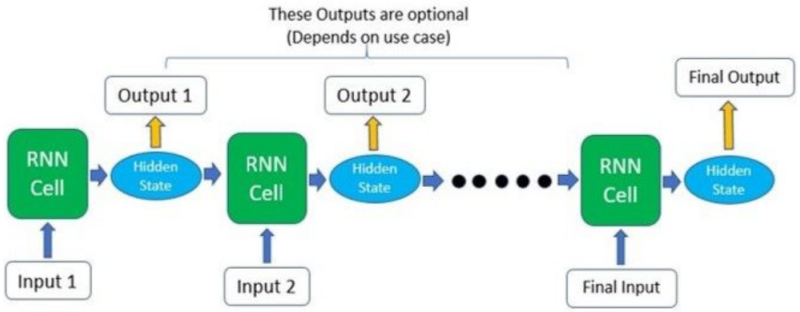


Figure 1: RNN architecture. Output of the cell will be joined with input of next cell to generate next output [14]

On the other hand, however, traditional RNN suffers from a drawback called gradient vanishing/exploding with which the gradient values shrink or blow up during the back-propagation process, which either prevents the weights from updating or causes the model to be unstable [17]. Intuitively, vanilla RNN gradually 'lose' information of the data it reads in previously. That is why vanilla RNN has an unsatisfying performance on carrying information from long sequences [18].

2.1.2 Long short-term Memory

To overwhelm the shortcoming of vanilla RNN above-mentioned, long short-term memory (LSTM) was introduced. As the name suggests, it was designed to hold on to long-term dependencies. LSTM is a variant of traditional RNN so it is equipped with the functionalities of vanilla RNN. Its gating mechanism is what sets it apart. Its input gate, forget gate and output gate together filter out useless information and maintain the gradient values during training process, and then back-propagate them through layers so that the network can retain more data for further processing [13]. To be precise, apart from short-term memory, the forget gate is added to carry long-term memory, as highlighted in Figure 2.

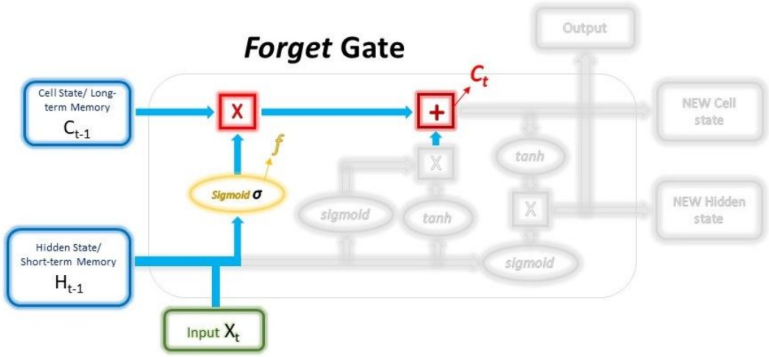


Figure 2: Inner structure of a LSTM cell. Forget gate mechanism is highlighted. [19]

2.1.3 Bidirectional long short-term memory

Bidirectional long short-term memory (Bi-LSTM) is an extension of LSTM consisting of two LSTMs which process data forward and backward, respectively, allowing the network to preserve the information both from the past and future. The network structure of Bi-LSTM is depicted in Figure 3. Benefiting from the architecture, Bi-LSTM is extremely powerful in the aspect of context classification and prediction, language translation [20] for example.

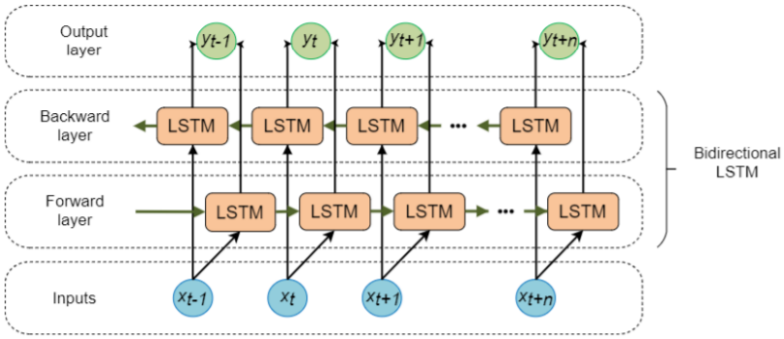


Figure 3: Architecture of Bi-LSTM. [23]

2.2 Performance Evaluation Metrics

There are only two possible outcomes of the networks we introduced above: 0 standing for the prediction that an earthquake will not happen and 1 for the prediction that an earthquake is expected to happen. For such binary classifier, we primarily choose confusion matrix as one of the evaluation metrics [21]. The real meaning of each matrix component in earthquake prediction task is listed.

- True Positives (TP): The number of times the model correctly predicts that an earthquake will happen in the next 30 seconds.
- True Negatives (TN): The number of times the model correctly predicts that an earthquake will not happen in the next 30 seconds.

- False Positives (FP): The number of times the model wrongly predicts that an earthquake will happen in the next 30 seconds.
- False Negatives (FN): The number of times the model wrongly predicts that an earthquake will not happen in the next 30 seconds.

We claim that FN raises the most serious concerns for the reason that an earthquake will cause massive loss if it is predicted to not happen. Meanwhile, we also pay attention to FP. It can bother the society with a evacuation drill if the earthquake eventually doesn't happen but is predicted to happen. Therefore, among the meaningful evaluation metrics derived from the confusion matrix, positive predictive value related to FP (PPV, also known as precision) and true positive rate related to FN (TPR, also known as sensitivity or recall) are used in our case. They are defined as follows.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Interpreted from the formulas, precision means the number of earthquakes that do happen among all that are predicted to happen. Recall means the number of earthquakes that are predicted to happen among all that do happen. Recall rate is penalized whenever a FN appears. Similarly, precision rate is penalized whenever a FP appears. Thus, to reduce the penalty caused by FN and FP, recall and precision are expected to be as high as possible.

To balance the effects precision and recall have on the evaluation, another metric named f1-score is used. It is defined as the harmonic mean [22] of the precision and recall:

$$\text{F1 score} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}}$$

Finally, test accuracy is included in the metrics, to directly show the overall performance of the model on unseen data, given that the dataset is relatively balanced. It is defined as the percentage of correct predictions for the test data:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

3 Implementation

In this section, we first give details on the procedures taken to process the raw earthquake data that happened in New Zealand from 1st January 2016 to 31st December 2020 until obtaining a balanced dataset for training, testing and validation. All the data can be found on the website of the International Federation of Digital Seismograph Networks (FDSN) [24]. Next, Details on the architecture design of three deep neural networks mentioned in Section 2.1 are given. Finally, we present the over-fitting problem encountered during the implementation and how such issue was solved.

