A Report

On

**Earthquake Prediction Using Deep Neural Networks**

Prepared By:

**Bharat Bhargava**

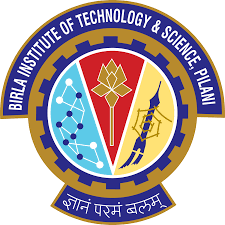
**2017A7PS0025P**

Supervised By:

**Dr. Sumanta Pasari**

**Assistant Professor, Department of Mathematics**

**BITS Pilani, Pilani**



**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI**

**20 June, 2020**

**EARTHQUAKE PREDICTION USING DEEP NEURAL NETWORKS**

1. **INTRODUCTION**

The Himalayan region has witnessed several earthquakes in the past, few of which have been really devastating and have caused huge socio-economic damages. As a consequence of ongoing collision between the Indian plate and the Eurasian plate, this region is one of the world’s biggest seismotectonic active areas. Improvement of seismic hazard assessment requires exhaustive catalogues. The exponentially rising volume of seismic data opens up the opportunity to automate earthquake detection and location and obtain fruitful results through the application of several algorithms.

It is pretty obvious to find earthquake signal detection at the core of observational seismology. It’s not only about detecting havoc-causing earthquakes; a good earthquake detection algorithm should also be sensitive to weak and small events as well, which can have a variety of waveform shapes. Further, it should be robust enough to distinguish between a seismic signal and background noise/non-earthquake signal. It should also be efficient for processing large data volumes.

As a part of my project, I make an attempt to build a fully functional, efficient earthquake detector cum forecaster based on deep neural networks, using a framework built using long-short-term memory (LSTM) units. It is expected to learn the temporal characteristics of the phases of our interest from time-series data recorded on specific earthquake stations. Additionally, an artificial neural network is trained and used as a baseline model to perform a few comparisons with our LSTM model, especially with respect to the ability to learn a hypothesis as a function of several different variables, or, in a more technical sense, features.

Even though the results obtained in this study are not very impressive, an attempt has been made to analyse the plausible reasons of the ineffectiveness. Nevertheless, the study conjectures the use of LSTM based framework over an artificial neural network for earthquake prediction and forecasting.

1. **LITERATURE REVIEW**

Before moving on to implement, it is important to have a fair idea of what has been done in the past. What I found through literature review is that **waveform similarity** from a **specific source** is the soul of several researches conducted so far for reliable automation of earthquake detection.

The methods of earthquake detection can be broadly classified into three categories, which are discussed below.

* 1. **TRADITIONAL METHODS**

Allen et al. (1982) used **automatic phase-detection algorithm** which applies filters to the input data and outputs a function characterizing seismic time series by comparing STA (short-term average) with some defined threshold. Though it turned out to be less sensitive than an algorithm designed only to detect the presence of a signal, it served as the base for many research works conducted thereafter. The work was delimited by absence of powerful computational tools.

Withers et al. (1998) performed comparison of select trigger algorithms through **WCEDS** (Waveform Correlation Event Detection and Location System). It was concluded that STA/LTA (short-term average/long-term average) algorithm with “adaptive window lengths” produced the output meeting the requirements in the best possible manner till that time.

A cumulative analysis of the drawbacks focuses on the following:

1. **Noise versus signal**: Events may be buried in seismic noise. STA/LTA fails in this case.
2. **Computational Limit**: With data growing exponentially, highly complex calculations were computationally prohibitive in these works. It was difficult to generalise them well for large data streams.
   1. **POST TWENTIETH CENTURY WORKS**

The 21st century witnessed two buzzing words in seismology, namely waveform autocorrelation and template matching. Autocorrelation is simply a linear mapping of a signal with a delayed copy of itself, as a function of delay. On the other hand, template matching simply involves a match of a captured waveform image with images of short representative waveforms.

Gibbons and Ringdal (2006) performed frequency-wave number analysis to achieve reliability and robustness in earthquake detection results, by cross-correlating the incoming data stream with the waveform template. The work involved using waveform autocorrelation, making use of **waveform similarity** for earthquake detection originating from a **single** region. The traces of correlation coefficients used to detect earthquakes provided better results than STA/LTA. However, it was computationally intensive to perform such task and hence it didn’t scale to long time series.

Attempts were made to eliminate the use of entire time series and achieve similar results through template matching, involving only a small set for correlation. This is evident from the works of Skoumal et al. (2014). However, the dependence of accuracy on the number of templates ascertains its computationally intensive nature. PCA (Principal Component Analysis) attempts to reduce the number of templates (Benz et al. (2015), Barrett et al. (2014)) as input to the model.

Template matching and PCA, however, do not generalize well to unseen signals; they mainly work well for **repeating signals**. Related works include FAST (Yoon et al.), an acronym for “**F**ingerprint **a**nd **S**imilarity **T**hresholding”, which though has similar disadvantages as template matching, but the computational cost involved is significantly less.

* 1. **DATA SCIENCE TO THE RESCUE**

Shallow neural networks for detecting seismic signals (Wang et al. (2009), Madureira et al. (2009)) have enabled learning a compact set of parameters based on input dataset. It predicts the output depending upon the number of classes to predict, and serves as the base to several deep learning researches conducted on earthquake detection.

Deep learning algorithms serve as the state-of-the-art for object detection, face recognition, natural language processing, speech recognition etc. The impact of deep learning on earthquake detection has been very significant and enthralling, achieving close to perfection in prediction of earthquake. Perol et al. (2018) used Convolutional Neural Networks for this purpose. The generalization of the results is way better than autocorrelation and template matching for waveforms never seen during training. Since there is only the need to store parameters rather than waveform images as a whole, the algorithm is computationally very cheap. It takes relatively lesser time to train the model, and unlike previous methods, the model also gives a probabilistic location of earthquake.

Time series was way earlier recognized as a tool for earthquake detection. The Recurrent Neural Networks, widely used in Natural Language Processing and Speech Recognition, make predictions based on input and the internal state and produces a time series probability of probability outputs. RNNs can learn dynamic temporal relations within a time sequence. Use of LSTM (**L**ong **S**hort **T**erm **M**emory) and GRU (**G**ated **R**ecurrent **U**nit) further help to avoid the problem of vanishing and exploding gradients.

