# Comparing multichannel mixed CNN-RNN to individual models for earthquake prediction

Maikel Houbaer*
Responsible professor: Elvin Isufi†
Supervisors: Mohammad Sabbaqi, Maosheng Yang‡
EEMCS, Delft University of Technology, The Netherlands

January 23, 2022

**Abstract**

Earthquakes can do great harm to the environment and people's daily lives. Being able to predict an earthquake moments before it happens could therefore reduce harm and save human lives. Traditional methods have not been successful yet, but with the rise of techniques focused on deep learning, there is a growing interest to apply them to the field of earthquakes. The placement of stations measuring seismic waves at various locations across regions has also greatly contributed to the possibility of applying data-driven techniques to the problem. A neural network that has been previously successful in the prediction of epileptic seizures - is a CNN mixed with RNN methods. In this paper, we validate the use of this model in predicting earthquakes and compare its performance to individual models. We do this based on seismic measurements before the earthquakes of different stations across New Zealand. The results suggest that our method is not capable of predicting earthquakes with higher accuracy than random guessing.

## 1 Introduction

An earthquake is a sudden shake of the earth's surface realizing energy and thereby creating seismic waves [1]. Such earthquakes could have devastating effects on its environment and human-built structures - thus also affecting people's normal life [2]. In 20 years, nearly a million deaths were caused by earthquakes. Therefore there is a growing interest in predicting an earthquake before it happens in the short term, to decrease damage to human life [3]. These methods are used for so-called earthquake warning systems: systems that can warn regions of damages caused by a potential future earthquake [4].

In the past, two types of techniques have been utilized to try to predict and detect earthquakes. One is more focused on the short term and the other one more on the long term. [5] The first one is trend-based prediction. This type relies on non-seismic measurements. They often involve general data from earlier earthquakes to predict earthquakes happening

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

in the long term. Therefore they are not able to predict earthquakes in the short term - which is essential for the warning systems we target. The other type is precursor-based prediction [6]. This one does involve types of measurements such as seismic waves, temperature, and more. Therefore this is more suited for earthquake prediction in the short term.

Seismic waves are measured by stations that are placed at various locations throughout an area. Traditionally, certain more standard and simple methods have been used to detect patterns in the measurements of these stations to predict an earthquake, such as principal component analysis and regression. [7] However, these techniques have not been quite successful yet to predict earthquakes in the short term. Therefore, there has been a growing interest in using machine learning and deep learning techniques for this task since recently [8]. Many of the recent researches that involve these more advanced techniques are using single-station measurements [7], which leads to sub-optimal results. Furthermore, these researches involve predicting the location of the earthquake moments before it strikes. While this means that preparations can be done with regards to the protection for the earthquake, it still does not predict whether an earthquake actually will happen, or how likely it is.

Research that did study early earthquake prediction independent from location, has not yet led to significant results [9]. But with the rise of new techniques that also have been able to tackle problems in different fields, it is important to use these techniques to also test if they apply to earthquake prediction. An example that we will further work on in this research is a method that is also successfully used for predicting epileptic seizures in patients [10]. The problems seem related to each other; both involve units at different locations measuring waves, which try to predict whether a certain event occurred. The research yielded positive results, reaching an accuracy of higher than 99%.

The goal of our project is to validate different neural networks to do the task of earthquake prediction. In this particular research, a convolutional neural network mixed with recurrent neural network methods will be tested and compared to individual models, such as the recurrent neural network only.

The research question is: "How do multichannel CNNs mixed with RNN methods compare with an individual model?". To answer this question requires a few steps to be taken. First, we need to determine in what ways we preprocess the earthquake data to feed to our neural networks. Second, we need to find the optimal structures for our neural networks, so that we can validate the different neural networks. Finally, we need to evaluate our neural network, so that we can compare it to others and measure the effectiveness of using it.

## 2 Methodology

We implemented and tested a certain deep learning model for early and accurate earthquake prediction. This model is a convolutional neural network (CNN) with recurrent neural network methods (RNN). For comparison, we used simpler individual models, such as an RNN only or a CNN with a multi-layer perceptron as the output layer. Our earthquake prediction problem is a binary classification task between seismic events leading to an earthquake and normal seismic behavior, as visually shown in Figure 1.