3.1 Data Preprocessing

In this section, we explain the steps taken to build the earthquake dataset. Section 3.1.1 describes the captured area and active stations that detected earthquakes. Section 3.1.2

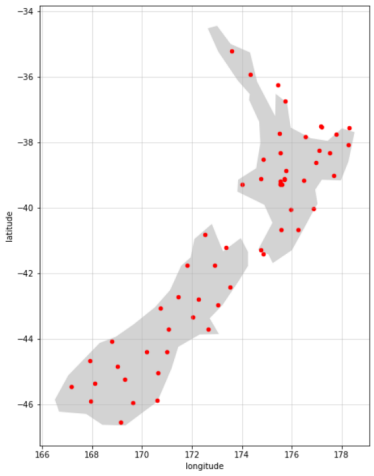
explicates how the earthquakes data is downloaded and filtered to obtain a balanced dataset. Next, Section 3.1.3 gives information about processing seismic waveform and lastly, final processed dataset is obtained after filtering stations based on a certain criteria in Section 3.1.4

3.1.1 Area of Interest and Stations

First of all, we define a bounding box[1] to figure a geographical area in which the earthquakes happened. A bounding box restrains an area determined by four coordinates (two longitudes and two latitudes in our case). The area defined gives geographical outline of mainland of New Zealand. Then we choose 58 stations[2] within the area. The map of New Zealand and geographical distribution of stations can be found in Figure 4.

Coordinate type	Value
Minimum Longitude	166.104
Maximum Longitude	178.990
Minimum Latitude	-47.749
Maximum Longitude	-33.779

(a) Coordinates specifying the Bounding Box for the New Zealand earthquake dataset



(b) Geographical distribution of the selected 58 stations

Figure 4: Area and stations defined by the Bounding Box

3.1.2 Earthquake Filtering

Next we request information of all earthquakes that happened in the Bounding Box in the mentioned time period[3]. For each earthquake, the available information makes up of time, latitude, longitude, magnitude and depth measurements. After filtering out the data without any of these characteristics, we obtain 122465 earthquakes, as depicted in Figure 5.

Furthermore, we plot the distribution of magnitude and depth of the earthquakes filtered, in Figure 6a and 6b, respectively. As can be seen from the charts, majority earthquakes happened with magnitude around 2 and with depth less than 200 kilometers. Thus, to leave out those earthquakes at the tail that might cause the dataset to be imbalanced and to make sure the earthquakes are comparable between each other, we filter earthquakes with

[1] https://wiki.openstreetmap.org/wiki/Bounding_Box

[2] We use 'Station Service' [24] to obtain information related to stations. Note that we only consider stations that were active after 2016.

[3] information related to earthquakes can be downloaded with 'Event Service' [24].

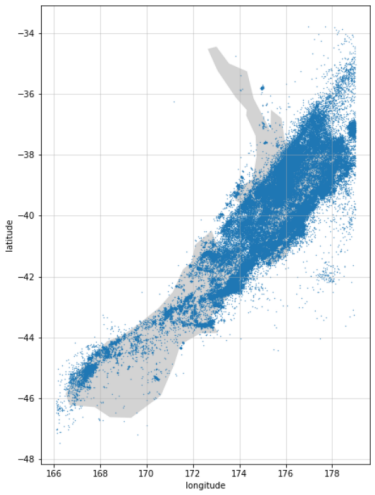
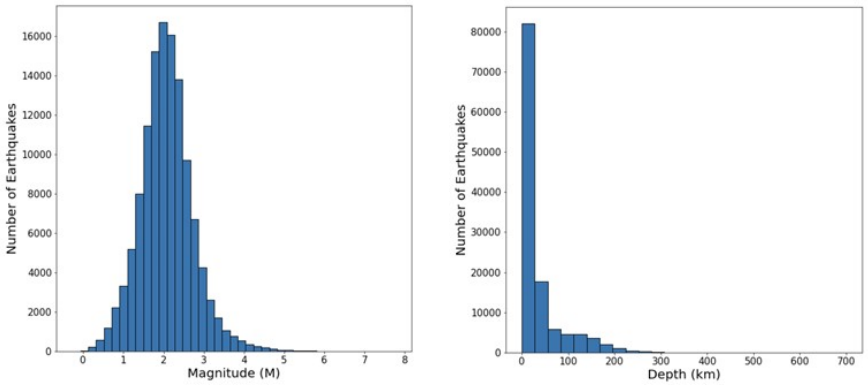


Figure 5: All earthquakes defined by Bounding Box in given time period

magnitude between 1 and 3, and with depth less than 200 kilometers. This step leaves approximately 100.000 earthquakes.



(a) Distribution of magnitude (b) Distribution of depth

Figure 6: Histograms of magnitude and depth for the gathered earthquakes

3.1.3 Seismic Waveform

We then investigate the seismic waveform [25] of the earthquakes. For each of them, we retrieve the waveform time series of 30 seconds before the earthquake taking place, measured along vertical axis and recorded of 100 HZ, i.e. 100 samples per second. Figure 7 shows the signal detected by three random stations. Next, we normalize the signals to make sure that they are in the same range for all stations, resulted in Figure 8. Furthermore, data with high frequency usually contains noise that can be transmitted to the network, which can negatively influence the result [26] and data with low frequency has risk of lacking

useful data features. Considering the time and device quality constraint, as well as model performances, we choose to downsample[4] the waveform to 50 HZ, as shown in Figure 9.

