

Predicting the Accuracy of Early-est Earthquake Magnitude Estimates with an LSTM Neural Network: A Preliminary Analysis

Massimo Nazaria

Istituto Nazionale di Geofisica e Vulcanologia (INGV)
Via di Vigna Murata 605, 00143 Roma, Italy

February 26, 2021

Abstract

This report presents a preliminary analysis of an LSTM neural network designed to predict the accuracy of magnitude estimates computed by Early-est during the first minutes after an earthquake occurs. Data and code are available at <https://github.com/massimo-nazaria/lstm>.

1 Introduction

Early-est is a software system that locates earthquakes on a global scale and computes their characterization parameters by using the data provided by seismic stations in real time [9][2]. Its results are the input for an automatic procedure that first inspects the computed earthquake parameters, then evaluates the possibility that a tsunami was generated, and finally prepares the text of the first message to be issued by the tsunami warning system managed by CAT-INGV. The position and size of the earthquake are recalculated at one-minute intervals, which makes them more and more accurate as seismic waves reach stations further away from the hypocenter and thus additional data becomes available. The ability to predict the accuracy of magnitude estimates during the first minutes since the occurrence of an earthquake would enable us to obtain the final magnitude in advance. While it is impractical to involve traditional programming to develop a software tool that computes such predictions, machine learning presents us with the possibility to attempt modeling a neural network which independently learns how to predict the accuracy of estimates by examining historical data without explicit instructions by the programmer. Unfortunately, we cannot assume that machine learning is a viable approach to our problem, as the data at our disposal may be unsuitable or insufficient for the automatic training of a neural network. Furthermore, to the best of our knowledge there are no neural network models in the literature we can adapt to our problem (see, e.g., [8] and references therein). In this report we present a preliminary analysis of an LSTM neural network that attempts to predict the accuracy of magnitude estimates computed by Early-est during the first minutes after the occurrence of an earthquake.



1.1 Recurrent Neural Networks and LSTMs

Recurrent Neural Networks (RNNs) are designed to selectively recognize patterns within input data which are organized as time series by inferring dependencies among subsequent data timesteps. RNNs are able to accomplish this with the use of a sort of internal memory which gets updated at each timestep being processed in order to keep track of meaningful information over the entire sequence of input. Nevertheless, the fact that RNNs repeatedly update their internal memory makes these networks suffer from the well known vanishing/exploding gradient problems [5][6]. In short, RNNs exhibit the tendency to overwrite their internal memory thus losing the ability to remember useful information from past data timesteps. On the other hand, Long Short-Term Memory (LSTM) networks are a kind of RNN which deals with the aforementioned problems with the use of a more complex internal architecture, that makes the network able to decide whether information at each timestep are going to be remembered or forgotten [7]. This enables LSTMs to have greater control over their internal memory updates in order to effectively infer dependencies among consecutive data timesteps thus providing better results. The fact that Early-est historical data consist of sequences of parameters computed at consecutive minutes makes LSTMs suitable for our needs. Namely, our LSTM should be able to learn how to predict the accuracy of magnitude estimates at a given minute on the basis of earthquake parameters computed at previous minutes.