Earlier research did not find any particular duration of apparent seismic activity or state that leads to an earthquake. It is very often suggested that earthquakes appear sudden [11] and thus not have long during preliminary signs. In our experiments, we chose to use 30 seconds of data. This means that in case of an earthquake, it is 30 seconds immediately preceding the earthquake. Due to the suggested abrupt nature of earthquakes, we wanted

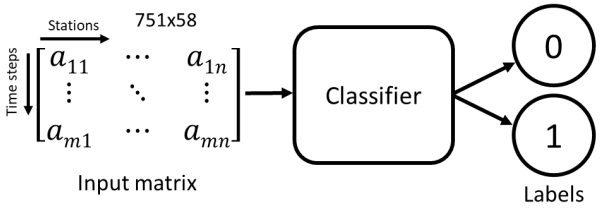


Figure 1: Binary classification task by the classifier, in our case the neural network

a time window that would concentrate on the last seconds. While also allowing signs that might arise tens of seconds before to be included.

We primarily used seismic waves measurements from multiple stations placed at different locations across a region as input data for the models. This data was kept as raw as possible to be fed to our neural networks, to eliminate any bias we might put into the learning process. We wanted the neural network to learn the features themselves. Some preprocessing was done to make sure the data was complete and equal to each other, this includes filtering, normalization, and balancing classes. Generally, the class with more samples tends to be classified with higher accuracy, thus influencing performance measures overall [12]. Therefore, we balanced the dataset with both positive and negative samples. Positive samples are 30 seconds immediately before the earthquakes and negative samples are 30 seconds not leading up to an earthquake. The negative samples were chosen to be 30 seconds starting 50 minutes before each earthquake. Due to the abruptness, seismic measurements should not indicate any signs 50 minutes before the earthquake. Furthermore, there is a very low probability that earthquakes are predetermined foreshocks for the next one [13].

We tested a convolutional neural network (CNN) mixed with a recurrent neural network. The idea behind the structure is to extract spatial features from the data while classifying time-sequential data more optimally. To verify the addition of the RNN component in the CNN-RNN, we compared it to a CNN with an MLP as the output layer instead of the RNN before the output layer. Furthermore, we compared our multichannel model, which takes in data from all the stations, to the individual RNN model that only takes in data from one station. This individual model is being researched by [14].

## 2.1 Dataset

The data we used to conduct our research is a New Zealand earthquake dataset retrieved from the FDSN web service [15]. It includes earthquakes from 2016 up to 2020. For forecasting time-series data, the technique used for preprocessing the data heavily influences the effectiveness of the performance of the neural network [16]. Therefore, we conducted the following steps of preprocessing the data to obtain the event dataset, the dataset with all earthquakes and time windows of normal behavior:

1. Earthquakes in this dataset were selected between longitudes 166.104 and 178.990 and latitudes -47.749 and -33.779, because that represents the area of New Zealand.

2. Earthquakes were filtered that were incomplete, lacking labels such as longitude, magnitude, and depth.

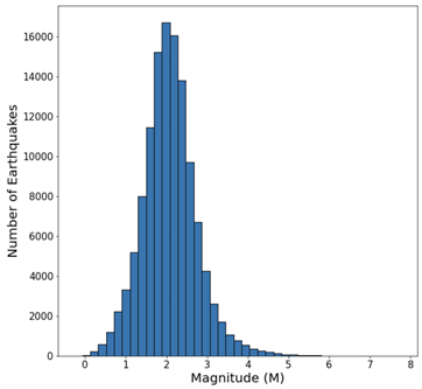


Figure 2: Distribution of magnitudes of earthquakes in the retrieved dataset

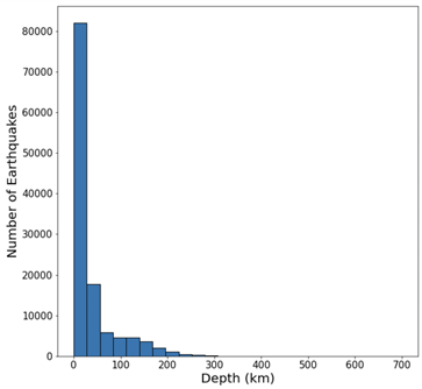


Figure 3: Distribution of depths of earthquakes in the retrieved dataset

3. Earthquakes are selected from a magnitude of 1 up to and including 3, to make sure the model can detect more similar patterns across earthquakes. This selection is based on the distribution of magnitudes of earthquakes in our dataset so far, as shown in Figure 2.

4. Earthquakes are filtered out with a depth of 200 or more to eliminate different patterns that potentially could occur with earthquakes at various depths. This is based on the distribution of depths of earthquakes in our dataset, as shown in Figure 3.

5. Normal behavior, or negative samples, are added as events to the dataset, done in the way described earlier.

At this point, approximately 106.000 earthquakes were left in the dataset, with an equal number of negative samples. For the final dataset, we needed seismic measurements of all selected stations with every event. Therefore we conducted the following steps for this seismic dataset:

1. For each event in the event dataset, seismic measurements of all stations were retrieved from the New Zealand earthquake dataset and put in the seismic dataset.