Considering the benefits of CNN and RNN, how about feeding the output of CNN to RNN? Well, this has been done by Mousavi et al. (2019). The corresponding paper introduces CRED, a model using CNN for feature extraction, RNN/LSTM for feature learning and finally fully connected layers for classification. Following are the highlights of the model:

1. The model learns spectral features rather than the waveforms themselves, hence fares well in distinguishing between noise and signal.
2. The model is residual in nature to avoid deeper layers not learning anything.
3. LSTM has been used instead of simple RNN to avoid vanishing/exploding gradient.
4. CRED fares better compared to STA/LTA, template matching and FAST in several aspects, as:
   1. **Performance**: Stores weights, rather than waveforms.
   2. **Accuracy of prediction**: Better precision and recall.

**3.** **EXPERIMENT-1: Reproducing the Results of CRED**

**3.1 DATASET**

The dataset used is hosted on <https://dl.orangedox.com/smTbdN7fdprNKR876Q>. It consists of a whopping 550,000, 30-second, 3-component seismograms, recorded by 889 stations in North California. The dataset is **pre-processed** and the accompanying metadata lists the associated P-wave and S-wave arrival times for events labelled as earthquakes. Further, the dataset is equally split into earthquake events and seismic noise, thus preventing the inherent skewness in data.

**3.2 ARCHITECTURE**

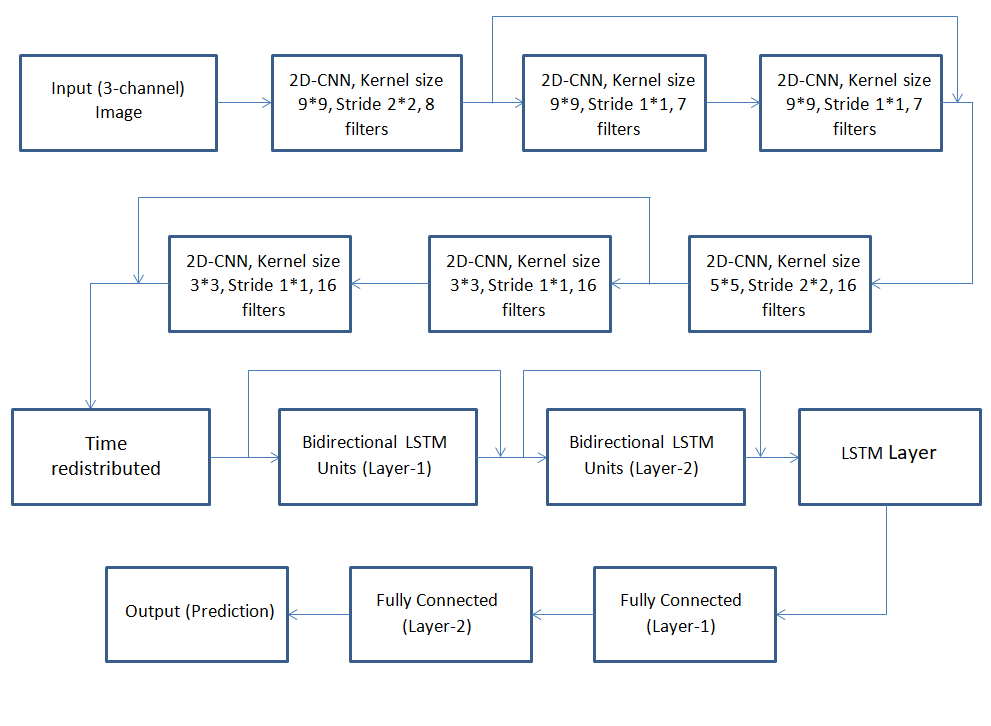
****

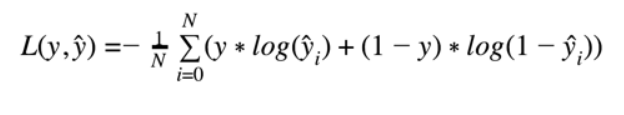
Figure 1. Architecture of the model

**3.3 EXPERIMENT**

The experiment involved a straightforward attempt to reproduce the results as mentioned in the research paper[[1]](#footnote-1). First, spectrograms were constructed using Short-Time Fourier Transform. Additionally, ground truths were generated so as to make the problem a supervised learning problem.

Thereafter, the entire dataset was split into training (80%), validation (10%) and test (10%) sets. **Binary cross-entropy** was used as the loss function and ADAM algorithm was used for optimisation.

**3.3.1 A Note on Binary Cross-Entropy**

Binary cross-entropy tells how far, on an average, the prediction is from the true value. Here, the true value is either zero or one. The formula for binary cross-entropy is: ****Here, ŷ­i is the predicted value while yi is the actual value for the ith instance

The formula for binary cross-entropy comes from an extension of Bernoulli distribution, using **M**aximum **L**ikelihood **E**stimation (MLE).

**3.4 RESULTS**

The results mentioned in the research paper are stupendous; the authors claim a maximum F-score of **99.95**. However, reproducing these results is a daunting task and requires huge computational resources. The authors claim to have performed 62 epochs on the entire training and validation set, with the entire training performed on one Tesla V100-PCIE GPU. They further claim to have trained the model in 82.5 hours!

With computational constraints, especially absence of a dedicated GPU, it became very difficult to reproduce the findings. After barely managing to train the model for four epochs, with each epoch taking approximately 1.5 hours on Google Colaboratory, I got a training accuracy of ~75%, a validation accuracy of ~57% and a test accuracy as low as 52%. The results cannot be accepted. Further, it is obviously known that it is a case of improper training of the model.

An additional attempt of reducing the number of (possibly correlated) features was carried out through **P**rincipal **C**omponent **A**nalysis (or, PCA for short). As considered an “arbitrary method” of analysis, it turns out that there was no improvement seen whatsoever- a single epoch took rather 106 minutes now.

**3.5 CONCLUSION**

A spectrogram provides extensive data for analysis of seismic waves. It provides a visual method of representation of the strength (loudness) of a signal at various frequencies present in a waveform over time, and hence carries along with it extensive information. As a result, there is a clear distinction between noise and a genuine signal. However, utilising such vast data to the fullest is something which demands huge computational resources.

As a result, further analysis has been carried out with only tabular data. This, in itself, imparts certain restrictions to the learning of our model, details of which will be provided during analysis of the corresponding work(s).

1. **EXPERIMENT-2**

This experiment involved using tabular time-series data for earthquake detection and prediction. The data is first pre-processed so as to extract data corresponding to only those features which are generally used in tabular data analysis for earthquake forecasting. Then, a baseline model in the form of an **a**rtificial **n**eural **n**etwork (ANN) is trained for classification and regression purpose. Thereafter, an LSTM-based model is trained and the results are compared with those of the ANN.