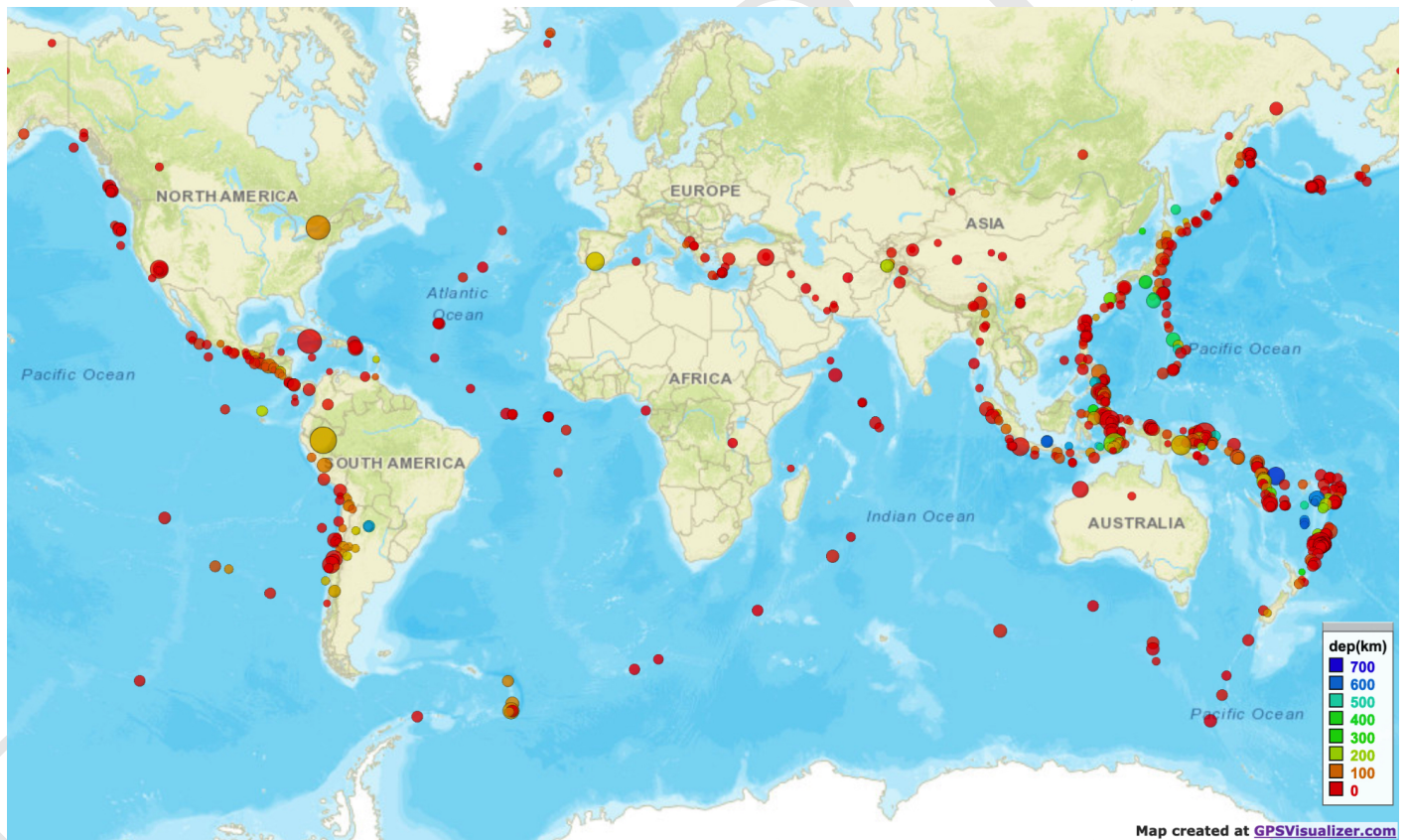


Figure 1: Location of the 608 earthquakes used in this work. The size of the circles is proportional to the magnitude of the earthquakes; the color depends on the depth of the hypocenter.

Name	Type	Description
loc_err_h	Double	Ellipsoid horizontal error
loc_err_z	Double	Ellipsoid vertical error
loc_err_resid	Double	RMS of the residuals
loc_num_st	Integer	Number of stations used for location
loc_min_dist	Double	Distance of the nearest station
loc_avg_dist	Double	Average distance of stations
loc_std_dist	Double	Standard deviation distance of stations
loc_1st_azim_gap	Double	Primary azimuthal gap
loc_2nd_azim_gap	Double	Secondary azimuthal gap
mag	Double	Estimated magnitude of type Mwp
mag_err	Double	Magnitude error
mag_num_st	Integer	Number of stations used for mag (\leq loc_num_st)

Table 1: Early-est earthquake parameters used in this work.

