Bachelor thesis TU Delft

How does a CNN mixed with LSTM methods compare with the individual one in predicting earthquakes?

Irtaza Hashmi[*]
Responsible Professor: Elvin Isufi[†]
Supervisors: Mohammad Sabbaqi, Maosheng Yang[‡]
EEMCS, Technische Universiteit Delft, The Netherlands

January 23, 2022

Abstract

Earthquakes are one of the most dangerous natural disasters that occur worldwide. Predicting them is one of the unsolved problems in the field of science. In the past decade, there has been an increase in seismic monitoring stations worldwide, which has allowed us to design and implement data-driven and deep learning solutions. In this paper, we will investigate how CNN mixed with LSTM methods compare to the individual ones in predicting earthquakes given 30 seconds of seismic data before an earthquake occurs, also known as precursor data. Preliminary results show that a CNN mixed with LSTM has the best training accuracy while an individual LSTM performs best on unseen data.

Keywords: Deep neural network, convolutional neural network, long short-term memory, earthquakes prediction, seismic data, precursor data

1 Introduction

Among natural disasters, earthquakes have the potential to cause the most damage in the shortest time. They have a destructive nature that can cause a lot of damage to the entire ecosystem and may cause serious injuries or loss of human lives. Due to this, researchers have developed an interest in seismic events from the past decade [1]. An earthquake occurs when there is a sudden movement in the tectonic plates that make up the Earth's crust. The most damage is caused at the edge where the tectonic plates meet. When two tectonic plates collide and grind against each other, the stress can travel large distances and affect other regions [2]. The time between different seismic events can be extremely irregular. This threat can't be averted due to its nature, but if analysed, we can minimize the damage it causes to the ecosystem and prevent the loss of human lives. This is precisely the main motivation behind this work.

One way to minimize the damage of earthquakes is to predict them and issue a warning in that region. This way, the region will be ready for the impact. Predicting earthquakes

[*]I.Hashmi@student.tudelft.nl
[†]e.isufi@tudelft.nl
[‡](m.sabbaqi, m.yang-2)@tudelft.nl

has been a field of interest for a long time but is generally considered a difficult task due to its random nature. By predicting an earthquake, we imply finding the time it will occur, its location and magnitude [1]. There are three types of earthquake predictions models that differ in the type of analysis and the time considered to make the prediction. Earthquake predictions can be short-term ($<$ 1 year), intermediate-term (1-10 years) or long-term (10 years or above) [3]. According to Hayakawa [4], "Short-term earthquake prediction is the only useful and meaning for protecting human lives and social infrastructures" from seismic events.

There are two general approaches to predicting earthquakes which are trend-based and precursor-based. Trend based relies on statistical methods to identify the periodicity in the occurrences of earthquakes without using seismic measurements [3]. This approach is usually used for long-term prediction of earthquakes and performs poorly in short-term predictions [5]. The precursor-based approach relies on geophysical methods that have a correlation with earthquake activity, such as seismic measurements electromagnetic anomalies, radon gas emissions etc., and is used for short-term predictions [6]. This paper will focus on the precursor-based approach as we are interested in short term predictions.

So far, great effort has been made by different researchers to develop multiple models that can predict earthquakes, however, they have been unsuccessful [7]. Earthquakes have a random behaviour that doesn't allow us to predict the exact time, location and magnitude of an impending earthquake. In the last decade, deep neural networks have been successful in solving complex pattern recognition and classification problems in different domains.

1.1 Background literature

Recently, researchers have published papers on using time-series data with deep neural networks to predict earthquakes. In these papers, the input is defined in several different ways. For example, Perol et al. [8] used Convolution Neural Network (CNN) for earthquake detection and location using a single raw time-series waveform partitioned into windows of equal time-steps. This method was orders of magnitude faster than previously established methods. Another approach used by Yoon et al. [9] was to transform the time-series data to a spectrogram of the waveform. The spectrogram would have the same information as the time-series data but the data representation would be more convenient for a deep neural network problem. Furthermore, research has also been conducted on predicting earthquakes using seismic precursor-data, which is exactly what we will be doing in this paper. In 2018, Ibrahim et al. [10]. compared 1D CNN, 2D CNN and RNN to predict earthquakes using seismic precursor data from Bay Area, California. 1D CNN, 2D CNN and RNN had the training accuracy of 56%, 60%, and 82.5%, while their test accuracy was 54.2%, 52.6% and 54.5% respectively. As we can see, the test accuracy is poor across all three models. It has been established that using seismic precursor data, there is no proven analytical method to predict earthquakes [11].

Additionally, researchers have also published papers on predicting different parameters of earthquakes that may help us predict an upcoming earthquake. For example, a three-layer perceptron neural network was implemented with a backpropagation algorithm to forecast earthquake magnitudes using time-series magnitude data in Greece [1]. The accuracy of the model for all seismic events in Greece was 80.55%, but only 58.02% for the major seismic events (events that had a magnitude of 5.2 or greater on the Richter scale). Another technique was developed by Sivaiahbellamkonda et al. [12] who compared Long Short-Term Memory (LSTM) with Feed Forward Neural Network (FFNN) to predict earthquakes parameters such as longitude, latitude, time, depth and magnitude. It was concluded that LSTM outperformed FFNN by 59% in terms of the R^2 score.

1.2 Research question

This paper raises the research question: *How does a CNN mixed with LSTM methods compare with the individual one in predicting earthquakes?* using seismic precursor waveforms as input. The paper will compare the three methods (LSTM vs CNN vs CNN mixed with LSTM) using various evaluation metrics and answer which method performs best.

The results of this paper will provide a baseline to compare the three different methods, allowing more accurate solutions and extensions to be built on top of the research covered in this paper.

2 Methodology

As stated in the previous section, since we are interested in short-term predictions, we will use the precursor-based approach. The classification is performed using seismic precursor data of 30 seconds, that is the seismic data of 30 seconds before an earthquake occurs. This is known as the precursor period, represented as the blue waveform in Figure 1. below. The orange waveform represents the warning period, and the green waveform represents the event period, which is when the earthquake occurs. On the right, we have the seismic geophones which record the earth's vibrations vertically and horizontally. This section will cover how the data set was retrieved, the preprocessing applied on the dataset, the deep learning approach that will be taken and the performance evaluation metrics that will be used to evaluate the models.