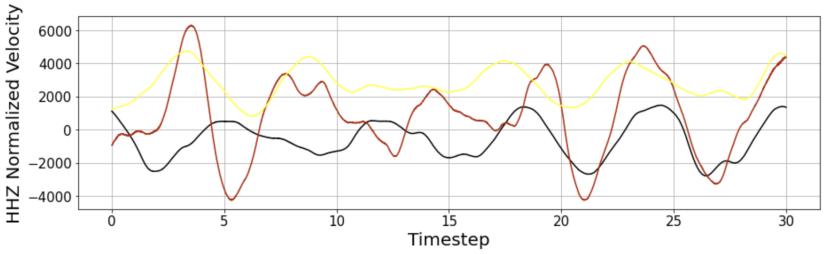


Figure 7: Original seismic signal for three random earthquakes

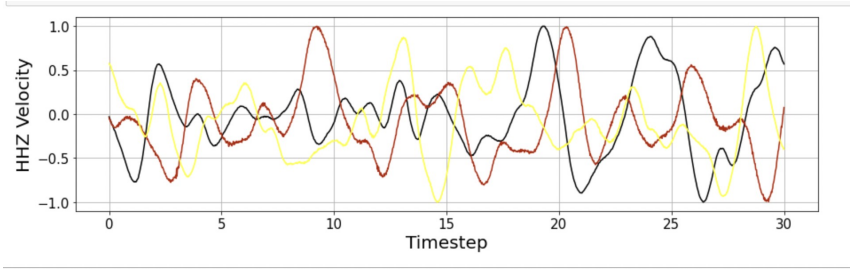


Figure 8: Normalized seismic signal for three random earthquakes

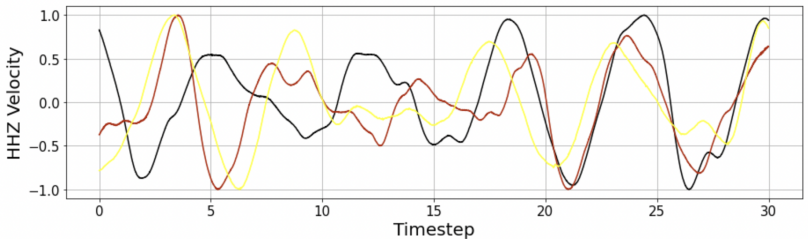


Figure 9: Downsampled seismic signal for three random earthquakes

To obtain normal waveform data, an algorithm is designed to find the maximum interval between the timestamps that two earthquakes happened. Intervals are sorted in descending order and starting from the biggest interval, a range of waveform in the middle is selected (we consider them as calm moments). If the timestamp of the waveform is not in the earthquake data list, then the waveform is classified as a normal waveform signal. This process results in approximately 160k normal signals.

3.1.4 Station Filtering

The number of pieces of earthquake data and normal waveform data varies among the selected 58 stations. For each station, we plot the number of earthquake signals and normal

[4]https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.resample.html

waveform signals, as shown in Figure 10. The way we assign earthquake signal and normal waveform signal is described in Section 3.3.

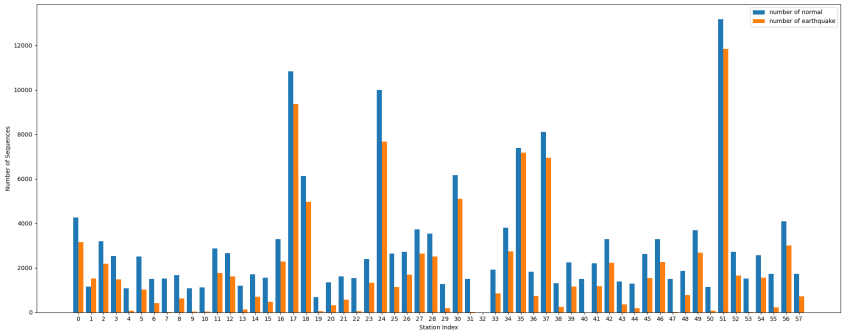


Figure 10: Number of earthquake signals and normal waveform signals assigned to each of 58 stations

As can be seen from the figure, the number of earthquake signals and normal waveform signals for some stations witnesses a huge disparity. The difference can lead to an extremely high training accuracy after even 1 epoch, which is not useful in the final evaluation. Thus, to reduce such bias, we set a threshold: those stations with number of normal waveform signals 2 times larger than the number of earthquake signals are discarded. This leaves in total 27 stations, shown in Figure 11.

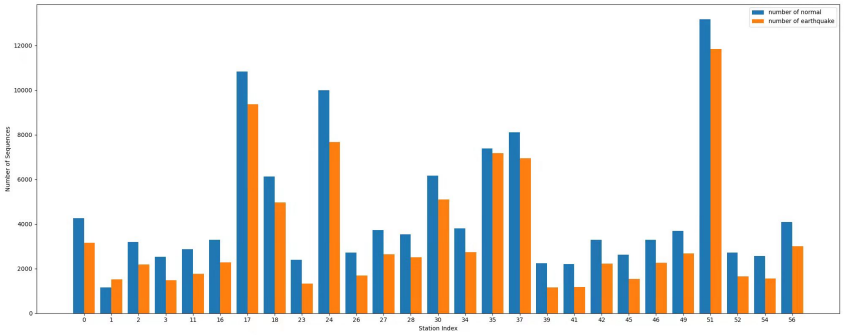


Figure 11: Stations index and number of earthquake signals and normal signals after filtering

3.2 Network Architecture Design

Once the dataset is well preprocessed, architecture for the networks should be designed. We first illustrate the architecture design of vanilla RNN, LSTM and Bi-LSTM for which a group of fixed parameters are given under the condition of cross validation (will be introduced in Section 3.4.3). Note that for this section, we only give the set of fixed parameters. Ranges of network parameters will be provided when grid search on hyperparameters comes to stage in Section 3.4.4. Then we elaborate the over-fitting problem during training phase and several trials to solve the issue. It is worth mentioning that network structures are designed to be simple in order to get free from long process time taken by cross validation and grid search.

3.2.1 Vanilla RNN Architecture

The network structure of vanilla RNN is designed as follows. An RNN layer with 128 neurons first reads in the input data, one sample at a time out of 1500 (30 seconds of waveform, each second has 50 samples). The last two dimensions of the input data (length of the whole sequence 'length' and dimension of input 'H_in') are flattened to one-dimensional tensor shown as length * H_in. A dropout layer with dropout rate 0.2 is then connected to ensure that output of activation function of random 1/5 neurons are set to 0. Since the earthquake prediction is a binary classification task, the connected linear layer (also known as dense layer or fully connected layer) transforms the output vector with dimension length * h_out to a number with the same dimension as that of the label (0 and 1 in our case). Finally, the number is squeezed by Sigmoid activation function into a probability between 0 and 1 and taken as output.

3.2.2 LSTM Architecture & Bi-LSTM Architecture

The structure of LSTM is designed to be highly similar to that of vanilla RNN. The difference is that LSTM defines a LSTM layer with 128 neurons instead. The architecture for Bi-LSTM is different by having two bidirectional LSTM layers. The input size of the first LSTM layer and hidden size share the same with vanilla RNN, while the input size for the second layer is two times that of hidden size. It is because the input of second layer needs to carry information both for forward propagation and backward propagation.