* 1. **NUMERICAL TIME-SERIES DATA**

Numerical time-series data, or tabular time-series data, or simply time-series data, is a sequence of discrete data points listed/indexed in time order. It is generally assumed that a time-series data has some systematic pattern, with trends and seasonality being the most common patterns.

There are two buzzing words associated with processing of time-series data, especially in the field of sequence modelling. They are:

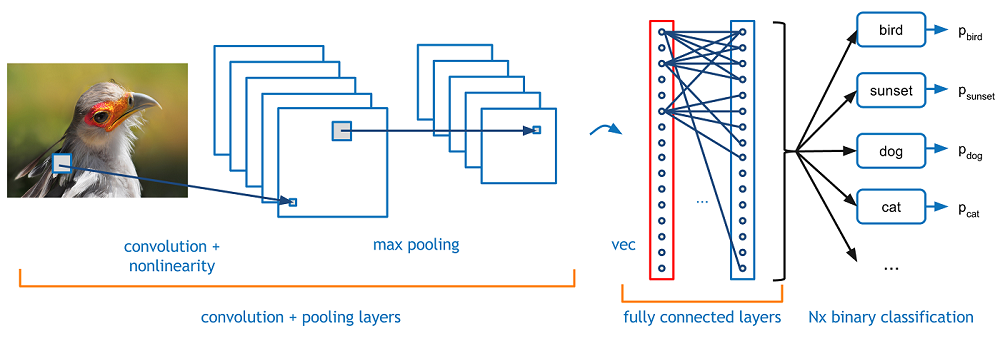
1. **Time-series analysis**: It includes methods for processing time-series data so as to extract meaningful statistics from it, or to find out some other characteristics of the data.
2. **Time-series forecasting**: It involves the use of a mathematical or a conceptual model to predict future values depending upon some pattern in previously observed values.

Today, time series analysis and forecasting find major applications in the fields of weather forecasting, astronomy, pattern recognition, mathematical finance and signal processing.

* 1. **TIME-SERIES DATA AND EARTHQUAKE PREDICTION**

Time-series data analysis is prevalent and effective in any domain of applied engineering which involves temporal measurements. As the earthquakes are expected to have a temporal dynamic behaviour, time-series analysis of the corresponding earthquake data seems theoretically sound. Several deep learning techniques such as RNN (**R**ecurrent **N**eural **N**etwork), GRU (**G**ated **R**ecurrent **U**nit) and LSTM (**L**ong **S**hort **T**erm **M**emory) are specifically designed to analyse time-series data. They are further seen to achieve state-of-the-art performance in this kind of analysis.

Research works on earthquake prediction are also seen to apply convolutional layers on spectral data, mainly for feature extraction. An important question which arises here is, given a tabular dataset, is CNN really required? To understand more about this, take a look at Figure 2.

Figure 2. Prediction using a Convolutional Neural Network

In simple terms, CNN performs feature extraction. Through the application of relevant filters, a CNN is able to learn certain high-level as well as low-level characteristics of our data. This is particularly a fruitful approach to apply especially in case of datasets involving **images**, such that a feature extracted from a group of pixels maintains the same relative position in the shrunken frame as the original image, thus reducing the computation to a large extent for further processing.

However, in case of tabular data, there does not seem any logical reason why we should apply a CNN before analysing the time-series: we are certainly not extracting out features such as latitude, longitude etc., rather we already have them in our dataset. Should a CNN layer be applied, it ought to assign some weights to the features and hence work in a similar manner as the artificial neural network, which too assigns weights to the variables and is able to learn a complex function for a problem.

In the light of this argument, two separate models are trained and their results are compared- the first is an ANN (**A**rtificial **N**eural **N**etwork) and the other is a LSTM-based framework.

* 1. **DATA EXPLORATION**

The dataset is obtained from **US** **G**eological **S**urvey **S**cience **D**ata Catalog (USGS SDC). Table 1 gives an overview of the dataset.

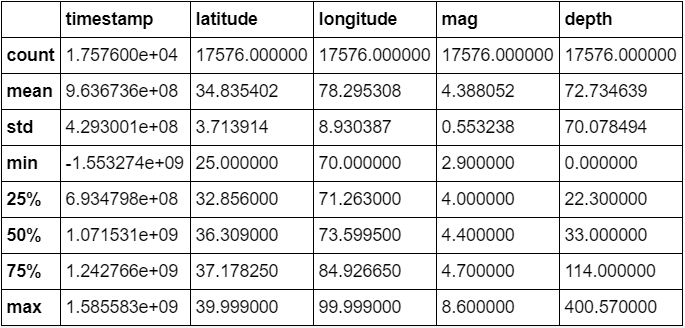


Table 1. Overview of the dataset

The salient features of the dataset are as follows:

* The dataset includes a total of **17576** earthquake events from 12-10-1920 to 30-03-2020.
* The latitude ranges from 25°N to 40°N, while the longitude ranges from 70°E to 100°E. This covers the Himalayan region.



Figure 3. Region under consideration

* The highest earthquake magnitude recorded was 8.6, while the lowest earthquake magnitude recorded was 2.9.

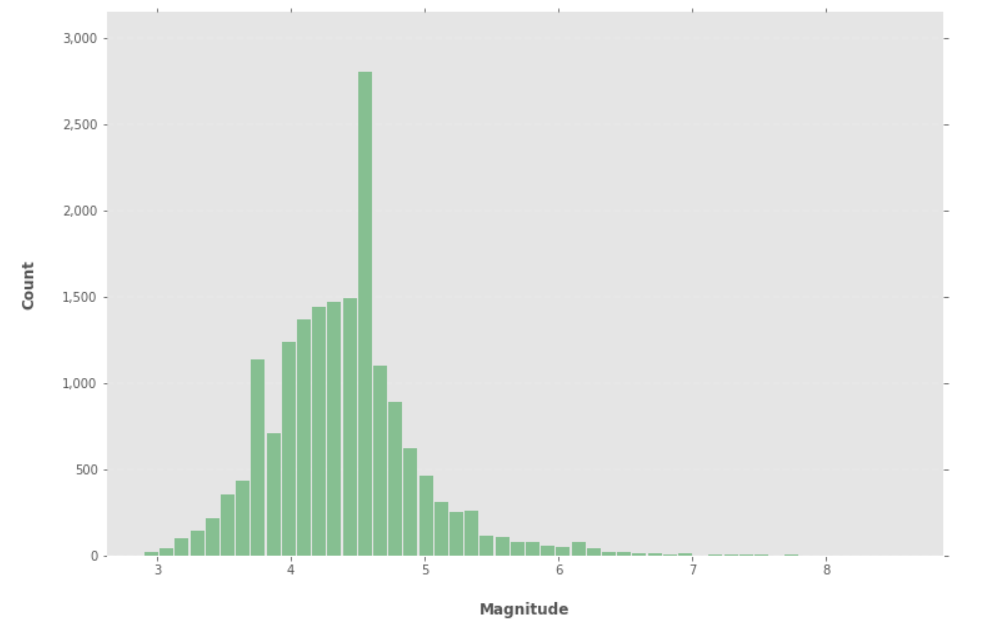


Figure 4. Histogram showing the number of earthquakes versus magnitude

* Figure 4, and a 3 quartile (75% quantile) value of 4.7 for the magnitude indicate the inherent skewness in the dataset.
  + 1. **DATA PRE-PROCESSING**