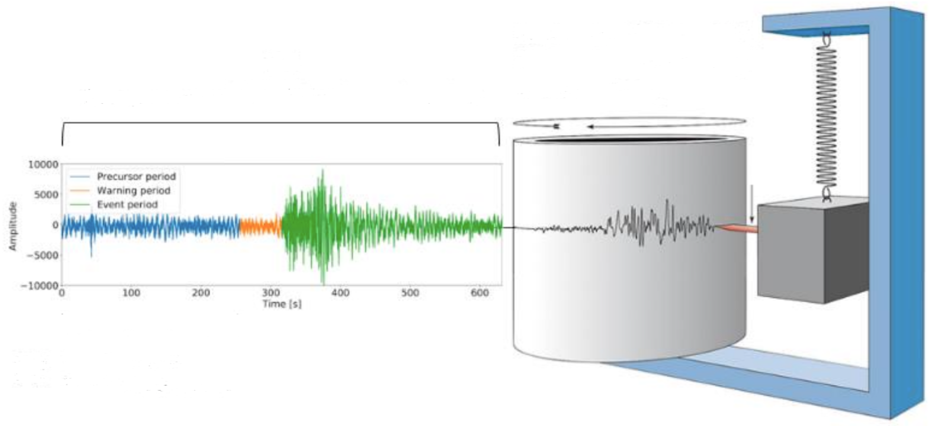


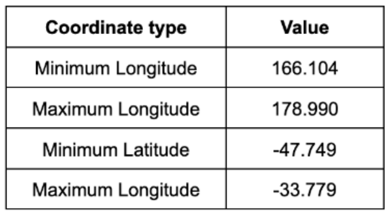
Figure 1: Precursor, warning and event periods shown from a seismic waveform recorded by seismic geophones. [10]

2.1 Retrieving and preprocessing the dataset

The dataset consisted of earthquakes in New Zealand from January 2016 to December 2020 provided by The International Federation of Digital Seismograph Networks (FDSN) [1]. Our dataset comprised 50% waveforms before an earthquake occurs and 50% waveforms of normal behaviour, which was used to train and evaluate the models.

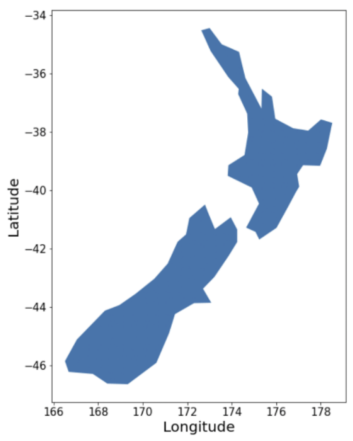
Earthquakes data

We are interested in earthquakes that occur in a specific area that is it occurs in the "bounding box" of New Zealand. The bounding box information of New Zealand is shown in Figure 2. below [13].



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

(a)



(b)

Figure 2: (a) Bounding box coordinates of New Zealand (b) Bounding box of New Zealand.

In total, the region had over 123.000 earthquakes, as shown in Figure 3. below. For each earthquake, we have the event id, timestamp, latitude and longitude coordinates of the epicentre, magnitude and the distance from the surface (depth). This earthquakes data was retrieved using the "Event Service" provided by the FDSN API.

[1]https://www.geonet.org.nz/data/tools/FDSN

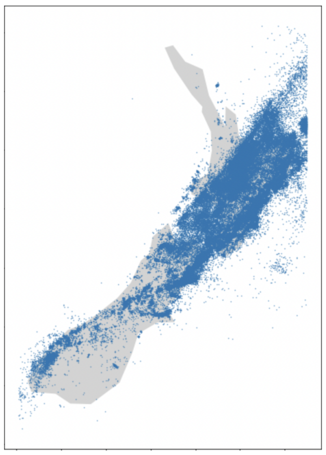
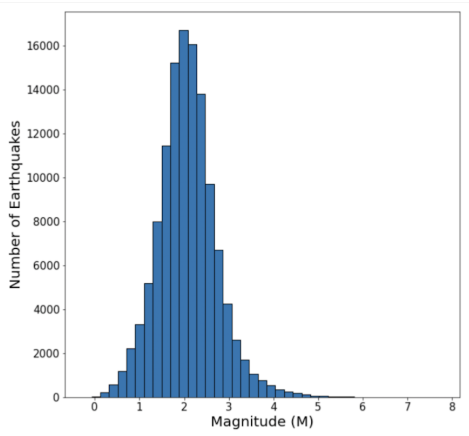
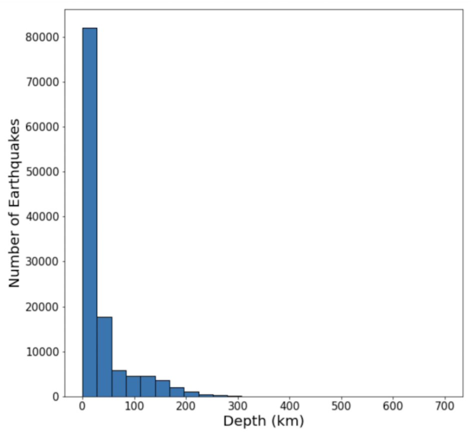


Figure 3: Earthquakes from January 2016 to December 2020. Each dot represents an earthquake on the map.

To get a better understanding of the data, the distribution of magnitude and depth for the earthquakes were investigated. The distribution of magnitude and depth are shown as histograms in Figure 4. below. As we can see, most of the earthquakes have a magnitude of two and a depth of less than 200 kilometres. To obtain a more comparable dataset, we retained earthquakes with magnitudes between one and three that have a depth less than 200 kilometres. After applying this filtering, we were left with approximately 106.000 earthquakes.



(a)



(b)

Figure 4: (a) Earthquakes magnitude distribution. (b) Earthquakes depth distribution.

Seismic data

The next step was to get the seismic precursor data for each earthquake. This data was retrieved using the "Data Select Service" provided by the FDSN API. For each earthquake, we obtained the waveform time series of 30 seconds before the earthquake occurs (precursor period). We had some default parameters for the query such as 'location=10', and 'channel=HHZ'. With these parameters, the waveforms consisted of weak motion (velocity) measured along the vertical axis at 100 Hz (100 samples per second), collected across 58 seismic stations [13]. The 58 seismic stations are mapped on the New Zealand map in Figure 5. below.

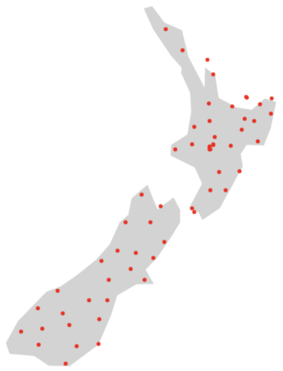


Figure 5: 58 seismic stations used to retrieve the seismic data.

The Figure 6. below shows the waveforms collected from three different stations, represented by three different colours, 30 seconds before an earthquake occurs (for clarity, only seismic waveforms from 3 stations are shown instead of 58). The earthquake occurs after the 30th timestep. As we can see, different stations have different ranges of velocity, therefore we normalized the data as shown in Figure 7. below. These normalized waveforms will be used to train and evaluate the models.

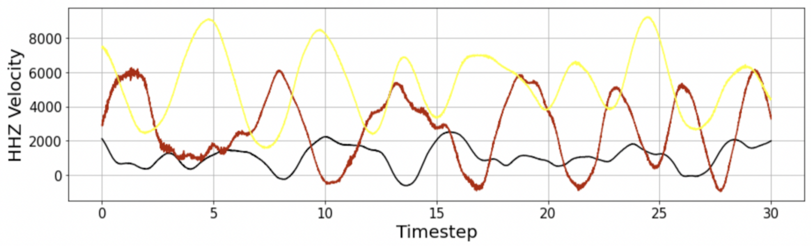


Figure 6: Precursor seismic waveforms sampled at 100 Hz. After the 30th timestep, the earthquake occurs. The three different colour waveforms represent three different stations.