3.3 Experimental settings

For each earthquake and normal signal, we first assign it to the closest station by calculating great-circle distance [27]. The input data for training consists of 30 seconds of waveform signal with 50 samples per second assigned to each of the 58 stations. The models are trained station-wise, with network input size of 1 at a time, representing 1 sample. The output of the network is the probability that the input sequence is classified as an earthquake. The likelihoods below 0.5 are truncated to 0, meaning normal behaviour and the ones larger than or equal to 0.5 are regarded as 1, representing earthquake behaviour. To use cross validation (will be mentioned in Section 3.4.3), 10% of the dataset is used for testing set. The other 90% is equally divided into 5 subsets of which 1 is for validation and the other 4 is used to train the model during each iteration. The number of epoch is decided by Early Stop technique [28], which is another way to prevent over-fitting (will be mentioned in Section 3.4). We force the training process to stop if the validation loss doesn't decrease for consecutive 7 epochs. Aiming at reduce the effect caused by over-fitting, we average the results over 5 iterations. Details on each parameter settings are given in the following section.

3.4 Over-fitting & Solution

Over-fitting points to a problem that the model fits closely or even perfectly to a specific dataset which is usually the training set in machine learning tasks, but may therefore fail to perform well on unseen data which is usually the test set [29]. More generally, the model may be negatively affected by the useless information in training set so that it is not able to learn the data features very well. We thus propose regularization (dropout included), cross validation and grid search on hyperparameters to solve the issue.

3.4.1 Regularization

Regularization is a commonly used approach that adds a penalty term to the loss function on training set to prevent over-fitting problem [30]. In cases that the data possesses a large number of useful features which are not expected to thrown away, regularization is utilized to reduce the weight of those features, and therefore reduce the complexity of the learned model. L_2 regularization (also known as weight decay) is applied to our models and ranges for trials are included in Section 3.4.4.

3.4.2 Dropout

In 2014, Nitish et al. [31] introduced an approach called dropout aiming to solve over-fitting problem. According to the paper, dropout is a technique that during the training process of the model, random neurons are temporarily discarded from the network according to a certain probability (set the output of activation function of such neurons to 0), which is equivalent to finding a thinner network from the original network. Ranges for trials on our models are included in Section 3.4.4.

3.4.3 Cross validation

Cross validation was introduced to let the model learn as many data patterns as possible, and reduce the noise that probably leads to high bias and variance [32, 33]. For our case, a variant called k-fold cross validation is used. As the name suggests, k-fold cross validation evenly splits the dataset into k subsets, each representing one fold. Then the model will be trained for k iterations. During each iteration, one out of the k subset is used for validation and the other k - 1 subsets are used as training set. Finally, the test error is usually averaged over k trials to get overall performance of the model [34]. For the earthquake prediction task, we set $k = 5$ due to time limit. 10% of the dataset was used for test set and the other 90% was evenly divided into 5 folds.

3.4.4 Grid Search on Hyperparameters

Another approach proposed to optimize the hyperparameters settings is called Grid Search [35]. Grid search is a technique that exhaustively searches and evaluates every possible combination of a set of parameters within a certain range. It is used to optimize the model performance with the help of k-fold cross validation and reduce the effect has by over-fitting. For the task we make use of a powerful toolkit called NNI[5] to automate hyperparameter tuning. The grid of hyperparameters optimized with grid search for all three models is depicted in Figure 12.

Hyperparameters	Parameter Grid
Dropout Rate	[0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5]
Regularization Parameter (Weight Decay)	[0.002, 0.004, 0.006, 0.008, 0.01]
Learning Rate	[0.00005, 0.0001, 0.00015, 0.0002]

Figure 12: Range of hyperparameters for grid search

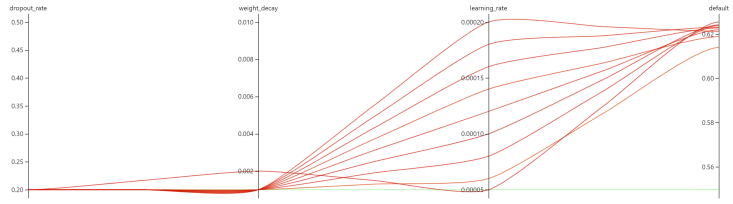
[5] https://nni.readthedocs.io/en/stable/

4 Results

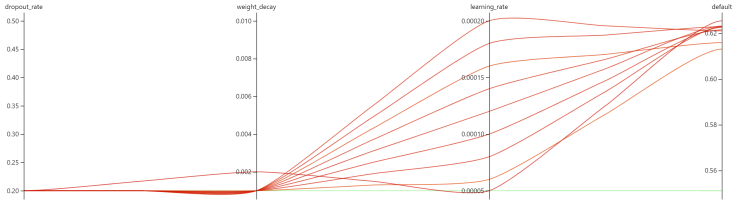
As explained in Section 3.1.4, the last step of data preprocessing leaves 27 stations in total for later experiment. The first trial with a group of fixed parameters reveals that 5 of 27 stations have over-fitting problem. Therefore we directly apply grid search on each of them with each network model. We present the performance before (shown in Figure 13) and after (shown in Figure 14) doing grid search for one certain station. Results for other four stations are in Appendix A.1.

	Accuracy	Precision	Recall	F1-score	FN	FP	TN	TP
vanilla RNN	0.553874	0.449956	0.553874	0.426538	593	26	469.2	21.8
LSTM	0.55009	0.464556	0.55009	0.430808	579.6	39.4	460	31
Bi-LSTM	0.556036	0.516802	0.556036	0.434852	584.8	34.2	458.6	32.4

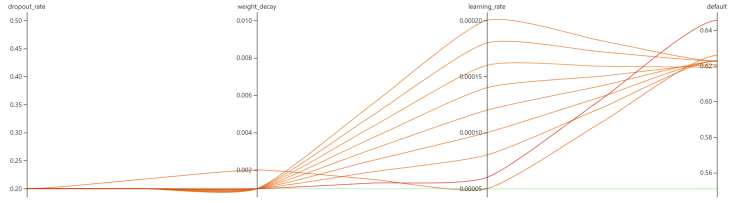
Figure 13: Performance of station 18 before Grid Search



(a) Grid Search result of station index 18 for vanilla RNN



(b) Grid Search result of station index 18 for LSTM



(c) Grid Search result of station index 18 for Bi-LSTM

Figure 14: Grid Search result of station index 18 with three models

To evaluate the performance of each model on the other 22 stations, we choose Boxplot to display the distribution of scores for each evaluation metric. Boxplot of each metric for each model is depicted in Figure 15. Distribution of each evaluation metric of each model for each station over 5-fold cross validation can be found in Appendix A.2.