2 Data Preparation

In the database with historical data by Early-est, the 12 earthquake parameters used in this work, which are described in Table 1, are associated with both the event IDs they refer to as well as the minutes they were computed at. Minutes are represented as integers and are relative to the time of arrival of the very first seismic data of their corresponding events. These 12 parameters are intended to provide us with useful information about the accuracy of both location and magnitude estimates (e.g. RMS of the residuals, number of stations used) as well as the geometry of those stations where the seismic data came from (e.g. average distance of stations, primary and secondary azimuthal gap). Studies showed how these parameters are correlated to the accuracy of the final estimates of both location and magnitude, and how more accurate locations result into more accurate magnitude estimates (see, e.g., [3] and citations therein).

Early-est Version	1.2.4
Number of Events	608
Time Interval of the Events	2019/04/15–2020/01/30
Minutes per Event	1, 2, 3, ..., 16
Magnitudes at Minutes 16	≤ 5
Range of Magnitudes	[4.3, 7.8]
NULL Values	None

Table 2: Information about the initial dataset.

2.1 Initial Dataset

Our initial dataset consists of a selection of historical events, which are summarized in Table 2. The data were computed by Early-est version 1.2.4 and comprise the 608 events in Figure 1 located all over the world in the period of time between 15 April 2019 and 30 January 2020. For each event in the dataset, there are the values of the 12 parameters in Table 1 that were computed at minutes from 1 to 16. The magnitude estimates at minutes 16 are all greater than or equal to 5. Finally, the range of the magnitudes is [4.3, 7.8] and no NULL parameter values are present.

2.2 Definition of Accuracy

Following a preliminary analysis of the data, we select the magnitude estimates computed at minutes 16 as the most accurate ones and define the accuracy of a given estimate mag_t^e computed at the t -th minute of an event e with the following formula:

$$accuracy(mag_t^e) := \frac{mag_t^e - mag_{16}^e + 1}{2}.$$

The accuracy of mag_t^e is the difference $mag_t^e - mag_{16}^e$ between the magnitudes estimated at the minutes t and 16 of the same event e . In order to make sure that all the accuracies fall into the range $[0, 1]$, we add 1 to the difference and divide the result by 2.

Figure 2 shows the magnitude accuracies of all the 608 events at minutes from 1 to 16. The increasing availability of data used by Early-est to recompute its estimates makes the latter more and more accurate as the minutes go by.