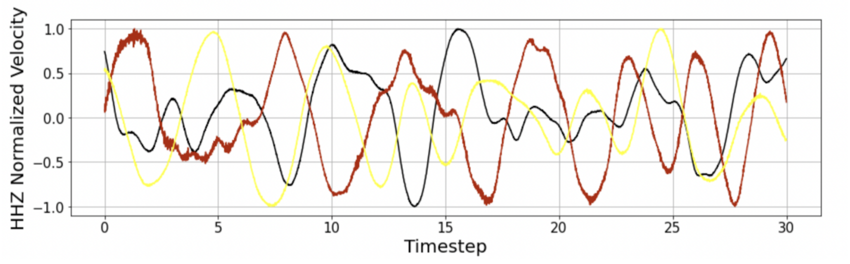


Figure 7: Normalized precursor seismic waveforms per station.

Normal behaviour data

Around the world, the chance that two earthquakes occur within 3 days is around 6% [14]. Therefore, to get the normal seismic data, 50 minutes was subtracted from the timestamp of the earthquakes and 30 seconds of (normal) seismic data was retrieved.

Downsampling of data

Different frequencies of data were used to train the model to see if it has any impact on the performance. Frequencies of 2, 5, 10, 25, 50 and 100 Hz were used. Also, downsampling usually helps us capture the essential features represented by the waveforms instead of getting bogged down by the high-frequency details. An example waveform that was downsampled from 100 Hz to 2 Hz is shown in Figure 8. below.

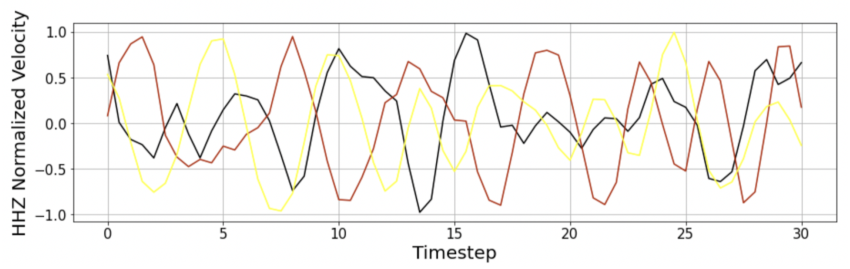


Figure 8: Precursor seismic waveforms downsampled to 2 Hz.

Input and output

The Figure 9. below shows an example of precursor waveforms collected from all 58 seismic stations before an earthquake. For each earthquake and normal behaviour event, such input will be used to train the model. The output of the model will be whether an earthquake occurs or not (1 or 0 respectively).

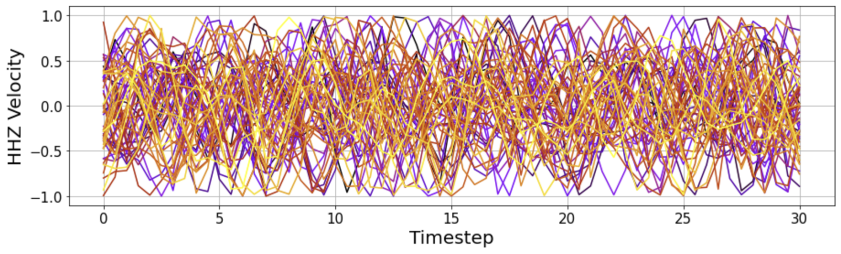


Figure 9: Seismic waveforms retrieved from all 58 stations for an earthquake event.

2.2 Deep learning approach

We will be evaluating CNN, LSTM and CNN-LSTM to see which model performs best in predicting earthquakes.

Convolutional neural networks

A convolutional neural network (CNN) is a type of artificial neural network (ANN) that tries to mimic the human visual system. The convolutional layers in the network are hierarchical feature detectors that are biologically inspired [15]. It has a deep feed-forward architecture and is great in generalizing compared to other fully connected networks [16]. It can learn abstract features and can efficiently identify different types of objects [17]. Furthermore, due to lesser parameters, CNN can be trained smoothly and doesn't suffer from overfitting compared to other deep neural networks [18]. CNN has been widely used on time-series data for forecasting. For example, in 2017, Perol et al. [8] used CNN to predict earthquake magnitudes using time-series data. In 2020, Mehtab et al. [19] used CNN to forecast stock prices using time-series data.

The general CNN architecture is shown in Figure 10. below. We pass our input to the convolutional layer first. The convolutional layer applies filters and extracts features from the input. Here, the network is also trying to learn the best possible filters. The output of the convolutional layer is passed to the pooling layer. The pooling layer is similar to the convolutional layer but performs a specific function on the result, such as max pooling or average pooling. Max pooling takes the maximum value in a filter region, and average pooling takes the average value in a filter region. This layer usually helps with the dimensionality reduction in the network. Next, we have the fully-connected layer, which flattens the results. The output of the fully-connected layer is passed on to the classifier layer where the classification is performed [20].

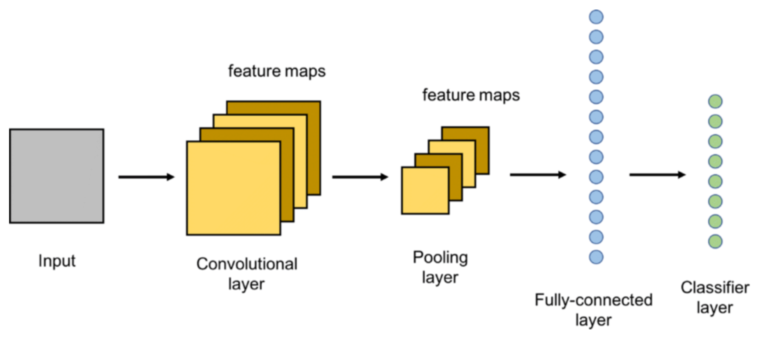


Figure 10: CNN architecture. [21]

Recurrent neural networks - long short-term memory

A recurrent neural network (RNN) is a type of artificial neural network (ANN), which uses time-series or sequential data as input and learns their varying patterns [22]. RNN has been applied to many different problems and stands out for their "memory" as they take information from their prior input, which is used to calculate the current input and output. As a result of their internal memory, RNN can retain the important information allowing them to precisely predict what's coming up next. One of the most applied applications of RNN is time-series prediction. For example, in 1998, C.Giles et al. [23] published a research paper on financial prediction using RNN. The next year, Costa et al. [24] published their findings on short-term load forecasting using RNN.

All RNN networks have a chain type architecture, where each cell is connected to the next one. Each cell has a simple structure, consisting of a single neural network tanh layer as shown in Figure 11. below. However, RNN has its limitation. It suffers from a long-term dependency problem, that is they are unable to connect information from long periods [25].