Finally, regarding the accuracy of each model, by averaging the number of each evaluation metric over 22 stations and 5-fold cross validation, we obtain the overall performance of each

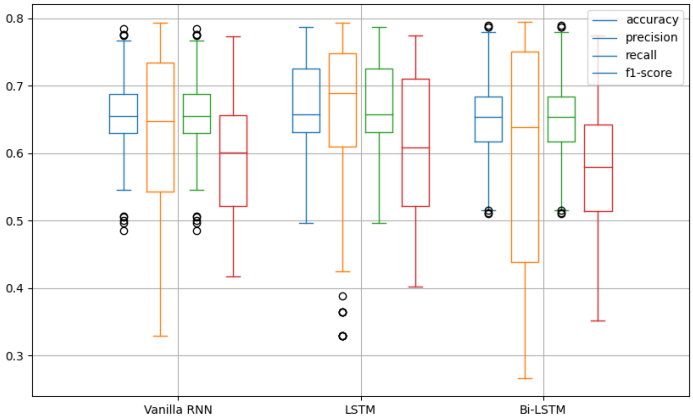


Figure 15: Boxplot of evaluation metrics over 22 stations for each model (for each model, values from left to right is accuracy, precision, recall, f1-score)

model, shown in Figure 16.

	Accuracy	Precision	Recall	F1-score	FN	FP	TN	TP
vanilla RNN	0.655875	0.622033	0.655875	0.594271	296.8182	58.09091	155.8182	82.13636
LSTM	0.664583	0.646016	0.664583	0.604485	298.9364	55.97273	153.0818	84.87273
Bi-LSTM	0.656961	0.606048	0.656961	0.582474	313.6455	41.26364	170.6091	67.34545

Figure 16: Averaged result of each evaluation metric for three models

5 Responsible Research

First and foremost, one of the typical ethic concerns related to machine learning tasks is that, different from problems that can be explained or resolved with a well-designed algorithm, machine learning tasks usually cannot guarantee a satisfying result for a variety of reasons. In our case for instance, it is hard to find a balanced point between data that has much noise and data that loses useful properties during preprocessing phase. We chose to use the data with frequency of 50 HZ. However, the optimal result may be obtained with other frequencies and it is infeasible to brute-force all possibilities because of time limit of the research. In addition, changes in each network layer during training process are difficult to track comparing with supervising changes in a formula or an algorithm. Thus, a machine can still make wrong prediction on classification tasks even with a well-trained model, which should be taken cautiously when solving problems with machine learning approaches.

Besides, experiments during the research were designed to be reproducible and repeatable. To be specific, data retrieval and preprocessing mentioned in Section 3.1 are well documented in notebook. Content of each step is integrated in single block with detailed and clear comments. Architecture design for each network in Section 3.2 and experiment settings in Section 3.3, as well as hyperparameter optimization are explained thoroughly so

that any researcher possessed with basic knowledge on deep neural network is able to reproduce the program. Furthermore, restrained by the time given and device quality, results were run over a large dataset multiple times with help of cross validation and grid search on hyperparameters. By doing so, adverse effects had by bias and variance of the dataset are mitigated. Other than that, we guarantee that all the data and results were made without manipulation, fabrication or trimming. Finally, it should be pointed out that the source code is proprietary but can be provided upon request.

Regarding the usage of raw data, we clarify that all the data are made open-sourced and free of charge. The website and related projects are sponsored by New Zealand Government with help of its agencies.

6 Discussion

In this section, we first interpret the results presented in Section 4 and evaluate performances of all three models, followed by what can be improved for future work.

6.1 Evaluation

Firstly, results in Section 4 and Appendix A.1 reveal the fact that grid search does improve the overall prediction accuracy on some stations, for example, from around 55% to over 62%, but not all. It might be because parameter ranges are set to be relatively small. Thus, over-fitting still exists but it is no doubt that model performance is increased by grid search.

Then, as can be seen from Figure 15, LSTM model tends to outperform other two in all these four metrics. Most of the values lie in higher positions with respect to accuracy and recall, which means that it correctly predicts the most earthquakes among all that happen. But some outliers in precision show that among its predictions that earthquakes will happen, some of them have very low probability to actually happen. Comparing with LSTM, outliers in accuracy and recall of vanilla RNN and Bi-LSTM illustrate that their predictions are not very stable and they are prone to more mistakes.

When it comes to the overall numerical results in Figure 16, accuracy indicates that LSTM performs the best out of three models. Bi-LSTM has the highest TN score and the lowest TP score, showing that it is better at predicting normal behavior than at predicting earthquakes.

Furthermore, all three models have higher accuracy under the condition that label is false (the ratio of FN to FP), meaning that they are all better at correctly classifying normal signals. But for earthquake signals, they tend to wrongly classify them as normal signals.

To summarise, LSTM have the best performance in our earthquake prediction task, probably because the gating mechanism guarantees its power at carrying information in long sequential data. On the other hand, RNN still suffers from loss of information from long sequence. For Bi-LSTM, it performs better than RNN and we claim that architecture design of two layers might cause it to learn more data features with noise. In all, we human beings are more desired to be sensitive at earthquakes so that we can take action to avoid loss rather than having satisfying result on classifying normal behavior. Therefore it is still a challenging task to correctly predict earthquakes.

6.2 Future Work

Time limit and device quality restraint obstruct some ideas and trials. First, regardless of the over-fitting presented by a few stations, it can be observed from the other few stations that training accuracy and validation accuracy fluctuated within a range during training and validation process. We claim that it is probably due to the simple network architecture design. So we can increase the number of layers for each network so that they are able to learn more sequential characteristics from the data [36].

Regarding to dealing with over-fitting, other approaches can be experimented, batch normalization [37] for instance. Apart from a variety of approaches to try, the range of hyperparameters can be extended. Particularly, weight decay can have more options to choose from. Also a range of hidden size of the network can be included.

What's more, grid search on all 27 stations can be deployed to find common hyperparameters.

At the same time, data with different frequencies is worth trying in order to find a balance point between data with many characteristics and data with few characteristics.

Finally, the result of this research demonstrated the fact that earthquake prediction is still a challenging problem. Thus, different types of deep learning methods can be used, either alone or combined [38] so that people can learn what can be the optimal solution to such time series forecasting tasks.

7 Conclusions

This paper raised a question on how do individual time series models, namely vanilla RNN, LSTM and Bi-LSTM compare on short-term earthquake prediction (30 seconds before earthquake). To answer the question, 30 seconds waveform signals of earthquakes happening between 2016 and 2020 in mainland New Zealand were retrieved and preprocessed together with generated normal behaviour signals as dataset. We chose to assign detected earthquakes and normal signals to the closest station. The networks were designed and trained station-wise with 30 seconds data as input. Training and validation process revealed that some stations had over-fitting problem. Several approaches such as regularization and grid search were applied trying to solve the issue. Regardless of the fact that the issue remains because of research time limit, the performances of the models have been improved. The final results showed that all three models perform better at correctly classifying normal behaviour signals rather than earthquake signals. Among the three models, LSTM has the best performance, with an averaged accuracy around 66.5% on predicting with unseen data, while Bi-LSTM tends to be negatively affected by data with noise. More approaches such as deepening the networks and training with data of different frequencies can be carried out in the future to improve the accuracy.