2. All data is "cleaned"; removing little artifacts in terms of size to make the size consistent.

3. All seismic measurements were normalized, because it often leads to more efficient training times [16]. In this particular case, it means that per window of 30 seconds, the maximum value is set to 1 and the minimum value to -1, and other values are scaled accordingly.

4. Stations are filtered so that every station in our dataset would have full data at every time step of the preceding 30 seconds of every event, thus not being interrupted. This leads to the selection of 58 stations.

5. All seismic measurements are downsampled from 100Hz to a user-defined frequency. This will reduce training time. As the standard downsampled frequency we used 25Hz, but also tested it against 2Hz to see whether it would affect its performance.

## 2.2 Mixed CNN-RNN model

The convolutional neural network mixed with a recurrent neural network is the main model we test in our research. The input data for the model is the input matrix, with one dimension being the time steps, and the other dimension being the stations. Each element in the matrix represents a seismic measurement. A visualization of this is shown in the left-most part of Figure 4.

Normally, the validation set is used to tune hyperparameters and to determine the architecture of a neural network by performing a search, such as a grid search or random search. In our case, however, the validation set never yielded results that correlated with any changes we made to the hyperparameters or the architecture of our model. Therefore we tuned them by considering the balance between the pace of training loss reduction and overfitting. We describe this process more elaborately in Section 3.2. Below, we describe the final model and its components.

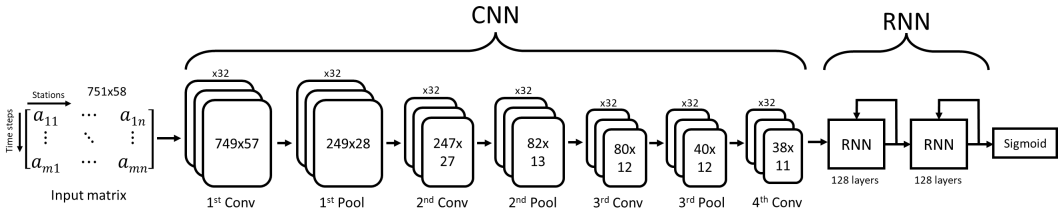


Figure 4: Visual representation of our mixed CNN-RNN model

### 2.2.1 Convolutional neural network

Convolutional neural network (CNN) is a type of artificial neural network that is most commonly applied to computer vision problems and analysis of visual imagery but also commonly applied to recommender systems, financial time series, audio processing et cetera. Because these fields seem to share concepts with the field of earthquake data research, we will also test its performance for earthquakes. Because of the equivariant nature that these types of networks possess, they are able to extract spatial features very well without much overfitting. Since they are not fully connected, they require less memory to store the weights as opposed to more simple networks, such as a multi-layer perceptron.

A typical CNN consists of multiple layers: convolutional layers, pooling layers, and the output layer. The convolutional layer generates a feature map by applying a filter to the input, where the weights are trainable. The pooling layer reduces the dimensions of its input to reduce computational complexity and thus runtime. Finally, there is the output layer to predict the label of the output of the convolutional and pooling layers. This could be a simple multi-layer perceptron or another type of neural network.

In our implementation, the CNN component consists of 4 convolutional layers. A max pooling layer is put between every convolutional layer. This setup also acted as a starting point. The number of kernels in each convolution layer in our network is 32. The kernel size in each convolution layer is $3 \times 2$, which means a size of 3 in the dimension of time steps and a size of 2 in the dimension of stations. The first two max pooling layers have a pool size of $3 \times 2$, and a stride of $3 \times 2$. Except for the third layer, which has a pool size of $2 \times 1$ and a stride of $2 \times 1$. Rectified Linear Unit (ReLU) activation function is used after each

convolution layer to add non-linearity to the model, which allows the identification of more complex patterns in the data.

#### 2.2.2 Recurrent neural network

Recurrent neural networks (RNNs) are a type of neural network. Their unique characteristic is their internal state or memory, which allows them to process sequences of data of variable length [17]. There are different types of recurrent neural networks, but here we will use a "vanilla" type, which is the most standard type. It consists of three layers: the input layer, the hidden layer, and the output layer. The output of the hidden layer is also input to itself for the next time step, allowing the structure to have a memory as data is put in sequentially. Like any artificial neural network, all layers consist of neurons, which all connect to other neurons to the successive layer, much like neurons in a brain. The output of a neuron is a non-linear or linear function applied to the weighted sums of the inputs. In our case, the output layer consists of a neuron indicating the predicted class with the associated input data, after the last time step has been processed of the data.