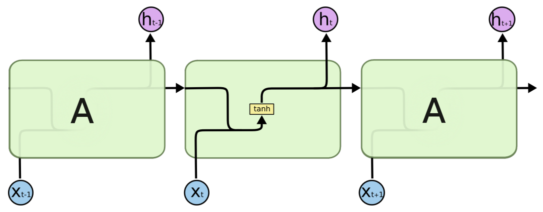


Figure 11: RNN architecture. [26]

Fortunately, long short-term memory (LSTM) doesn't have this problem. LSTM are a type of RNN which are specialised in learning long-term dependencies. That is, they are good at connecting information from long periods. LSTM's architecture is similar to RNN, however, they differ in the cell structure. Instead of having one single neural network tanh layer, there are four of them (represented by the mustard colour), each interacting with each other as shown in Figure 12. below. These layers help the LSTM to retain long-term dependencies which is essential in many different problems.

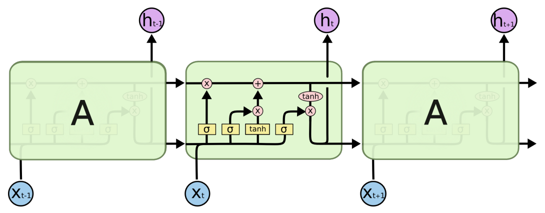


Figure 12: LSTM architecture. [26]

CNN-LSTM networks

The CNN-LSTM architecture includes using CNN layers with LSTM ones. In this type of architecture, CNN focuses on feature extraction while LSTM helps with sequence prediction. This architecture was developed for problems that have time-series data. The model has flexibility and can be applied to a variety of tasks that involve sequential input and output [27]. For example, in 2019 Kim et al. [28] used CNN-LSTM to predict residential energy consumption. The architecture of a CNN-LSTM model first consists of a CNN model, followed by an LSTM model and finally by Dense layer(s), as shown in Figure 13. below.

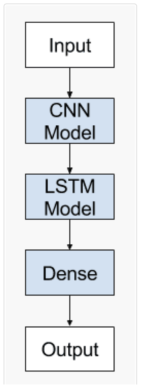


Figure 13: CNN-LSTM architecture. [29]

2.3 Performance evaluation metrics

To evaluate the different deep learning models in predicting earthquakes using binary classification, we will use several performance evaluation metrics:

1. True positives (TP): The total number of times the model predicted an earthquake and an earthquake occurred in the next 10 seconds.
2. True negatives (TN): The total number of times the model didn't predict an earthquake and no earthquake occurred in the next 10 seconds.
3. False positives (FP): The total number of times the model predicted an earthquake but no earthquake occurred in the next 10 seconds.
4. False negatives (FN): The total number of times the model didn't predict an earthquake and an earthquake occurred in the next 10 seconds.

Several other performance evaluation metrics are derived from the previous four metrics. Two of the most used statistical metrics are Sensitivity $\mathrm{S_n}$ and Specificity $\mathrm{S_p}$. $\mathrm{S_n}$, also known as the true positive rate (TPR), is the fraction of samples that are genuinely positive and are correctly identified as such. $\mathrm{S_p}$, also known as true negative rate (TNR), is the fraction of samples that are genuinely negative and are correctly identified as such [30].

$$S_{\mathrm{n}} = \frac{TP}{TP + FN} \tag{1}$$

$$S_{\mathrm{p}} = \frac{TN}{TN + FP} \tag{2}$$

The other two most used performance evaluation metrics for predicting earthquakes are negative predictive value ($\mathrm{P_0}$) and positive predictive value ($\mathrm{P_1}$) [30]. $\mathrm{P_0}$ is the fraction of true negatives among all the negatives predicted by the model. $\mathrm{P_1}$ is the fraction of true positives among all the positives predicted by the model [31].

$$P_0 = \frac{TN}{TN + FN} \tag{3}$$

$$P_1 = \frac{TP}{TP + FP} \tag{4}$$

Another performance evaluation metric that highlights the overall performance of the model is the accuracy. It is the fraction of total number of correct predictions out of all predictions made by the model. It is calculated as follows:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{5}$$

Each of the performance evaluation metric mentioned above highlights a certain aspect of the model's performance. By using different evaluation metrics, we will be analysing every aspect of the model's performance.

3 Implementation

All three models were implemented using Python 3.8. Deep learning library *TensorFlow* [2] was used to train and evaluate the models. Python module *pickle* was used to load and save data while *matplotlib* [3] was used for visualization. These libraries were chosen because they are one of the best in their respective areas, have good documentation and have a large community that uses them. This section covers the conditions under which different models were implemented and trained. Furthermore, the architecture of each model and preventing overfitting is also covered in detail.

3.1 Training conditions

The data was randomly shuffled and split into training, validation and test dataset using the 70, 20, 10 split respectively. It was made sure that the three different datasets contained the same number of zeros (normal behaviour) and ones (earthquakes) for training, as shown

[2] `https://www.tensorflow.org/`
[3] `https://matplotlib.org/`

in Figure 14. below. This was done to make sure we get the best possible results. All models were compiled using the Adam optimizer. Binary Cross Entropy was used as the loss function, and accuracy was used as the metric. The hyperparameters of the models were tuned using the grid search method. The grid searched hyperparameters were batch size, dropout rate, the number of neurons in the layers and the learning rate.

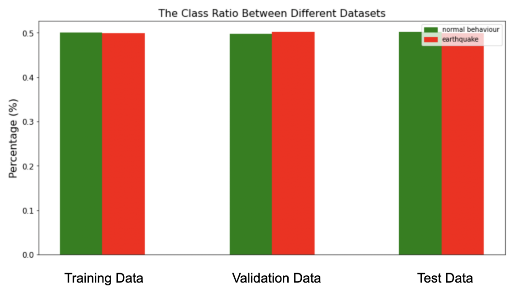


Figure 14: The class ratio between different datasets.

3.2 CNN model architecture

The CNN model was created using the Sequential model with TensorFlow. The first and second layer consists of Conv1D with 128 filters, kernel size of 3 and ReLU as the activation function. The third layer is the Dropout layer with a dropout rate of 0.5 followed by a MaxPooling1D layer with a pool size of 3. Next, the Flatten layer is applied and finally, the Dense layer is used for binary classification using the sigmoid activation function. A batch size of 64 was used to train the model.

3.3 LSTM model architecture

The LSTM model was created using the Sequential model with TensorFlow. The model had an LSTM layer with 60 neurons, a Dropout layer with a dropout rate of 0.4 and a Dense layer with sigmoid activation function for binary classification. A batch size of 64 was used to train the model.

3.4 CNN-LSTM model architecture

The mixture model of CNN and LSTM has a more complex architecture than the individual ones. The first two layers of the model are Conv1D layers (filters = 128, kernel size = 3), each followed by a MaxPooling1D layer (pool size = 3). This part of the model architecture is responsible for feature extraction. Next, the model had two LSTM layers with 128 neurons, each followed by a Dropout layer with a dropout rate of 0.6. This part of the architecture was responsible for time patterns. Finally, a Dense layer was applied with sigmoid activation function for binary classification. A batch size of 256 was used to train the model.

3.5 Preventing overfitting

In order to prevent overfitting, various techniques were experimented with to see which one(s) improve the model's performance. These techniques were adding a regularizer to the loss function, adding a dropout rate and batch normalization. Each of these techniques is covered in detail below.