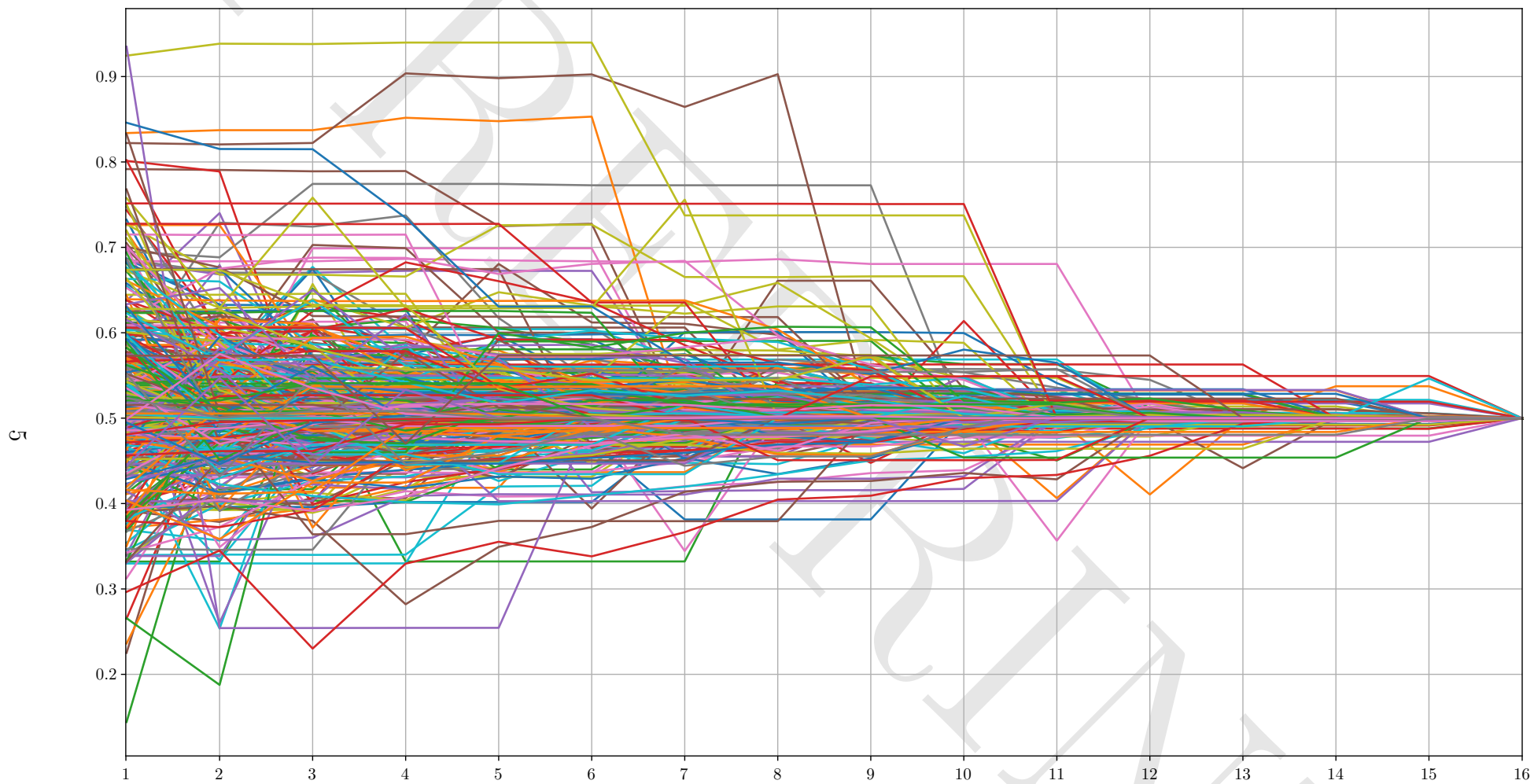


Figure 2: Magnitude accuracies of the 608 selected events at minutes from 1 to 16.

3 Feature Extraction and Labeling

The training samples for our neural network consist of pairs with inputs and desired outputs we refer to as features and labels, respectively. Firstly, let us define $Par_t^e(p)$ as the value of the parameter p of a given event e at minute t . Secondly, let $Par_t^e := (Par_t^e(\text{loc_err_h}), \dots, Par_t^e(\text{mag}), \dots)$ be defined as the sequence that contains values of all the 12 parameters seen in Table 1 of a given event e at minute t . Finally, let us define $Par^e := (Par_1^e, Par_2^e, \dots, Par_N^e)$ as the sequence of N sequences Par_i^e , with $1 \leq i \leq N$, and $N := 16$.

3.1 Features

For each event e , we prepare input features from Par^e by computing a number of $N - K + 1$ subsequences of length K , with $K := 2$, in this way:

$$\begin{aligned} X_1^e &:= (Par_1^e, Par_2^e, \dots, Par_K^e), \\ X_2^e &:= (Par_2^e, \dots, Par_K^e, Par_{K+1}^e), \\ &\vdots \\ X_{N-K+1}^e &:= (Par_{N-K+1}^e, Par_{N-K+2}^e, \dots, Par_N^e). \end{aligned}$$