The RNN component in our implementation for the mixed CNN-RNN model consists of two hidden layers having 128 neurons each. The individual RNN model [14] has one hidden layer of 128 neurons.

#### 2.2.3 Final architecture and training

For our final mixed CNN-RNN, we combined the two components to try to exploit the advantages of both networks. The general architecture and data flow of this model are shown in Figure 4.

This mixed model means that data flows to the network in the following way: the input data first passes the CNN, which reduces the complexity of the data and tries to generalize spatial features. It then passes the RNN, which tries to detect time-sequential patterns. Finally, the output of the hidden layer is going through a fully connected layer which reduces the number of features to one. A sigmoid activation function is applied to the output to predict the label because it is ideal for binary classification due to its range from 0 to 1.

Backpropagation is used to train the model, and the Adam algorithm is used to optimize the model [24]. The learning rate is set at 0.001, which we tested as a sweet spot between training speed and volatility of the training loss. $\beta_1$ and $\beta_2$ for the Adam algorithm are set at 0.9 and 0.999 respectively. As the loss function, binary cross-entropy loss is used.

### 2.3 CNN with MLP

To validate the addition of the RNN in the CNN-RNN, we compared its performance to the individual CNN component connected to an MLP. The CNN has the same architecture used as in the mixed CNN-RNN. The MLP has hidden layers of sizes 400, 100, 50, and 20, in sequential order. ReLU is used as the activation function for each neuron. The MLP serves as the output layer; it predicts the label associated with the input data. A sigmoid is applied to the final output here as well. Backpropagation and the Adam algorithm were used to train and optimize the model. Binary cross-entropy is used as the loss function.

## 2.4 Individual RNN model

We are using the results of [14] to compare our mixed CNN-RNN with the individual RNN model researched by the paper. An essential element of comparison is the structure of the input data: the individual model only takes data from one station during the training phase at a time, instead of them all.

The data for the individual model is constructed by taking the earthquakes from the same New Zealand dataset we use and assigning each earthquake to its closest station. If a model is trained on a station, it takes in data from earthquakes assigned to that station.

This model is relatively simple: the RNN has one hidden layer with 128 neurons, and a fully connected layer with a sigmoid is the output.

## 2.5 Prevent overfitting

Overfitting happens when the model classifies too closely based on a particular set of data, in this case, the training data. Therefore it may fail to reliably fit the underlying distribution, and thus additional data or future data. Multiple techniques could be deployed to prevent or at least reduce overfitting. Several of which we implemented in our models.

### 2.5.1 Regularization

A common way to prevent overfitting is to implement regularization. A penalty term is added to the loss function to penalize a more complex or flexible model. This avoids the risk of overfitting and is more likely to converge to a simpler model that represents the data.

There are multiple ways we can add a penalty term. For our model, we used ridge regression, which means adding the L2 norm of the weights to the loss function used to optimize the model. The regularization parameter is used to determine how much we penalize the model with the L2 norm of the weights. This parameter is highly dependent on the underlying architecture and the input data. For each setup, we picked a value that severely reduced the pace of overfitting while also allowing the model to train.

### 2.5.2 Dropout

Dropout is another technique to combat overfitting. It randomly omits neurons in a layer with a user-defined probability. As an effect, certain classifications are becoming less dependent on certain neurons, thereby improving the ability to generalize, which also results in a reduction of overfitting [18]. Traditionally, this technique is used for fully connected layers, but because recent research has also shown positive results of applying dropout to CNNs [19], we are also implementing it in our CNN.

This technique has shown great improvements on many tasks [20]. In that paper, the dropout probability was shown to display positive effects between 0.2 and 0.5, while showing adverse effects above 0.5. For our setup, we used a dropout probability of 0.3 for our CNN layers, and 0.5 for the RNN. These values were starting points chosen by standard practices. Because a portion of the neurons is disabled during the training phase, the number of epochs it takes to train the model also increases, but training time per epoch is also reduced. Overall this results in no significant increase in training time.

### 2.5.3 Batch normalization

Batch normalization is used to increase the speed and stability of a neural network [21]. It standardizes the input to every layer for each mini-batch. It is argued that training phases are slowed down by the fact distributions among mini-batches and network activations change during training and that this solves this problem. However, other arguments for its effectiveness have also been proposed, such as [22].

Batch normalization is a technique that can be used for CNNs, as well as fully connected layers. For our model, batch normalization is used in the CNN layers between the convolution itself and the activation function. We tested if its implementation would show any reduction in overfitting and an increase in generalization.