Regularization

Regularization is the technique applied by modifying the learning algorithm to prevent overfitting. The most common way is to add a penalty on the loss function to certain weights. This is especially useful when the data has a large number of features. The most used regularization methods are called L_1 and L_2. The L_1 penalty sums the absolute values of all terms and adds it to the loss function. The L_2 penalty sums the square of all terms and adds it to the loss function. Both of the regularization methods were applied to all three models with different penalty values, however, they didn't seem to improve the model performance. In fact, it made it worse. The validation loss started to increase rapidly while the validation accuracy decreased, therefore regularization was not used in the final model implementation.

Dropout

Dropout is a technique where randomly selected neurons are ignored during the training of the model to prevent overfitting. This method was applied to all three models, which improved their performance. The validation loss decreased while the validation accuracy increased, therefore the dropout technique was used in the final model implementation.

Batch Normalization

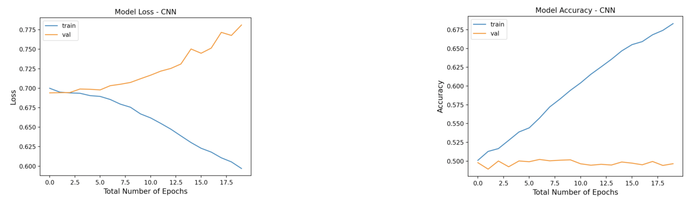
Batch normalization is a technique that standardizes the inputs to a layer for each batch. This stabilizes the learning process and decreases the number of epochs required to train a model to prevent overfitting. For CNN, batch normalization drastically decreased TP, which is not desirable as we want our TP to be as high as possible that is we want to predict an earthquake whenever it occurs. For LSTM, the model didn't learn after batch normalization was applied, and for the mixed model, the validation loss increased drastically while there was no change in validation accuracy. After analysing these results, batch normalization was not applied in the final model implementation.

4 Results and Discussion

This section will report and analyse the final results of the three different models using the performance evaluation metrics described in Section 2.3 above. The training loss, validation loss, training accuracy and validation accuracy for each model are also plotted for comparison.

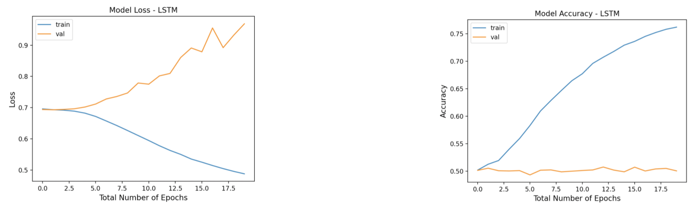
Models' results

The model loss and accuracy are plotted below in Figures 15, 16 and 17 for CNN, LSTM and CNN-LSTM respectively. The blue line represents the training loss and accuracy while the orange line represents the validation loss and accuracy. As the total number of epochs increases, the training loss decreases while the validation loss increases for all three models. This means that the model is performing better on the samples it has seen and worse on unseen data. A similar result is seen on the accuracy graph on the right. As the total number of epochs increases, the training accuracy increases while the validation accuracy remains constant. This phenomenon is known as overfitting. Multiple techniques were used to prevent overfitting, as mentioned in Section 3.5 above. Dropout was the only technique that improved the models' performance, therefore it was used. The rest of them decreased the models' performance, thereby, they were not included in the final model architecture.



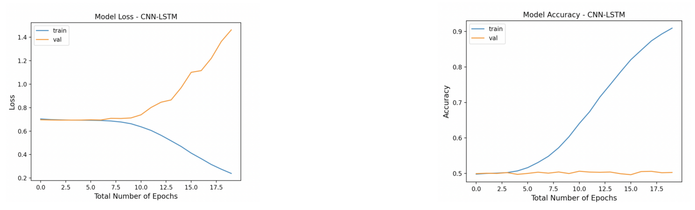
(a) (b)

Figure 15: (a) Training and validation loss for CNN. (b) Training and validation accuracy for CNN.



(a) (b)

Figure 16: (a) Training and validation loss for LSTM. (b) Training and validation accuracy for LSTM.



(a) (b)

Figure 17: (a) Training and validation loss for CNN-LSTM. (b) Training and validation accuracy for CNN-LSTM.

The performance evaluation metrics for all three models are given in Table 1. below. As we can see, the CNN-LSTM model is the best out of the three in predicting earthquakes (TP). This may be due to feature extraction and sequence prediction working together as one unit to predict upcoming earthquakes. It was also the model with the best training accuracy, which makes sense as the architecture of the model was specialised and more complex than the other two. The CNN model is the best in predicting normal behaviour

(TN). Finally, the LSTM model had the best test accuracy. One reason this may be the case beacause it had simple architecure and the overfitting was minumum, which helped it to perform well on unseen data. To take all performance metrics into account, we averaged them out and concluded that CNN-LSTM performed the best out of the three models with an average accuracy of 0.568.

Table 1: Results of the three models using performance evaluation metrics.

Performance Evaluation Metrics for CNN, LSTM, and CNN-LSTM			
Metric Type	CNN	LSTM	CNN LSTM
TP	1874	2227	2470
TN	2541	2266	2039
FP	1889	2164	2391
FN	2530	2177	1934
S_n	0.426	0.506	0.561
S_p	0.574	0.511	0.460
P_0	0.501	0.510	0.513
P_1	0.498	0.507	0.508
Training Accuracy	0.718	0.745	0.858
(Test) Accuracy	0.511	0.513	0.510
Average Accuracy	0.538	0.549	0.568

Using different frequency data

The data that was initially used for the comparison between the three models were sampled at 2 Hz. To further investigate and compare the models, the data was sampled at 5, 10, 25, 50, and 100 Hz. Increasing the frequency of the model didn't improve the model performance, and the results were quite similar to the one above. The results for each frequency and model is in Appendix A.

5 Responsible Research

Reproducibility and repeatability are important aspects of any type of research experiment. This research was designed and conducted in such a way that would allow the experiments to be reproduced and repeated. The steps that were taken to retrieve and preprocess the dataset (see Section 2.1), the training conditions under which the models were trained (see Section 3.1) and the architecture of each model (see Sections 3.2-3.4) are documented in detail such that the experiments are reproducible and repeatable. The models were run multiple times on the data to get reliable results with minimum variance and bias. If the models are run many times, according to the Law of Large Numbers [32], the results should be close to the ones that were obtained, taking randomness into account. Also, it should be noted that the source code is proprietary, which may lead to minor deviations in results and therefore reproducibility, however, the code can be provided upon request.