**The** dataset contains 17576 rows and 22 columns. Columns indicating the errors in the measurement are expected not to give any significant information with respect to our analysis, and are ignored for now. Since the type of each event is earthquake, this information is redundant and hence removed from the dataset. Further, we don’t require the information about the sources which have supplied this information (with almost all the cells filled with the entry “us”).

Thus, the new dataset contains only six columns: **latitude**, **longitude**, **date**, **place**, **depth** and magnitude **(mag)**, which are in accordance with several of the tabular datasets used in this field. Additionally, one might question the requirement of place as a feature when we already have latitude and longitude as features. So, in this study, “place” is just used as a marker; this attribute will **not** be used during analysis.

Thereafter, each entry in the **date** column is replaced with the corresponding timestamp for ease of subsequent data processing. Figure 5 shows two rows of our dataset after pre-processing.

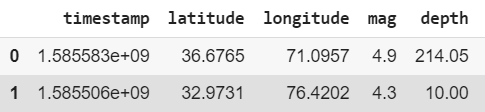


Figure 5. A subset of the dataset after pre-processing

* + 1. **FEATURE SELECTION**

For efficient training, it is a good practice to find out correlation among features. In case two or more variables are found to be highly correlated, we can choose only one feature among all the correlated variables for training our model and eliminate the rest. This generally improves the computational efficiency of the model without any compromise with the learning of the model.

Taking help of the Pearson’s correlation coefficient as the measure, Table 2 gives the correlation matrix obtained.

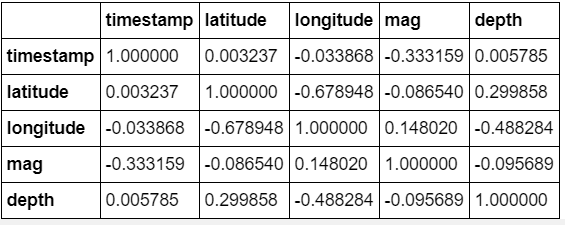


Table 2. Correlation Matrix

Following are the observations made:

* There is a strong correlation[[2]](#footnote-2) between **latitude** and **longitude**. It’s pretty intuitive to find such a correlation, for the Himalayan region, where a majority of the earthquakes are recorded in our region of our interest, witnesses a gradual fall in latitude up to Bhutan as one moves eastwards (See Figure 6).



Figure 6. The Himalayan belt

However, these two define a location on the map and hence both of them are necessary variables.

* There is a medium correlation[[3]](#footnote-3) between timestamp and magnitude. Undoubtedly, there is no variable among these two which can be eliminated.
* There is a small correlation[[4]](#footnote-4) between any other pair of variables.

Thus, there is no further elimination of the features to be passed to our models. From the next section onwards, the study focuses on the models used for prediction and forecasting of earthquakes.

* 1. **FORECASTING USING AN ANN**

An **A**rtificial **N**eural **N**etwork (ANN) is trained in the study for classification and regression purposes. As a classifier, it is used to predict the class of an earthquake event, details of which are provided in section 4.4.3. As a regressor, the ANN predicts the magnitude of the earthquake events, and the results are compared with the magnitudes of the earthquake events present in the test set. Appendix-1 gives an account of an artificial neural network.

* + 1. **NEED OF TRAINING AN ANN**

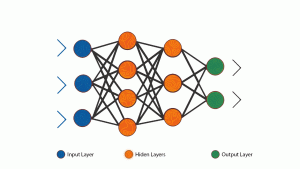
There was quite a good number of research papers found which reported an ANN being used for earthquake forecasting, claiming state-of-the-art accuracy in prediction. For instance, the paper “*Artificial neural networks for earthquake prediction using time series magnitude data or Seismic Electric Signals*” (Maria et al.) reports 58% accuracy for predicting major seismic events, and 80% accuracy of binary classification (earthquake v/s no earthquake).

Compared to LSTM or RNN, an ANN is mathematically much simpler to understand and much easier to train. Further, it has also been seen to perform at-par with deep learning algorithms in certain cases. Moreover, an ANN can be very easily used for regression as well as classification separately by making minor changes in the model.

Training an ANN can serve as a good **baseline model** for further study, especially for comparison of results obtained from other models. In the study, the results obtained from LSTM-based framework are compared with those of ANN and plausible conclusions are drawn.

* + 1. **ANN MODEL**

Figure 7 gives an outline of the ANN model used. It consists of an input layer (the input features: **four** in number- latitude, longitude, depth and timestamp), two hidden layers (each with **sixty** units) and an output layer (different for classification and regression).

****

60

60

4

Figure 7. Architecture of the ANN model. The numbers on the top of each layer indicate the number of units in the corresponding layer. Note that for classification, the number of output units is **4**, while for regression, there is **1** unit in the output layer.

* + 1. **ANN FOR CLASSIFICATION**

For the classification purpose, the entire dataset is divided into four classes depending upon the magnitude:

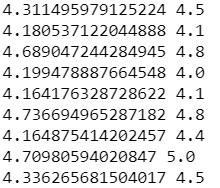
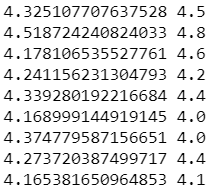
* Magnitude < 3.5: Class **0** (Zero)
* 3.5 ≤ Magnitude < 4.5 : Class **1** (One)
* 4.5 ≤ Magnitude < 6.0 : Class **2** (Two)
* Magnitude ≥ 6.0 : Class **3** (Three)

The **magnitude** column is removed from the input dataset for this experiment and used as ground truth for checking whether the model predicts the correct class or not. A soft-max layer is used as the output layer to predict one of the four classes, and the results are compared with the ground truth for subsequent back-propagation of the loss.