## 2.6 Training and evaluation methods

As described earlier, in the training process we balanced the dataset with the two labels we are trying to predict, to avoid any issues associated with an imbalanced dataset. The dataset is split randomly into three sets: the train set, validation set, and test set. This is done randomly to ensure generality and robustness while maintaining the property of balanced labels. They are split into portions of 70%, 20%, and 10% respectively. The train set is used to train the model itself. The validation set is used to validate the performance of the model during and after training, to tweak its hyperparameters, such as the size of a layer or the structure of the model. The test set is used to evaluate the final model.

During training, we are validating the performance of the models by measuring the binary cross-entropy loss and accuracy of the train set as well as the validation set at each epoch. The final performance of the models is evaluated with several measures. We use the test set on the model to construct a confusion matrix, which in our case is 2 by 2. Based on this confusion matrix, we can measure its accuracy, specificity, and sensitivity [23]. Specificity means the percentage of normal seismic behavior predictions when there is no earthquake, which is relevant for avoiding false alarms. Sensitivity means the percentage of predicting an earthquake when there is indeed an earthquake, which is relevant for absences of an alarm in case there are earthquakes. Usually, there is a trade-off between specificity and sensitivity, which is not captured by accuracy. Furthermore, we consider the F1-score, which defines the harmonic mean between specificity and sensitivity. Often it is seen as a better indicator of performance than accuracy, . A higher score is better for all stated measures. Below our measures are defined.

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

$$\text{Specificity} = \frac{TN}{TN + FP}$$

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

$$\text{F1-score} = 2 * \frac{\text{Specificity} * \text{Sensitivity}}{\text{Specificity} + \text{Sensitivity}}$$

were TP means true positive, TN true negative, FP false positive and FN false negative.

# 3 Experimentation and Results

## 3.1 Experimental Setup

The environment we use to develop, train and test our networks makes use of the PyTorch framework. This framework uses Python 3 and makes it quicker and easier to develop our models. To preprocess the data, we made use of Python 3 in Jupyter Notebook. There we fetched the data from the web client through a library and preprocessed it. The neural networks were developed in another Jupyter Notebook with the PyTorch framework. We also trained and tested them there.

The number of epochs we use to train the models is highly dependent on the convergence of the training loss if there is any. Most experiments were run for at least 50 epochs, up to around 250. The batch size used is 32.

## 3.2 Experimentation of mixed CNN-RNN

Normally, hyperparameters and the overall structure would be improved by testing the validation set on the trained model and measuring the improvement the changes brought or not. For example, this could be done by doing a grid search over a set of hyperparameters or a random search by experimenting with different structures. If this is unsuccessful, more drastic changes to the architecture should be made.

For the mixed CNN-RNN, we began experimentation with a setup that is similar to the model defined in [10]. The first model appeared to be too simple; the training loss did not decrease. Therefore, we increased the complexity in a balanced fashion of both the CNN and RNN until the training loss declined. However, in this case, the loss of the validation set would increase, while the validation accuracy would swing around 50%. For example, we tried 3 and 4 convolution layers, multiple kernel sizes, different channel sizes for the layers in the CNN, one or more recurrent layers for the RNN, and multiple values for the number of hidden neurons in the RNN, such as 64, 128 and 256. Yet none of the changes would affect validation.

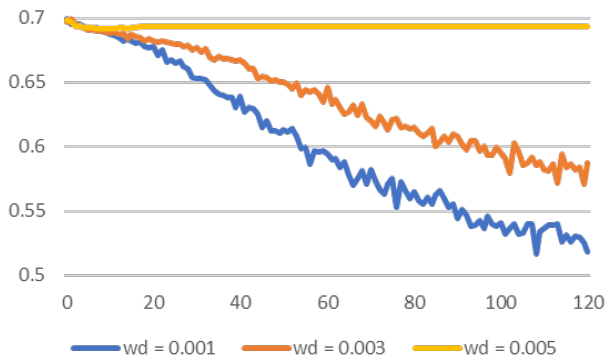


Figure 5: Cross entropy loss of different decay rates during training, against the number of epochs

### 3.2.1 Overfitting measures

At this point, we experimented with multiple measures to combat overfitting, as it appeared that a simple model would not be able to reach higher accuracy. So we carefully implemented the measures we defined in Section 2.5. We tried different setups to see whether several types would conflict with each other. All measures ended up decreasing the rate the training rate would decline, but would not conflict with each other. Therefore we stacked these methods and balanced the parameters so that we would leave a small amount of space for the model to overfit. This results in the parameters found in Section 2.5.