3.2 Labels

For each feature X_i^e , we prepare the corresponding label Y_i^e , with $1 \leq i \leq N - K + 1$, as follows:

$$\begin{aligned} Y_1^e &:= \frac{Par_K^e(\text{mag}) - Par_N^e(\text{mag}) + 1}{2}, \\ Y_2^e &:= \frac{Par_{K+1}^e(\text{mag}) - Par_N^e(\text{mag}) + 1}{2}, \\ &\vdots \\ Y_{N-K+1}^e &:= \frac{Par_N^e(\text{mag}) - Par_N^e(\text{mag}) + 1}{2} = \frac{0+1}{2} = 0.5. \end{aligned}$$

As a result, we obtain the following 15 samples (X_i^e, Y_i^e) for each event e , with $1 \leq i \leq 15$:

$$\begin{aligned} (X_1^e, Y_1^e), \text{ where } X_1^e &:= (Par_1^e, Par_2^e) \text{ and } Y_1^e := \frac{Par_2^e(\text{mag}) - Par_{16}^e(\text{mag}) + 1}{2}, \\ (X_2^e, Y_2^e), \text{ where } X_1^e &:= (Par_2^e, Par_3^e) \text{ and } Y_2^e := \frac{Par_3^e(\text{mag}) - Par_{16}^e(\text{mag}) + 1}{2}, \\ &\vdots \\ (X_{15}^e, Y_{15}^e), \text{ where } X_{15}^e &:= (Par_{15}^e, Par_{16}^e) \text{ and } Y_{15}^e := \frac{Par_{16}^e(\text{mag}) - Par_{16}^e(\text{mag}) + 1}{2}. \end{aligned}$$

Each feature X_i^e consists of parameter values computed at 2 consecutive minutes, whereas each label Y_i^e represents the accuracy of the last magnitude estimate in its corresponding feature.

3.3 Data Scaling

In order to increase the chances of a successful training outcome, we scale parameter values in the training samples to the range $[0, 1]$ by using the min-max scaling approach with the Scikit-learn software library [10]. This data scaling operation makes data samples more digestible to the training algorithm, which is responsible for teaching the neural network how to predict magnitude accuracies. More precisely, data scaling operations are generally known to improve the effectiveness of gradient descent-based training algorithms, which optimize the weights of neural network models in order to minimize the prediction error by automatically choosing appropriate hyper-parameters (e.g. the learning rate) that are crucial to a fast and optimal convergence towards the final trained network.