The model predicts a mean accuracy of **61.22%**, with a standard deviation of 1.84%. Even though the results indicate further tuning of hyper-parameters, especially the number of hidden layers and/or the number of units in each hidden layer, the accuracy is still better compared to hit-and-trial accuracy, where the chance of choosing the correct class is 25% (randomly choosing one class out of four).

* + 1. **ANN FOR REGRESSION**

The soft-max layer of the ANN in case of multi-class classification is replaced by a regression layer. The ground truth used for comparison is again the magnitude of the earthquake for each event. A similar procedure is followed as before for training the ANN. Figure 8 shows two snippets of the results obtained after regression. The first column gives the predicted magnitude, while the second column gives the actual magnitude of the corresponding earthquake data point.

8.a 8.b

Figure 8. Results of regression analysis

As can be seen from Figure 8, it seems like a good fit. Further, the **m**ean **s**quared **e**rror (MSE) turns out to be **0.2407**. This is indeed a very promising figure- this means that the predicted magnitude, on an average, differs from the actual magnitude by around 0.5 only.

This ANN model will be explored again after discussing the LSTM-based model.

* 1. **METRIC FOR COMPARISON**

So far, we’ve come across two metrics- accuracy and MSE. There are several other metrics used by machine learning practitioners- precision, recall, F1-score are a few of them. Usually, a particular metric is targeted during a research work and efforts are made to achieve state-of-the-art results with respect to that metric.

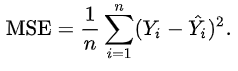
This section expresses an opinion on using accuracy for determining the reliability and robustness of the model. It must be made clear that the conclusion is just an opinion based on a few arguments; an extensive literature review needs to be done to come up with some more concrete arguments to cater to the problem of the appropriateness of a metric.

* + 1. **ACCURACY AS A METRIC**

Accuracy is perhaps one of the most common metrics used to evaluate the success of a model. For prediction of the magnitude and classification of the earthquake in this study, however, there are a few arguments which need to be pondered upon:

* Let’s say, our model predicts 4.4 as magnitude. However, our actual magnitude is 4.5. Is it considered a correct prediction? If yes, we’re certainly applying some threshold. The question still remains regarding what an appropriate threshold is. Say, we keep a threshold of 0.25. This means, in case if the actual magnitude is 5.25, any predicted value ranging from 5 to 5.5 is accepted to be a correct prediction. Now, the energy released corresponding to an earthquake of magnitude 5 is 2\*1012J while the energy released corresponding to an earthquake of magnitude 5.5 is 11.2\*1012J. This gives a clear idea of how much energy range we’re covering. And, 0.25 is not a figure which is too much; a threshold ranging from 0.2 to 0.3 is quite common in the state-of-the-art research works. However, this can be considered as a limitation from the side of the dataset- after all, we don’t have a spectrogram which gives such intricate details regarding the signal.
* Usually, the earthquakes which occur have magnitudes ranging from 3 to 5. Hence, the plot of the number of earthquakes versus magnitude is generally as skewed as the one shown in Figure 4. With regard to my dataset, I can simply predict “4.4” (the mean value) as magnitude with a threshold of 0.3 and achieve close to 50% classification accuracy. Now, a simple question which arises here is, is the model learning anything? Absolutely not. In fact, it’s not a model at all! If the dataset is even more skewed, the accuracy of this so-called “model” might even increase. This leads us to the conclusion that there should be some mechanism to deal with such skewness. This issue can be taken deeper and deeper without leading to any concrete conclusion.
* Again, the dataset does not provide any information about seismic noise. This is yet another important aspect of earthquake prediction- the model must be robust enough to distinguish between earthquake and seismic noise. It would not be considered a good model which gives an accuracy of 85-90% on some data but detects even seismic noise as earthquake.
  + 1. **METRIC FOR COMPARISON**

Since there has to be some metric chosen, the MSE (**M**ean **S**quared **E**rror) has been chosen as the metric for the rest of the study. Its formula is given by:



Here, Yi is the ground-truth value while Ŷi is the predicted value of the ith sample.

Visual inspection will be used for carrying out verification of results as and when required. This will be aided by plotting suitable graphs and curves.

* 1. **FORECASTING USING AN LSTM MODEL**

Since earthquakes exhibit temporal dynamic behaviour, application of an LSTM-based framework on earthquake dataset might yield favourable results in terms of prediction and forecasting. An LSTM cell is designed such that it is able to process a time series effectively. Appendix-2 gives an account of an LSTM unit.

* + 1. **LSTM MODEL**

Figure 9 gives an outline of the LSTM model used. The model has **4** LSTM layers arranged sequentially, followed by a fully connected layer (dense layer) for outputting the prediction. Each LSTM layer has **50** LSTM units. A dropout of 0.2 after each LSTM layer ensures regularisation, so as to avoid over-fitting.

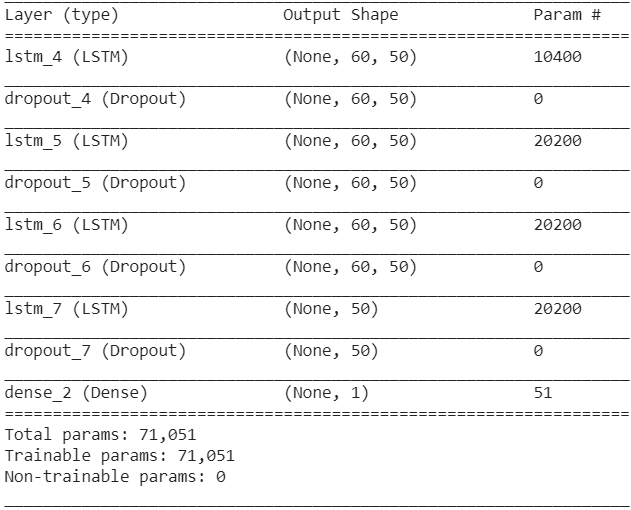


Figure 9. Architecture of the LSTM model

* + 1. **PRELIMINARY RESULT**

LSTM or RNN based models are generally very slow to train. So, in order to reduce the time required for training the model, the dataset is reduced so as to include only those earthquake events which have occurred after 01-01-2012. Further, the training set includes all the earthquake events before 2019 (till 31-12-2018), while the test set includes the earthquake events from 2019 onwards. Figure 10 gives a plot of the magnitude of earthquakes occurring as a function of time.