Figure 5 shows an example of using multiple values for the decay rate. It shows the cross-entropy loss of the training set during training. A higher decay rate would decrease the training pace at which the training loss reduces, with eventually the training loss stalling at one value. Interestingly, the model would with a decay rate of 0.005 figure out to always return 0 or 1 to maintain the accuracy at 0.50. For our final model, we used the number in the middle: a decay rate of 0.003.

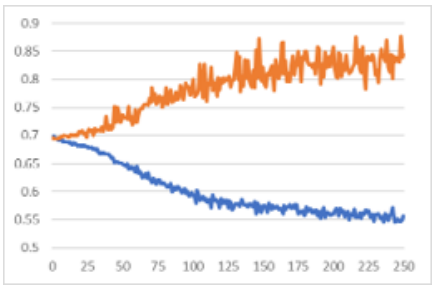


Figure 6: Cross entropy loss of the train set (blue) and the validation set (orange), against the number of epochs

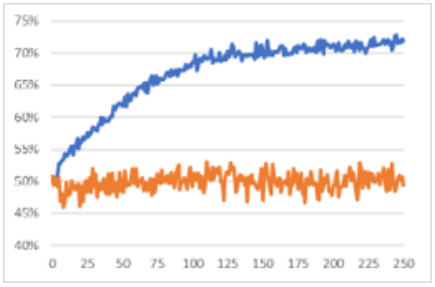


Figure 7: Accuracies of the train set (blue) and the validation set (orange) in percentage point against the number of epochs

### 3.2.2 Final CNN-RNN

In the end, the described process of making the model more complex while reducing overfitting did not succeed to show any positive results where the validation loss would decrease or validation accuracy would increase. The other evaluation measures, sensitivity, specificity, and the F1-score of the validation set also stayed at 0.50. This suggests that there is no correlation between the trained model and the validation set, making it very hard to find a model that would work.

Therefore, we settled on the model as described in Section 2.2. We chose this model to compare to the other models in a rhetorical manner: we already expected it would not yield any positive results. The model itself is a balance between severely trying to reduce overfitting while leaving a little space for the parameters to change.

Figure 6 shows the training and validation losses for this model during a training phase. As we can see, the training loss decreases, without perfectly overfitting all labels. However, the validation loss increases from the start, suggesting the model does not generalize. For a further inspection, we look at Figure 7. Here we see the training and validation accuracies. The training accuracy goes up to around 70%, while the validation accuracy remains stable

swinging around 50%. This also suggests that even though the network tries to perform any type of learning, it fails to generalize the underlying distributions.


Figure 8: Training accuracy and loss of the CNN-MLP against the number of epochs

## 3.3 Experimentation of CNN-MLP

Besides our main mixed CNN-RNN model, we also tested the CNN-MLP model to validate the use of the RNN in the mixed model. The architecture was established by looking at the MLP used in [10] as a starting point. Figure 8 shows the training loss and accuracy for the structure described in Section 2.3. As it shows, the model simply would not train at all. More complex architectures were also tried for this model. Yet only models with a very high number of hidden neurons would succeed to decrease the training loss, nearing the number of events in the dataset. In those cases, it would overfit and the validation accuracy would not change.

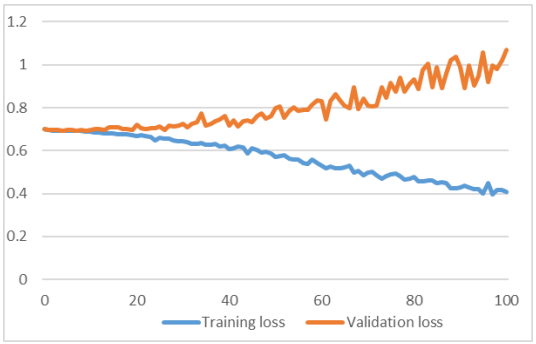


Figure 9: Training and validation loss against the number of epochs

### 3.3.1 Validating downsampled frequency

For our training phase, we chose a downsampled frequency of 25Hz, to make sure the data capture the most details. To test whether this frequency was sufficient, we also applied our model to a dataset with a downsampled frequency of 50Hz. For this particular dataset, we adjusted the decay rate to 0.02 so that it would not overfit too quickly. In Figure 9 we can

still see that this frequency does not bring any advantages for this particular model, as the same problems persist as before.

| Model | Accuracy | Specificity | Sensitivity | F1-score |
|---|---|---|---|---|
| CNN-RNN | 0.512 | 0.487 | 0.537 | 0.511 |
| CNN-MLP | 0.498 | 0.504 | 0.492 | 0.498 |
| RNN | 0.463 | 0.732 | 0.287 | 0.412 |

Table 1: Performance measures of the evaluated models

### 3.4 Test results of the models

Table 1 shows the evaluation measures of the models we tested. For our CNN-RNN and CNN-MLP, we applied the test set to the trained model. For the individual RNN model, we used test results from 7 stations having more than 4000 earthquakes assigned to them. We then calculated our performance measures from the confusion matrix.