References

[1] T. Bhandarkar et al. "Earthquake trend prediction using long short-term memory RNN". In: International Journal of Electrical and Computer Engineering 9.2 (2019), p. 1304, 2019.

[2] Q. Wang et al. "Earthquake prediction based on spatio-temporal data mining: an LSTM network approach". In: IEEE Transactions on Emerging Topics in Computing, 2017.

[3] A.Boucouvalas, M.Gkasios, N.Tselikas, and G.Drakatos. "Modified fibonacci-dual-lucas method for earthquake prediction". In Third International Conference on Remote Sensing and Geoinformation of the Environment, pages 95351A-95351A. International Society for Optics and Photonics, 2015.

[4] Chouliaras, G. "Seismicity anomalies prior to 8 June 2008 earthquake in Western Greece". Nat. Hazards Earth Syst. Sci., 9 (2): 327-335, doi:10.5194/nhess-9-327-2009, 2009.

[5] J. Fan, Z. Chen, L. Yan, J. Gong, and D. Wang. "Research on earthquake prediction from infrared cloud images". In Ninth International Symposium on Multispectral Image Processing and Pattern Recognition (MIPPR2015), pages 98150E-98150E. International Society for Optics and Photonics, 2015.

[6] M. Hayakawa, H. Yamauchi, N. Ohtani, M. Ohta, S. Tosa, T. Asano, A. Schekotov, J. Izutsu, S. M. Potirakis, and K. Eftaxias. "On the precursory abnormal animal behavior and electromagnetic effects for the kobe earthquake (m 6) on april 12, 2013". Open Journal of Earthquake Research, 5(03):165, 2016.

[7] M. Moustra, M. Avraamides, and C. Christodoulou. "Artificial neural networks for earthquake prediction using time series magnitude data or seismic electric signals". Expert systems with applications, 38(12):15032-15039, 2011.

[8] Mohsen Yousefzadeh, Seyyed Ahmad Hosseini, Mahdi Farnaghi. "Spatiotemporally explicit earthquake prediction using deep neural network". Soil Dynamics and Earthquake Engineering, Volume 144, https://doi.org/10.1016/j.soildyn.2021.106663, 2021.

[9] Luana Ruiz, Fernando Gama and Alejandro Ribeiro. "Gated Graph Convolutional Recurrent Neural Networks". arXiv:1903.01888, 2019.

[10] Parisa Kavianpour, Mohammadreza Kavianpour, Ehsan Jahani, Amin Ramezani. "A CNN-BiLSTM Model with Attention Mechanism for Earthquake Prediction". arXiv:2112.13444, 2021.

[11] G. Mazzola. "Graph-Time Convolutional Neural Networks". MSc thesis, TU Delft, 2020.

[12] Williams, Ronald J.; Hinton, Geoffrey E.; Rumelhart, David E. . "Learning representations by back-propagating errors". Nature. 323(6088):533-536, October 1986.

[13] Petneházi, Gábor. "Recurrent neural networks for time series forecasting". arXiv:1901.00069, 2019-01-01.

[14] https://blog.floydhub.com/a-beginners-guide-on-recurrent-neural-networks-with-pytorch/. Accessed: 2022-01-19.

[15] Graves, A., Schmidhuber, J.. "Offline handwriting recognition with multidimensional recurrent neural networks. Advances in neural information processing systems", 21, 545-552, 2008.

[16] Graves, Alex; Mohamed, Abdel-rahman; Hinton, Geoffrey E.. "Speech Recognition with Deep Recurrent Neural Networks". Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on: 6645-6649, 2013.

[17] Manaswi N.K. "RNN and LSTM". In: Deep Learning with Applications Using Python. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-3516-4_9, 2018.

[18] Schmidhuber, Jürgen; Gers, Felix A.; Eck, Douglas. "Learning nonregular languages: A comparison of simple recurrent networks and LSTM". Neural Computation. 14 (9):2039-2041, 2002.

[19] https://blog.floydhub.com/long-short-term-memory-from-zero-to-hero-with-pytorch/. Accessed: 2022-01-19.

[20] Sundermeyer, Martin, et al. "Translation modeling with bidirectional recurrent neural networks.". Proceedings of the Conference on Empirical Methods on Natural Language Processing, October 2014.

[21] Shaikh, S.A.. "Measures Derived from a 2 x 2 Table for an Accuracy of a Diagnostic Test". Journal of biometrics biostatistics, 2, pp. 1-4, 2011.

[22] Wirth F. Ferger. "The Nature and Use of the Harmonic Mean". Journal of the American Statistical Association, 26:173, 36-40, 1931.

[23] Kulevome, D. K., Wang, H., Wang, X.. "A Bidirectional LSTM-Based Prognostication of Electrolytic Capacitor". Progress In Electromagnetics Research C, 109, 139-152, 2021.

[24] FDSN webservice for New Zealand. https://www.geonet.org.nz/data/tools/FDSN. Accessed: 2021-11-15.

[25] Weik M.H.. "seismic wave". In: Computer Science and Communications Dictionary. Springer, Boston, MA. https://doi.org/10.1007/1-4020-0613-6_16842, 2000.

[26] Shivani Gupta et al. "Dealing with Noise Problem in Machine Learning Data-sets: A Systematic Review". Procedia Computer Science 161 (2019) 466-474, 2019.

[27] R. Bullock. "Great circle distances and bearings between two locations". In: MDT, June 5, 2007.

[28] Yao, Yuan, Lorenzo Rosasco, Andrea Caponnetto. "On Early Stopping in Gradient Descent Learning". Constructive Approximation. 26 (2): 289â315, 1st August, 2007.

[29] Chicco, D. "Ten quick tips for machine learning in computational biology". BioData Mining 10, 35. https://doi.org/10.1186/s13040-017-0155-3, 2017.

[30] Stephen Merity, Nitish Shirish Keskar, and Richard Socher. "Regularizing and optimizing lstm language models". arXiv:1708.02182, 2017.

[31] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", 15(56):1929-1958, 2014.

[32] K. Taunk. "A Brief Review of Nearest Neighbor Algorithm for Learning and Classification". 2019 Int. Conf.Intell. Comput. Control Syst., no. Iciccs, pp. 1255â1260, 2019.

[33] G. Guo, H. Wang, D. A. Bell, and Y. Bi. "KNN Model-Based Approach in Classification KNN Model-Based Approach in Classification". On the move to meaningful Internet Systems Intl conference, 2003.