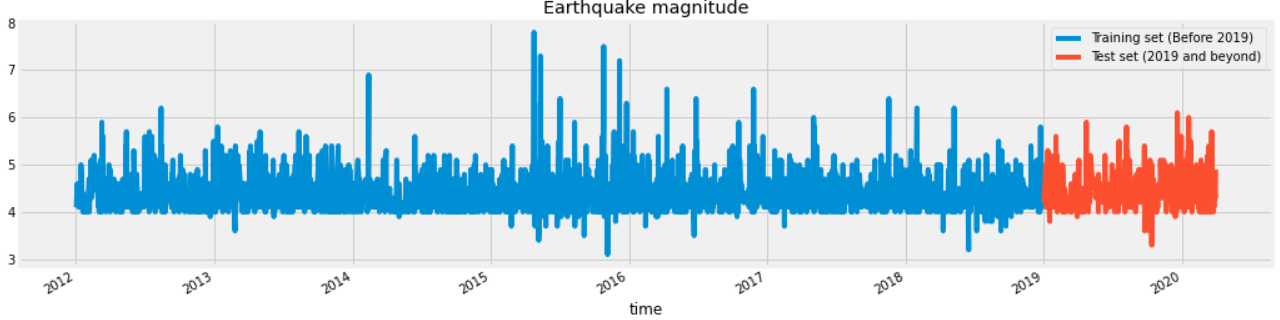


Figure 10. Plot of earthquake magnitude as a function of time

Surprisingly, it was observed that the model didn’t learn anything. Moreover, during the training phase, after ten epochs, the loss remained almost constant even after training the model for more number of iterations. Figure 11 gives a plot of the predicted magnitude and the real magnitude on the test set as a function of time. The MSE turns out to be a staggering **96.04** and indicates an immediate modification in the approach.

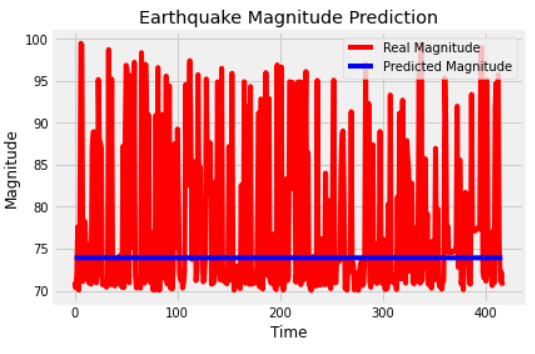


Figure 11. Results obtained on the test set

* + 1. **MODEL RETRAINING – A TRIAL**

The model is kept the same as before. Rather, the original dataset is now reduced to include only those rows where the earthquake magnitude lay between 3.5 and 5 only so as to nullify the effect of skewness. Out of the rows selected, 2109 rows were further selected depending upon proximity. Hence, the region of interest now predominantly reduced to Nepal-China border, thus reducing the spatial diversity in the dataset. The latest 500 rows are included in the test set, while the remaining dataset is used for the purpose of training.

This “trail” produced the result as shown in Figure 12. The MSE for the test set turns out to be **0.2048**, which is pretty good.

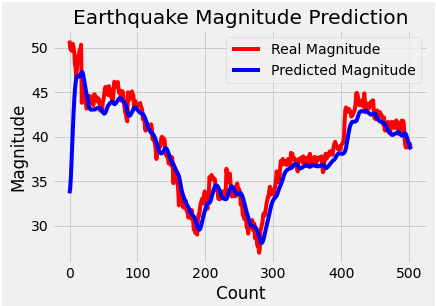


Figure 12. Result obtained after changing the approach. MSE = **0.2048**

* + 1. **LSTM MODEL – CUMULATIVE ANALYSIS**

An attempt of enforcing spatial dependency in the dataset comes with a **loss of temporal continuity**. Considering the Himalayan belt as a single entity, many data points are lost in the process. Coupled with choosing a specific magnitude range, it seems more like a model which **fits the metric** but is not generic. However, in general, works on earthquake forecasting seem to carry this limitation.

Questions may also be raised on the reliability of forecasting. The re-trained LSTM model is bound to predict majority of the earthquakes which would occur in the next few years from now near the Nepal-China border, owing to the restrictions imposed on the dataset. It’s very likely that the model has only learnt some function which fits for earthquakes in the magnitude range from 3.5 to 5 but may not forecast much devastating earthquakes at all. This problem needs to be addressed and attempts need to be made to reliably forecast earthquakes across all magnitudes.

Again, the question arises, is MSE an appropriate metric? This question will be dealt with in the next section.

1. **THE R2 SCORE**

In this question, a new metric is used to cater to the question asked at the end of the previous section. The results of ANN and LSTM-based approaches are compared and appropriate conclusions are drawn.

* 1. **THE NEED FOR A DIFFERENT METRIC**

As per the discussion so far, the ANN has not only produced comparable results to the re-trained LSTM model (in terms of the MSE) but has also not imposed any restriction whatsoever on the dataset. Thus, our ANN can be considered a better model in terms of adaptivity to generality. However, theoretically, under the assumption that our tabular data has some temporal pattern, LSTM should outperform all the other approaches.

An important observation comes from the fact that the predictions of our neural network are generally between 4 and 5 only. It seems like the hypothesis learnt by the model to minimize loss was severely affected by the skewness in the dataset, and even though the MSE is very low, the predictions cannot be called good at all. Figure 13 leads us to the conclusion that we need some other metric for comparison.

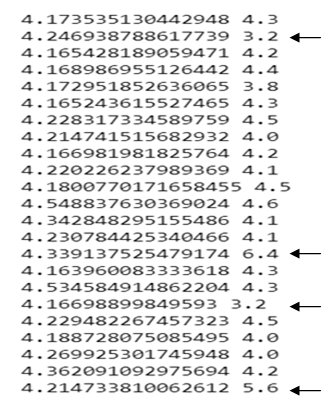
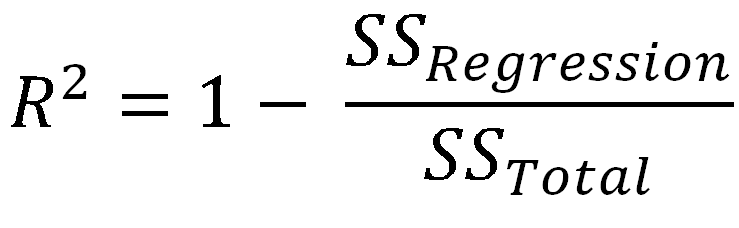