As stated before, the results for our multichannel models are disappointing. Because the variance is relatively high compared to the distance of these numbers to 0.50, we interpret them as they are 0.50. This means it has the same score as a random classifier. And so it failed to capture any correlation between the input data and the associated labels.

The individual RNN model does initially seem to show correlation to the input data compared to our models. However, due to the individual dataset being imbalanced, this is not necessarily true. The accuracy is lower than 50%, just as the specificity and the F1-score. Specificity is higher than 0.5, meaning it gives fewer false alarms than our model on its test data. But due to its low F1-score, we cannot say this model performed better.

## 4 Responsible Research

The responsibility of this research mostly regards the reproducibility of the research and the ethical implications. The experiments that we conducted can easily be reproduced. The dataset we used is publicly available through the reference we provided and the preprocessing steps are clearly stated. The models we used can also be easily recreated as most of them are made using standard libraries provided by PyTorch. Furthermore, relevant hyperparameters and other relevant values used are stated in this paper as well.

There also could be ethical concerns. If the outcomes of research regarding predicting earthquakes were actually to be taken into account and used for earthquake prediction systems, that would bring a huge responsibility with it. This responsibility would stretch to many corners of our society, including the economy. A society might become too dependent on the algorithm for short-term warnings, which in case of mispredictions might do more harm than good. For example, the algorithm might give too many false negatives, in which case the society was less prepared for the sudden earthquake than it was without the warning system. In case it gives too many false alarms, society might either be reluctant to them or be too safe, which could also lead to economic damages.

Many of these ethical concerns could be (partially) solved by informing the population as well as governments well about the techniques used to predict the earthquakes. This

allows all instances to make more balanced decisions about preparations and their response to warnings by a system using deep learning techniques.

# 5 Discussion

In this section we briefly reflect on the results, discuss what the limitations were and relate it to other research. The results are initially disappointing, indicating no improvement in performance over guessing. Some improvements that were tried to improve generalization, such as regularization, have shown no positive effects for our model. There are some limitations in our research, and potentially some mistakes could have been made that led to this conclusion.

First, the preprocessing of our data could have some flaws. As stated earlier, the preprocessing steps taken for generating the final dataset have a high influence on the performance of neural networks. In our case, some decisions were made that could have detrimental effects on the results. First off, we locked a time window of 30 seconds, while maybe only the last 5 seconds could have shown signs of an earthquake. This means that the other 25 seconds could greatly overrule the contributions of the 5 seconds. Furthermore, we normalized every time window, instead of a more global approach. This could lead to differences in mean and variance, which might harm the performance of our model. There also could be some limitations in the selection of our data. For our research, we exclusively used the measurements of seismic waves. We did this to purely focus on the relations between seismic waves, as other types of measurements might draw other conclusions. However, other types of measurements, such as temperature, might show correlations with each other or with seismic waves which could mean the model could give positive results.

We compared our model to the individual RNN model in [14] that was researching roughly the same problem as we did in parallel. But the numbers suggested that we could not compare the multichannel model and the individual model on the same level. Namely, the data they used to test was imbalanced; the number of true labels was greater than of false labels. Therefore we could not compare a measurement such as accuracy well. The paper was in progress as well, so the results may be not the final results and potentially could contain flaws.

Finally, it could also mean that the problem was too complex for the available data, or that training the model just would take too long for research of this scope. If we look at other papers, no real positive results have been found yet where the location of the earthquake is irrelevant. In [5], a similar problem was researched, but also there either the occurrence of the earthquake was implicated, or the location of it.

# 6 Conclusions and Future Work

In this paper, we described and implemented our convolutional neural network (CNN) mixed with methods from recurrent neural networks (RNN) to predict occurrences of earthquakes in the short term. We evaluated this neural network model and compared it to other individual models, including a CNN with an MLP as output and an RNN.

The results in Section 3 show that our mixed CNN-RNN does not predict the occurrences of earthquakes in the short term better than random guessing. It means that the mixed CNN-RNN is not suitable to use for any implementation regarding earthquake warning systems. Therefore, there is practically no need to compare our model to the other models in terms

of improvement, since its performance is the practical bottom line. We tried to improve our model by making it more complex and by combating overfitting with several techniques. We also tried adjusting the frequency of the input seismic measurements. However, none of these approaches was able to improve our model in terms of generalization. This should not come as a great surprise - previous literature also did not show any positive results for our problem. This illustrates that this fundamental problem is very hard to solve, even for newer deep learning techniques.