6 Conclusions and Future Work

The paper raised the question that how does a CNN mixed with LSTM methods compare with the individual ones in predicting earthquakes using seismic precursor waveforms as input. To investigate this problem, the dataset from FDSN was retrieved which had the earthquake and normal behaviour seismic data from New Zealand. The data was preprocessed and trained using the three models. It was concluded that the CNN-LSTM model was the best at predicting earthquakes (TP), while the CNN model was best at predicting normal behaviour (TN). The CNN-LSTM had the best training accuracy, while the LSTM had the best test accuracy. Once all performance metrics were averaged, CNN-LSTM had the highest average of 0.568. The models were trained with different frequencies such as 2, 5, 10, 25, 50 and 100 Hz, but the results were similar. By using this paper as a foundation, with further research, these models can be improved to achieve higher test accuracy. Predicting earthquakes is a challenging problem that does not have a solution yet. The models can be trained with larger and cleaner datasets which may lead to better results. Moreover, different types of deep learning models may be compared to see which ones perform the best aside from the one implemented in this paper.

References

[1] 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, vol. 38, no. 12, pp. 15032–15039, 2011.

[2] A. Bhatia, S. Pasari, and A. Mehta, "Earthquake forecasting using artificial neural networks," ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. XLII-5, pp. 823–827, 11 2018.

[3] T. Bhandarkar, V. K, N. Satish, S. Sridhar, R. Sivakumar, and S. Ghosh, "Earthquake trend prediction using long short-term memory rnn," International Journal of Electrical and Computer Engineering (IJECE), vol. 9, p. 1304, 04 2019.

[4] M. Hayakawa, Earthquake Precursor Studies in Japan, pp. 7–18. 06 2018.

[5] N. R. Council, Predicting Earthquakes: A Scientific and Technical Evaluation, With Implications for Society. Washington, DC: The National Academies Press, 1976.

[6] R. Geller, "Earthquake prediction: a critical review," Geophysical Journal International, vol. 131, pp. 425 – 450, 04 2007.

[7] K. Tiampo and R. Shcherbakov, "Seismicity-based earthquake forecasting techniques: Ten years of progress," Tectonophysics, vol. 522, 02 2012.

[8] T. Perol, M. Gharbi, and M. Denolle, "Convolutional neural network for earthquake detection and location," Science Advances, vol. 4, 02 2017.

[9] C. Yoon, O. O'Reilly, K. Bergen, and G. Beroza, "Earthquake detection through computationally efficient similarity search," Science Advances, vol. 1, pp. e1501057–e1501057, 12 2015.

[10] M. A. Ibrahim, J. Park, and N. Athens, "Earthquake warning system: Detecting earthquake precursor signals using deep neural networks," Stanford, 2018.

[11] M. Wyss, R. L. Aceves, S. K. Park, R. J. Geller, D. D. Jackson, Y. Y. Kagan, and F. Mulargia, "Cannot earthquakes be predicted?," Science, vol. 278, no. 5337, pp. 487–490, 1997.

[12] Sivaiahbellamkonda, Lavanyasettipalli, V. Ramachandran, and M. Vemula, "An enhanced earthquake prediction model using long short-term memory," Turkish Journal of Computer and Mathematics Education (TURCOMAT), vol. 12, pp. 2397–2403, 08 2021.

[13] G. Mazzola, "Graph-time convolutional neural networks," TU Delft, 2020.

[14] "What is the probability that an earthquake is a foreshock to a larger earthquake?." https://www.usgs.gov/faqs/what-probability-earthquake-foreshock-larger-earthquake.

[15] J. Fieres, J. Schemmel, and K. Meier, "Training convolutional networks of threshold neurons suited for low-power hardware implementation," The 2006 IEEE International Joint Conference on Neural Network Proceedings, pp. 21–28, 2006.

[16] S. Indolia, A. K. Goswami, S. Mishra, and P. Asopa, "Conceptual understanding of convolutional neural network - a deep learning approach," Procedia Computer Science, vol. 132, pp. 679–688, 2018. International Conference on Computational Intelligence and Data Science.

[17] Z. Zhang, "Derivation of backpropagation in convolutional neural network(cnn)," 2016.

[18] E. A. Smirnov, D. M. Timoshenko, and S. N. Andrianov, "Comparison of regularization methods for imagenet classification with deep convolutional neural networks," AASRI Procedia, vol. 6, pp. 89–94, 2014. 2nd AASRI Conference on Computational Intelligence and Bioinformatics.

[19] S. Mehtab and J. Sen, "Analysis and forecasting of financial time series using cnn and lstm-based deep learning models," 12 2020.

[20] M. Stewart, "Simple introduction to convolutional neural networks," Medium, 2019.

[21] L. Zaniolo and O. Marques, "On the use of variable stride in convolutional neural networks," Multimedia Tools and Applications, vol. 79, 05 2020.

[22] L. C. Jain and L. R. Medsker, Recurrent Neural Networks: Design and Applications. USA: CRC Press, Inc., 1st ed., 1999.

[23] C. Giles, S. Lawrence, and A. Tsoi, "Rule inference for financial prediction using recurrent neural networks," 1998.

[24] M. Costa, E. Pasero, F. Piglione, and D. Radasanu, Short term load forecasting using a synchronously operated recurrent neural network, vol. 5. 02 1999.

[25] G. Van Houdt, C. Mosquera, and G. Napoles, "A review on the long short-term memory model," Artificial Intelligence Review, vol. 53, 12 2020.

[26] C. Olah, "Understanding lstm networks." https://colah.github.io/posts/2015-08-Understanding-LSTMs, 2015.

[27] F. Elmaz, R. Eyckerman, W. Casteels, S. LatrÃ©, and P. Hellinckx, "Cnn-lstm architecture for predictive indoor temperature modeling," Building and Environment, vol. 206, p. 108327, 2021.

[28] T.-Y. Kim and S.-B. Cho, "Predicting residential energy consumption using cnn-lstm neural networks," Energy, vol. 182, pp. 72–81, 2019.

[29] "CNN Long Short-Term Memory Networks," Aug 2019.

[30] J. Reyes, A. Morales-Esteban, and F. MartÃnez-Ãlvarez, "Neural networks to predict earthquakes in chile," Applied Soft Computing, vol. 13, no. 2, pp. 1314–1328, 2013.

[31] K. M. Asim, F. Martínez-Álvarez, A. W. Basit, and T. Iqbal, "Earthquake magnitude prediction in hindukush region using machine learning techniques," Natural Hazards, vol. 85, pp. 471–486, 2016.

[32] A. N. Shiryayev, "On the law of large numbers," pp. 43–47, 07 2011.

A Appendix

A.1 5 Hz

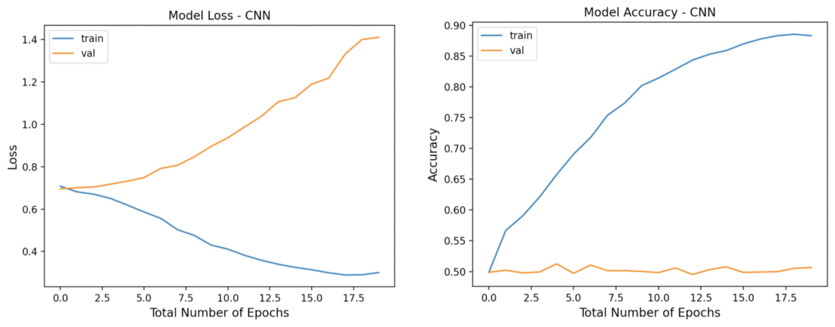


Figure 18: Model loss and accuracy for CNN using 5 Hz data.