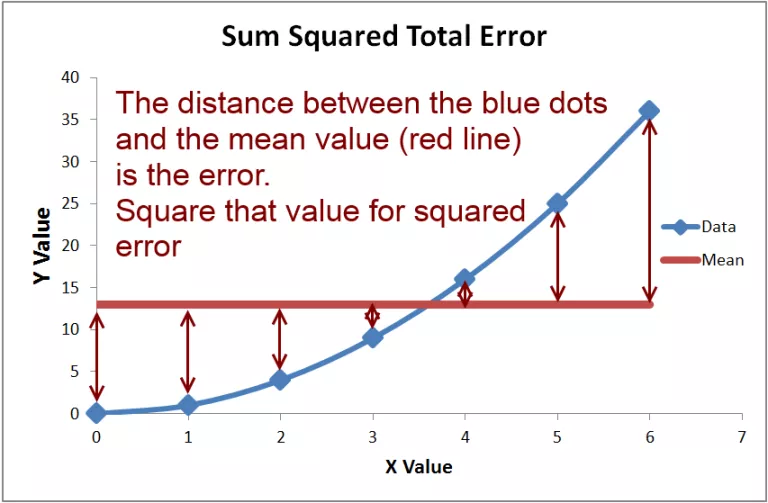
Figure 13. Result of the ANN. Note the arrows. As before, the first column refers to the predicted magnitude, while the second column refers to the actual magnitude.

* 1. **THE R2 SCORE- DEFINITION**

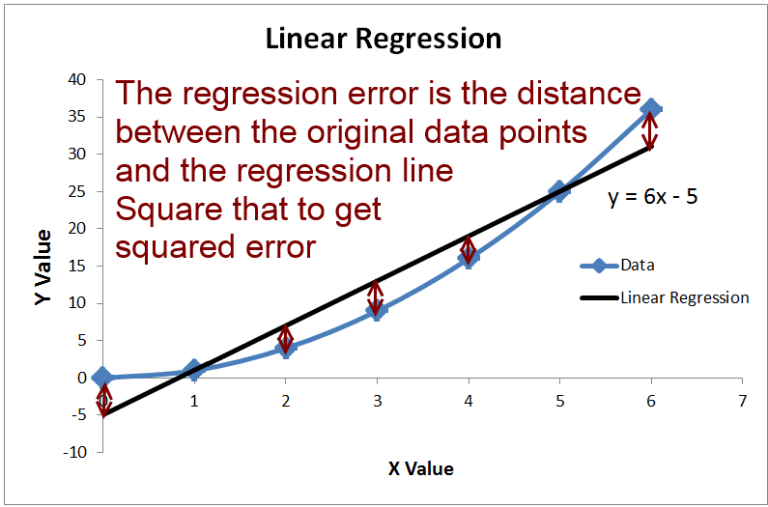
Also known as the coefficient of multiple determination, this score tells how much better the regression curve is than a simple horizontal line through the **mean** of the data. The formula for the R2 score is as follows:



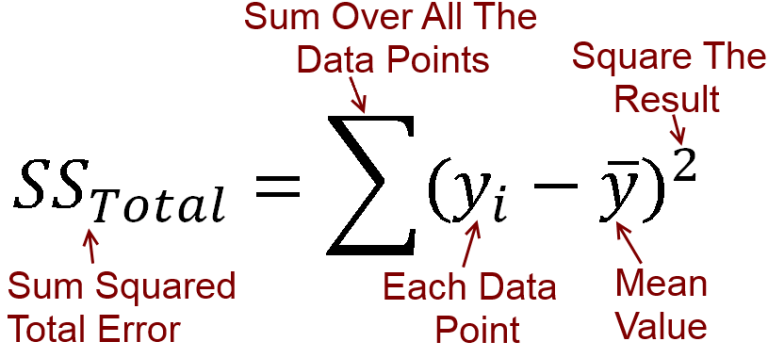
The following figures explain the meaning of the variables used in the formula[[5]](#footnote-5).



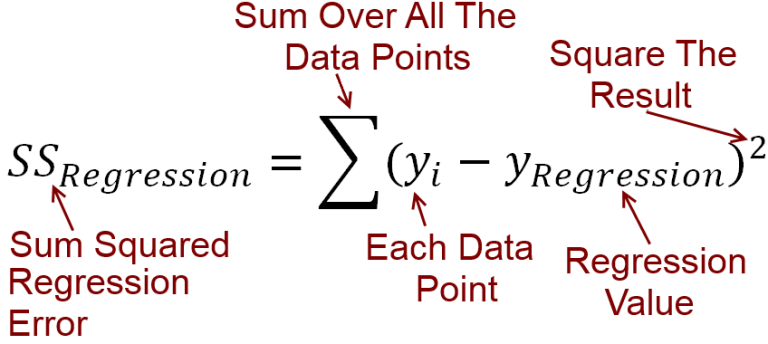
14.a



14.b



14.c



14.d

Figure 14. Explanation of the variables used in the formula for R2 score

* 1. **THE R2 SCORE- RESULTS**

In case of ANN, the average R2 score turns out to be **-0.471**. A negative R2 score simply means that the hypothesis is unable to make a better prediction compared to a hypothesis predicting just the mean value, and hence it is a clear example of under-fit. The more important thing here is, the prediction per sample is almost completely governed by the **timestamp** value, i.e. even if we remove all the other features (latitude, longitude and depth), we might expect a similar result.

In case of LSTM (the original model), the average R2 score turns out to be **-0.252**. Again, the score is negative, implying an under-fit. However, this time, the model learns a hypothesis which assigns weights to all the features rather than a single feature.

* 1. **THE R2 SCORE- CONCLUSION**

On an absolute scale, both the models do not seem impressive enough to produce reliable forecasts. Both the models need to be worked upon and the hyper-parameters need to be tuned to enhance the reliability of our predictions.

On a relative scale, ceteris paribus, LSTM clearly outperforms ANN by **46.5%**. Further, LSTM learns a model taking into consideration all the features, unlike ANN. Thus, LSTM holds an upper handover ANN for earthquake prediction and forecasting. This argument is supported by the underlying Mathematics of the two models, provided the analysis is upon a time-series data containing some pattern(s). Appendix-1 and Appendix-2 give an account of the Mathematics behind ANN and LSTM respectively.

1. **DIRECTIONS FOR FUTURE WORK**

The inability to generate an appropriate result necessitates the need to look for alternatives in addition to modifications in the existing models.

In section 4.2, the appropriateness of using CNN for processing tabular data was discussed. Despite not sounding logical, CNN-based approaches have been used on numerical data and achieved very accurate results. A CNN can not only be used for feature extraction; it can also help to create new features as well. The output feature vectors of the different inner layers of a CNN can be combined to form a new feature vector. This can be done for all the samples to obtain a feature matrix. This feature matrix can be used for further processing. An important advantage of such a kind of approach is that it imparts robustness to the model. This kind of approach finds extensive application in tracking.