Future work could concentrate on the data that is used. This should be extended beyond the analysis of seismic waves, and use multiple types of measurements such as temperature. Furthermore, the effects of normalization and other types of data manipulation should be investigated for this particular dataset. Of course, the newest invented techniques should be deployed on this problem to test whether they are in any way successful for earthquake prediction.

# References


[1] Earthquake. (2021). Retrieved December 6, 2021, from https://en.wikipedia.org/wiki/Earthquake.

[2] Earthquake environmental effects. (2021). Retrieved December 6, 2021, from https://en.wikipedia.org/wiki/Earthquake_environmental_effects

[3] Wald, L. (2019). *Earthquake Early Warning - Fine-Tuning for Best Alerts.* https://www.usgs.gov/natural-hazards/earthquake-hazards/science/earthquake-early-warning-fine-tuning-best-alerts.

[4] Ibrahim, M. A., Park, J., & Athens, N. (2018). Earthquake warning system: Detecting earthquake precursor signals using deep neural networks. *Technical Report CS 230.*

[5] Isufi, E., & Mazzola, G. (2021). Graph-Time Convolutional Neural Networks. *arXiv preprint arXiv:2103.01730.*

[6] Bhandarkar, T., Satish, N., Sridhar, S., Sivakumar, R., & Ghosh, S. (2019). Earthquake trend prediction using long short-term memory RNN. *International Journal of Electrical & Computer Engineering (2088-8708), 9*(2).

[7] Perol, T., Gharbi, M., & Denolle, M. (2018). Convolutional neural network for earthquake detection and location. *Science Advances, 4*(2), e1700578.

[8] Huang, J. P., Wang, X. A., Zhao, Y., Xin, C., & Xiang, H. (2018). Large earthquake magnitude prediction in Taiwan based on deep learning neural network. *Neural Network World, 28*(2), 149-160.

[9] Uyeda, S., Nagao, T., & Kamogawa, M. (2009). Short-term earthquake prediction: Current status of seismo-electromagnetics. *Tectonophysics, 470*(3-4), 205-213.

[10] Daoud, H., & Bayoumi, M. A. (2019). Efficient epileptic seizure prediction based on deep learning. *IEEE transactions on biomedical circuits and systems, 13*(5), 804-813.

[11] Bhandarkar, T., Satish, N., Sridhar, S., Sivakumar, R., & Ghosh, S. (2019). Earthquake trend prediction using long short-term memory RNN. *International Journal of Electrical & Computer Engineering (2088-8708), 9*(2).

[12] Chawla, N. V., Japkowicz, N., & Kotcz, A. (2004). Special issue on learning from imbalanced data sets. *ACM SIGKDD explorations newsletter, 6*(1), 1-6.

[13] United States Geological Survey. (n.d.). *What is the probability that an earthquake is a foreshock to a larger earthquake?*. Retrieved December 8, 2021, from https://www.usgs.gov/faqs/what-probability-earthquake-foreshock-larger-earthquake

[14] Du, X. (2022). *Short-term Earthquake Prediction via Recurrent Neural Networks*. Delft University of Technology.

[15] FDSN web services for New Zealand. (n.d.). Retrieved November 14, 2021, from https://www.geonet.org.nz/data/tools/FDSN.

[16] Bhanja, S., & Das, A. (2018). Impact of data normalization on deep neural network for time series forecasting. *arXiv preprint arXiv:1812.05519.*

[17] Tealab, A. (2018). Time series forecasting using artificial neural networks methodologies: A systematic review. *Future Computing and Informatics Journal, 3*(2), 334-340.

[18] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research, 15(1)*, 1929-1958.

[19] Park, S., & Kwak, N. (2016). Analysis on the dropout effect in convolutional neural networks. In *Asian conference on computer vision* (pp. 189-204). Springer, Cham.

[20] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580.*

[21] Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning* (pp. 448-456). PMLR.

[22] Santurkar, S., Tsipras, D., Ilyas, A., & Madry, A. (2018). How does batch normalization help optimization?. In *Proceedings of the 32nd international conference on neural information processing systems* (pp. 2488-2498).

[23] Zhu, W., Zeng, N., & Wang, N. (2010). Sensitivity, specificity, accuracy, associated confidence interval and ROC analysis with practical SAS implementations. *NESUG proceedings: health care and life sciences, Baltimore, Maryland, 19*, 67.

[24] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980.*