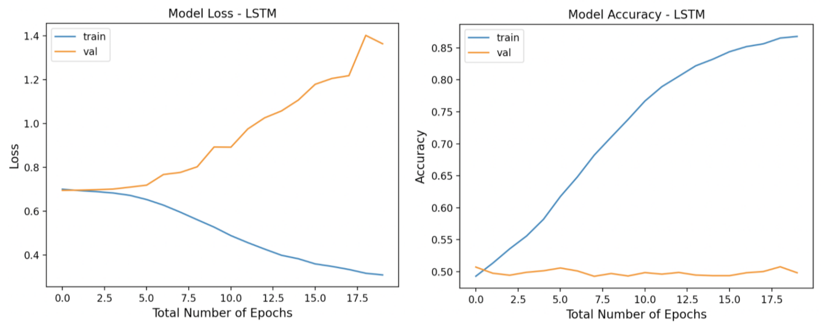


Figure 19: Model loss and accuracy for LSTM using 5 Hz data.

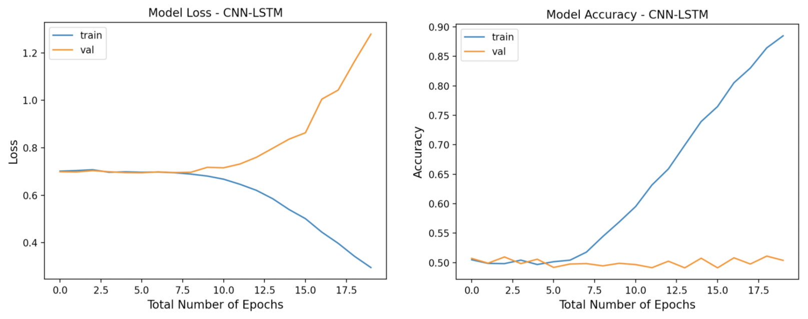


Figure 20: Model loss and accuracy for CNN-LSTM using 5 Hz data.

A.2 10 Hz

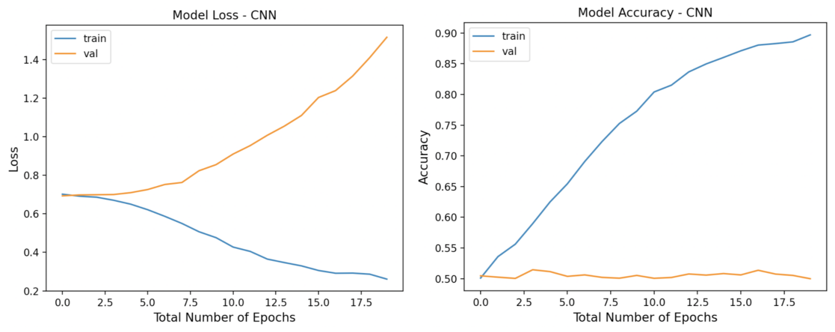


Figure 21: Model loss and accuracy for CNN using 10 Hz data.

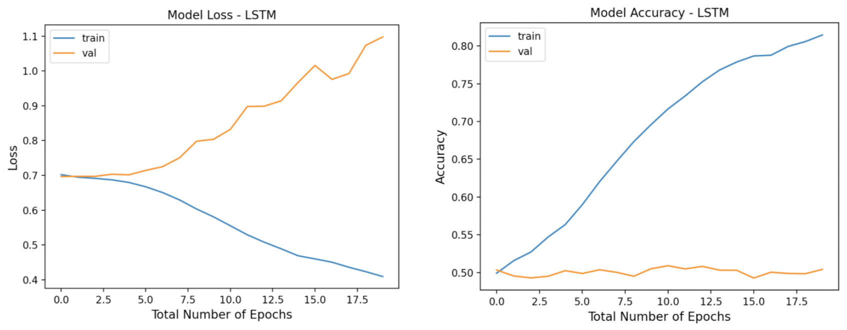


Figure 22: Model loss and accuracy for LSTM using 10 Hz data.

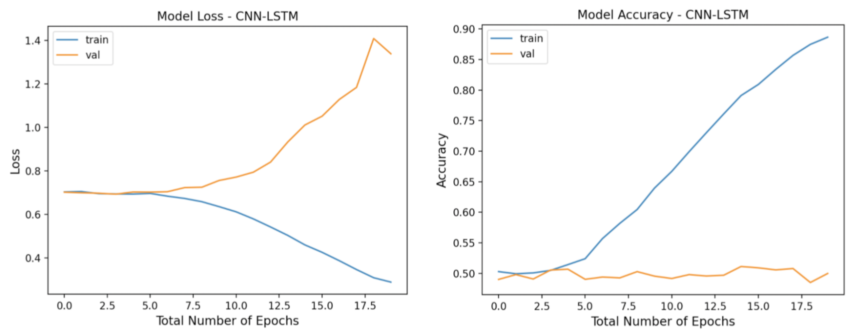


Figure 23: Model loss and accuracy for CNN-LSTM using 10 Hz data.

A.3 25 Hz

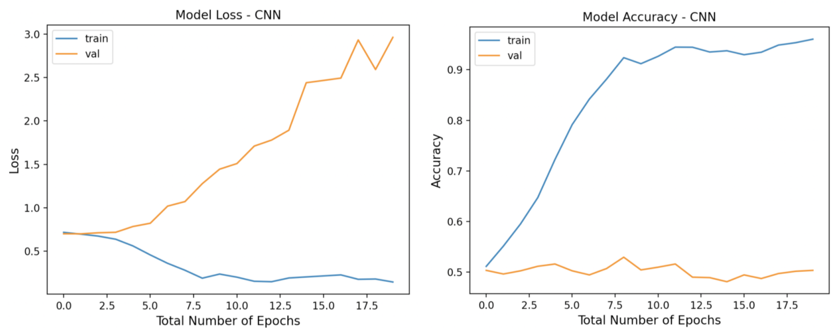


Figure 24: Model loss and accuracy for CNN using 25 Hz data.

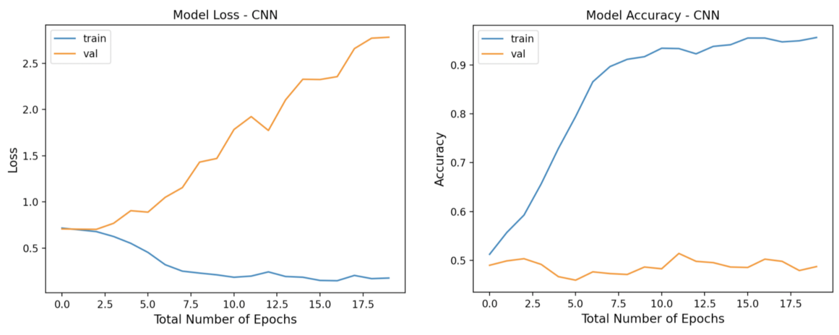


Figure 25: Model loss and accuracy for LSTM using 25 Hz data.