In case of Computer Vision, particularly in the field of object detection and object tracking, associating the detections involves the use of an RNN-based framework. A few research works in this field have used an **E**cho **S**tate **N**etwork (ESN) instead of LSTM, Temporal Pooling or RNN for estimating the trajectory. With a sparsely connected hidden layer, it is computationally cheaper than LSTM. So, as another experiment, the LSTM cell in our case can be replaced with an ESN cell.

There still remains an absence of information regarding noise. So, an attempt can be made to convert data from spectrograms into a tabular data, which can also provide information regarding the frequency of signal, the magnitude of energy released etc. in addition to the magnitude of earthquake on the Richter scale.

And, finally, it still remains to be seen whether the catalogue procured through seismological centres holds some temporal pattern or not. This means, even though an LSTM model might fetch very good results on the events seen, it might fare poorly on time-series forecasting, especially for future events.

**REFERENCES**

R. Allen, Automatic phase pickers: their present use and future prospects, Bulletin of the Seismological Society of America 72 (1982) S225-S242.

M. Withers, R. Aster, C. Young, J. Beiriger, M. Harris, S. Moore, J. Trujillo, A comparison of select trigger algorithms for automated global seismic phase and event detection, Bulletin of the Seismological Society of America 88 (1998) 95-106.

S. J. Gibbons, F. Ringdal, The detection of low magnitude seismic events using array-based waveform correlation, Geophysical Journal International 165 (2006) 149-166.

R. J. Skoumal, M. R. Brudzinski, B. S. Currie, J. Levy, Optimizing multi-station earthquake template matching through re-examination of the Youngstown, Ohio, sequence, Earth and Planetary Science Letters 405 (2014) 274-280.

S. A. Barrett, G. C. Beroza, An empirical approach to subspace detection, Seismological Research Letters 85 (2014) 594-600.

H. M. Benz, N. D. McMahon, R. C. Aster, D. E. McNamara, D. B. Harris, Hundreds of earthquakes per day: The 2014 Guthrie, Oklahoma, earthquake sequence, Seismological Research Letters 86 (2015) 1318-1325.

C. E. Yoon, O. OReilly, K. J. Bergen, G. C. Beroza, Earthquake detection through computationally efficient similarity search, Science advances 1 (2015) e1501057.

G. Madureira, A. E. Ruano, A neural network seismic detector. IFAC Proceedings Volumes 42 (2009) 304–309.

J. Wang, T. Teng, Artificial neural network-based seismic detector. Bulletin of the Seismological Society of America 85 (1995) 308–319.

T. Perol, M. Gharbi, M. Denolle, Convolutional neural network for earthquake detection and location. Science Advances 4 (2018) 1700578.

H. S. Kuyuka, O. Susumua, Real-Time Classification of Earthquake using Deep Learning. Procedia Computer Science 140 (2018) 298–305

S. M. Mousavi, W. Zhu, Y. Sheng, G. C. Beroza, CRED: A Deep Residual Network of Convolutional and Recurrent Units for Earthquake Signal Detection. Scientific Reports (2019).

M. Ibrahim, J. Park, N. Athens, Earthquake warning system: Detecting earthquake precursor signals using deep neural networks.

<https://cs230.stanford.edu/projects_spring_2018/reports/8291012.pdf>

## M. Moustra, M. Avraamides, C. Christodoulou, Artificial neural networks for earthquake prediction using time series magnitude data or Seismic Electric Signals. [Expert Systems with Applications](https://www.sciencedirect.com/science/journal/09574174), [Volume 38, Issue 12](https://www.sciencedirect.com/science/journal/09574174/38/12) (2011) 15032-15039.

## APPENDIX-1

## Artificial Neural Network (ANN)

## An artificial neural network (ANN) is a computing system inspired from the biological neural network- the network of neurons that constitutes animal brain. An ANN is a collection of connected units known as neurons. Typically, these neurons are arranged into layers.

## It’s important to understand what a logistic regression unit or a single neuron is. A neural network is just a “networked set” of such logistic regression units.

## We start with the cost function for logistic regression, J(θ).

## 

## Where:

## 

## This can be simplified to obtain:

## 

## Here, hθ(x(i)) is the predicted value while y(i) is the ground truth (actual) value for the ith sample.

## Now, we pass our input to the logistic regression unit and get the loss as J(θ). The next task is, in layman terms, to tell the model about the error obtained. Well, this is what back-propagation achieves, which updates the weights after completing the iteration (θ in our case). A common way of back-propagating the loss is the gradient descent algorithm, which back-propagates the loss after every iteration so as to minimise the loss as we iterate over the samples. Following is the gradient descent algorithm:

## 

## This can be simplified to obtain the gradient descent algorithm for our logistic regression unit:

## 

## The main aim of any ML algorithm, in general, is to minimise the loss function. Updation of weights in our case achieves this feat, and by applying forward and backward propagation repeatedly, we are said to “train” our logistic regression unit. In a similar manner, we can train our ANN: the underlying computation does not change, just that the same computation is carried out on every neuron.

## APPENDIX-2

## LONG SHORT TERM MEMORY (LSTM)

## LSTMs are a special kind of RNN which are capable to learn long-term dependencies. They are an improvement over a Recurrent Neural Network (RNN) in the sense that they avoid the vanishing and exploding gradient problems prevalent in RNN-based frameworks. LSTMs find major application in sequence modelling.

## The repeating module in an LSTM has four interacting layers. See Figure 15.

## 

## Figure 15. The internal structure of an LSTM unit (see the figure in the middle)

## STEP-BY-STEP WALKTHROUGH THROUGH LSTM[[6]](#footnote-6)

## a)

## b)

## c)

## d)

1. Implementation based on <https://github.com/smousavi05/CRED.git> [↑](#footnote-ref-1)
2. Pearson’s correlation coefficient value lies between ±0.5 and ±1 [↑](#footnote-ref-2)
3. Pearson’s correlation coefficient value lies between ±0.3 and ±0.5 [↑](#footnote-ref-3)
4. Absolute value of Pearson’s correlation coefficient value lies below 0.3 [↑](#footnote-ref-4)
5. Figures retrieved from <https://stats.stackexchange.com/questions/183265/what-does-negative-r-squared-mean> [↑](#footnote-ref-5)
6. Figures taken from <https://colah.github.io/posts/2015-08-Understanding-LSTMs/> [↑](#footnote-ref-6)