[34] Hussin Ragb, Redha Ali, Elforjani Jera, Nagi Buaossa. "Convolutional neural network based on transfer learning for breast cancer screening". arXiv:2112.11629, pp. 3-4, 2021.

[35] Shashank Shekhar, Adesh Bansode, Asif Salim. "A Comparative study of Hyper-Parameter Optimization Tools". arXiv:2201.06433, p. 2, 2022.

[36] Yoav Levine, Or Sharir, Amnon Shashua. "Benefits of Depth for Long-Term Memory of Recurrent Networks". https://openreview.net/forum?id=HJ3d2Ax0-, 2018.

[37] Ioffe, Sergey; Szegedy, Christian. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". arXiv:1502.03167, 2015.

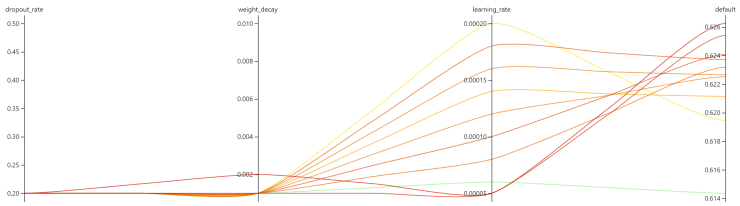
[38] Indu Kant Deo, Rajeev Jaiman. "Learning Wave Propagation with Attention-Based Convolutional Recurrent Autoencoder Net". arXiv:2201.06628, 2022.

A Appendix

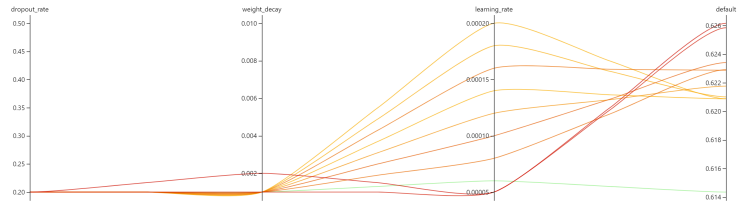
A.1 Results of Grid Search on Station 17, 24, 30 and 51

	Accuracy	Precision	Recall	F1-score	FN	FP	TN	TP
vanilla RNN	0.623541	0.639815	0.623541	0.596364	928	160	601.2	332.8
LSTM	0.682097	0.682139	0.682097	0.679198	828.4	259.6	383.2	550.8
Bi-LSTM	0.697923	0.698825	0.697923	0.694854	849	239	371.8	562.2

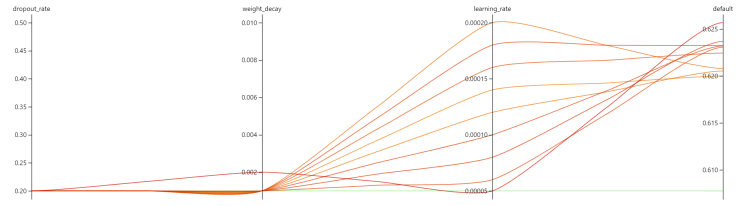
Figure 17: Performance of station 17 before Grid Search



(a) Grid Search result for vanilla RNN



(b) Grid Search result for LSTM

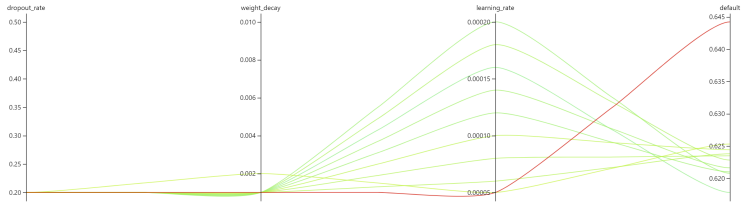


(c) Grid Search result for Bi-LSTM

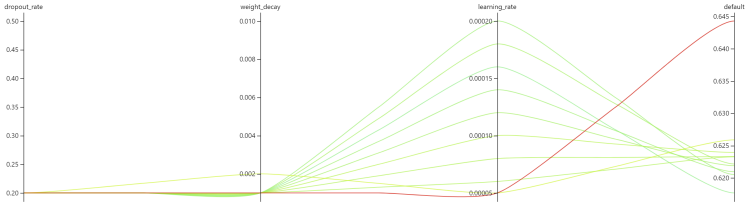
Figure 18: Grid Search result of station index 17 with three models

	Accuracy	Precision	Recall	F1-score	FN	FP	TN	TP
vanilla RNN	0.650934	0.651412	0.650934	0.635754	814.8	180.2	436.6	335.4
LSTM	0.644935	0.645255	0.644935	0.628915	812.4	182.6	444.8	327.2
Bi-LSTM	0.648104	0.648146	0.648104	0.632969	810.8	184.2	437.6	334.4

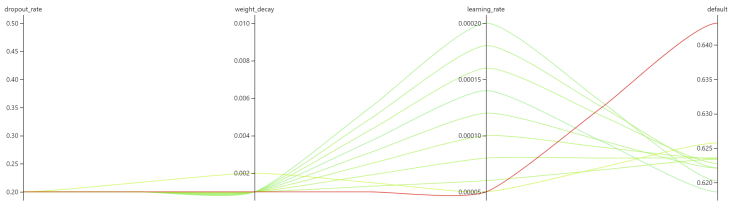
Figure 19: Performance of station 24 before Grid Search



(a) Grid Search result for vanilla RNN



(b) Grid Search result for LSTM

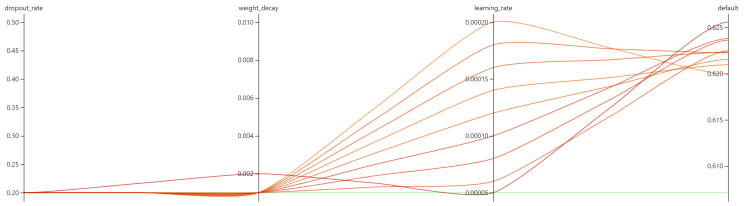


(c) Grid Search result for Bi-LSTM

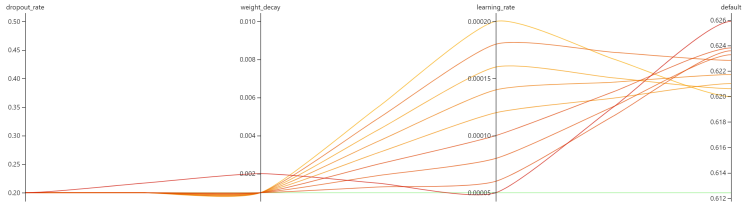
Figure 20: Grid Search result of station index 24 with three models