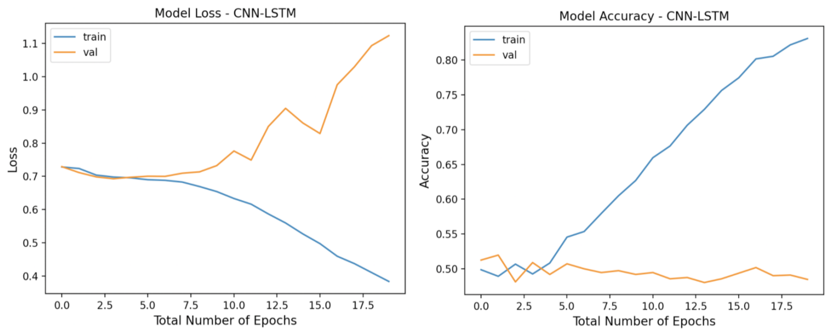


Figure 26: Model loss and accuracy for CNN-LSTM using 25 Hz data.

A.4 50 Hz

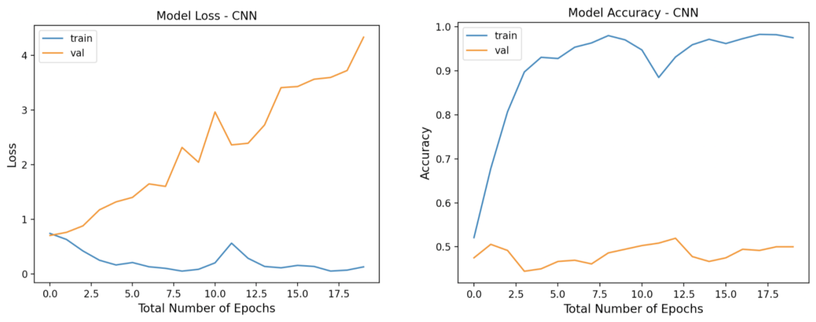


Figure 27: Model loss and accuracy for CNN using 50 Hz data.

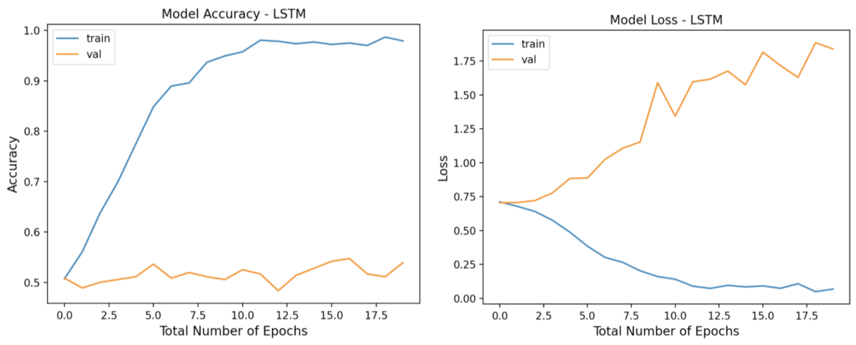


Figure 28: Model loss and accuracy for LSTM using 50 Hz data.

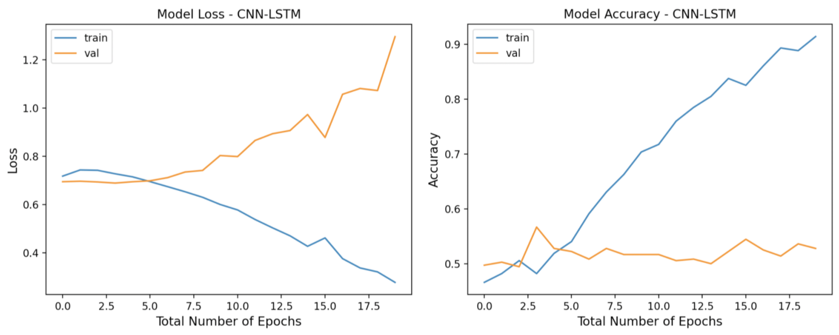


Figure 29: Model loss and accuracy for CNN-LSTM using 50 Hz data.

A.5 100 Hz

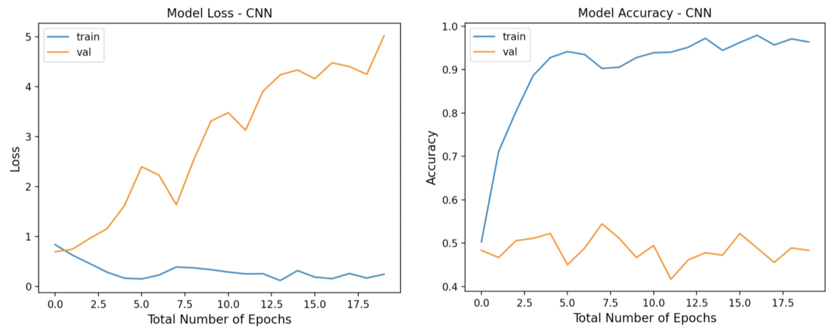


Figure 30: Model loss and accuracy for CNN using 100 Hz data.

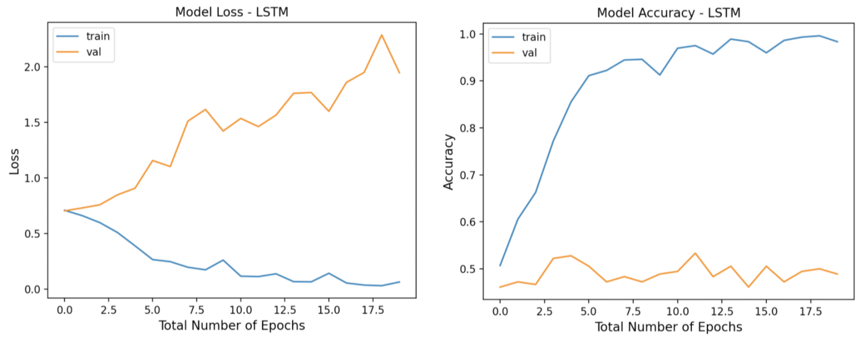


Figure 31: Model loss and accuracy for LSTM using 100 Hz data.

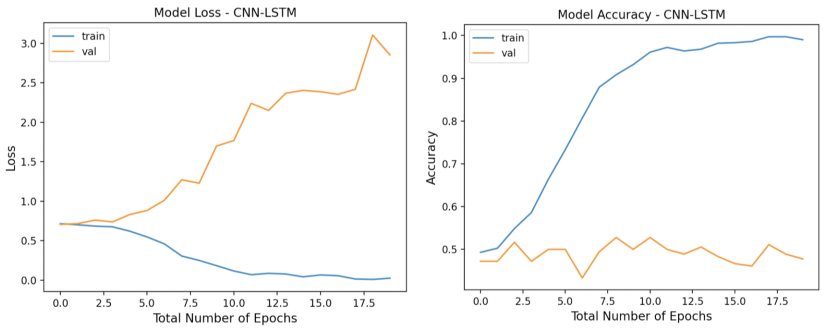


Figure 32: Model loss and accuracy for CNN-LSTM using 100 Hz data.