	Accuracy	Precision	Recall	F1-score	FN	FP	TN	TP
vanilla RNN	0.626465	0.690923	0.626465	0.569125	579.6	40.4	380.2	125.8
LSTM	0.621314	0.666545	0.621314	0.56568	575.8	44.2	382.2	123.8
Bi-LSTM	0.616163	0.657902	0.616163	0.558218	576.6	43.4	388.8	117.2

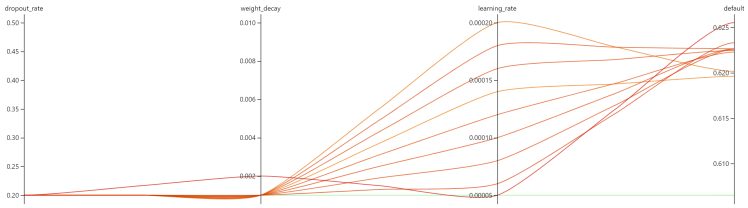
Figure 21: Performance of station 30 before Grid Search



(a) Grid Search result for vanilla RNN



(b) Grid Search result for LSTM

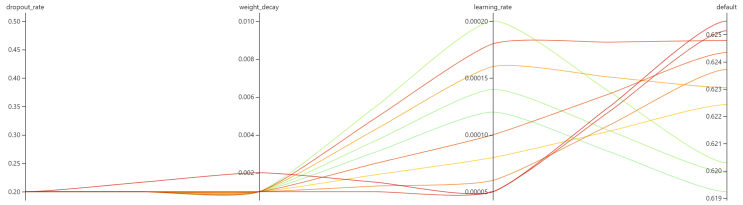


(c) Grid Search result for Bi-LSTM

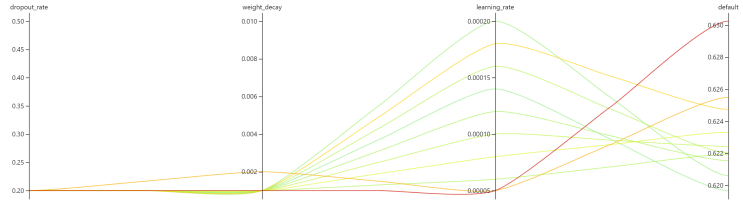
Figure 22: Grid Search result of station index 30 with three models

	Accuracy	Precision	Recall	F1-score	FN	FP	TN	TP
vanilla RNN	0.63693	0.638138	0.63693	0.635183	881.8	406.2	502.2	711.8
LSTM	0.63781	0.639243	0.63781	0.635456	907.2	380.8	525.4	688.6
Bi-LSTM	0.636291	0.639472	0.636291	0.632765	907.8	380.2	529.8	684.2

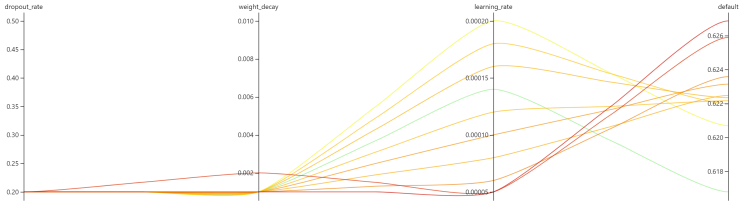
Figure 23: Performance of station 51 before Grid Search



(a) Grid Search result for vanilla RNN



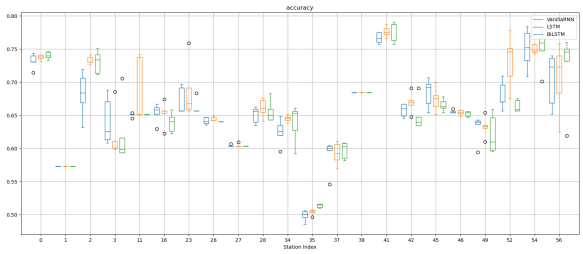
(b) Grid Search result for LSTM



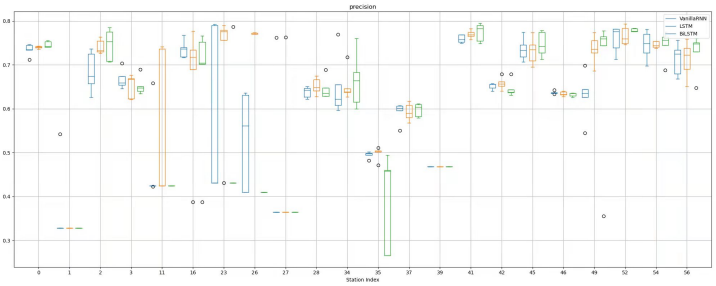
(c) Grid Search result for Bi-LSTM

Figure 24: Grid Search result of station index 51 with three models

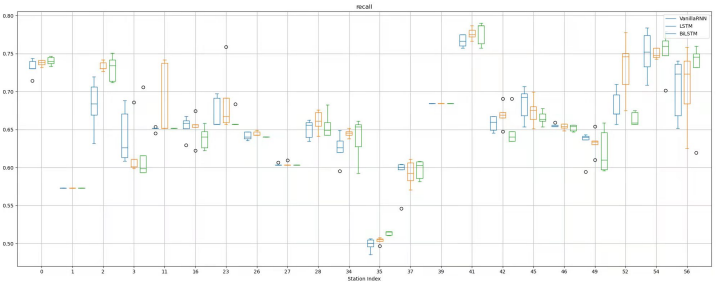
A.2 Boxplot of each evaluation metric of three models for each of 22 stations



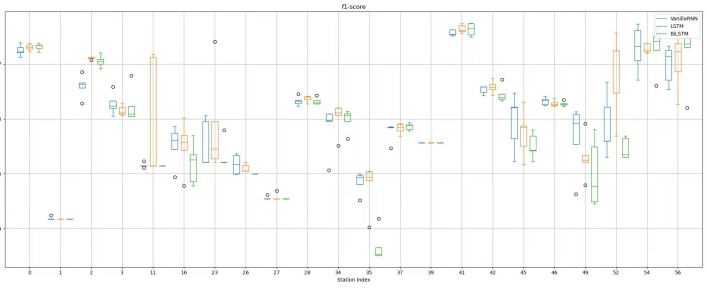
(a) Boxplot of accuracy of three models for each of the 22 stations



(b) Boxplot of precision of three models for each of the 22 stations



(c) Boxplot of recall of three models for each of the 22 stations



(d) Boxplot of f1-score of three models for each of the 22 stations

Figure 25: Boxplot of evaluation metrics of three models for each of the 22 stations (for each station, values from left to right is Accuracy, precision, recall, f1-score)