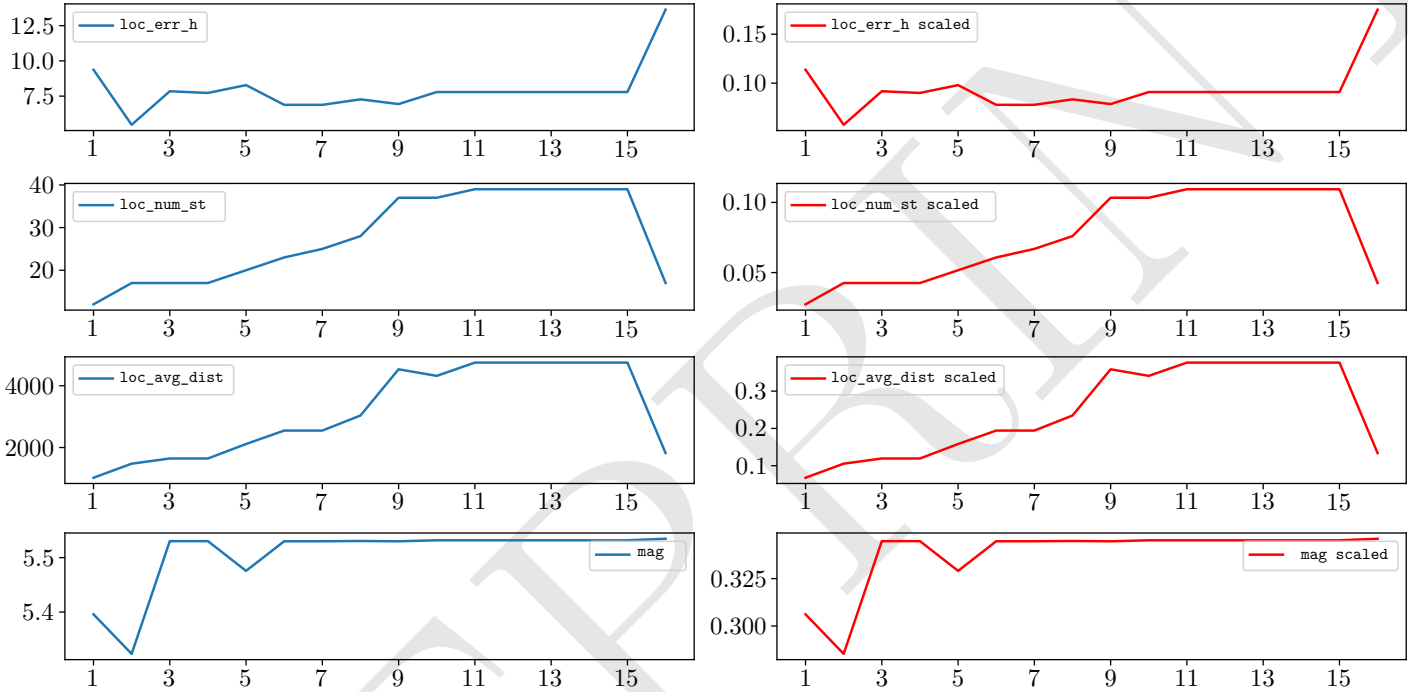


Figure 3: Original data (right) and scaled data (left).

Figure 3 shows how the trends of parameter values remain unchanged after the scaling process, which means the information about how the parameters develop over time is totally preserved.

4 Model Definition and Training

Table 3 illustrates a layer-by-layer definition of our neural network model. First, there are two stacked LSTM layers with rectified activation functions (ReLUs) and input-output dimensions of $(K, P)-(K, 20)$ and $(K, 20)-(10)$, respectively. $K := 2$ is the number of consecutive minutes per input feature, and $P := 12$ is the number of parameter values per minute. Finally, there is a fully connected layer (i.e. Dense) with a sigmoid activation function with input-output dimensions of $(10)-(1)$. Both modeling and training processes are performed with Tensorflow and Keras software libraries [1][4].

Layer	Type	Act. Func.	Input Dim.	Output Dim.
1	LSTM	ReLU	(K, P)	$(K, 20)$
2	LSTM	ReLU	$(K, 20)$	(10)
3	Dense	Sigmoid	(10)	(1)

Table 3: Layers of the neural network model.

Training Set	70% of the samples
Test Set	30% of the samples
Optimizer Algorithm	Adam
Loss Function	Mean squared logarithmic error
Batch Size	16
Validation Set	25% of the training set
Number of Epochs	100

Table 4: Initial setup of the training process.

4.1 Training Process

The initial setup of the training process is presented in Table 4. The training set consists of 70% of the samples in the initial dataset shown in Section 3, while the remaining 30% of the samples represent the test set, which is used after the training process in order to evaluate how the final trained model performs on never-before-seen data. The training algorithm is the Adam optimizer, which iteratively updates the weights of the model in order to reduce the outcomes of the loss function, which consists of the mean squared logarithmic error between the predicted outputs and the actual outputs. The batch size of 16 tells the algorithm how many samples it must process before updating the weights, while the number of epochs of 100 indicates how many times the algorithm must inspect the entire training dataset by randomly drawing batches of samples. At the beginning of each epoch, the algorithm randomly populates a validation set with 25% of the training set in order to show us a comparison of the training loss and the validation loss in Figure 4, which provides us with useful information on how our model behaves during the training process.

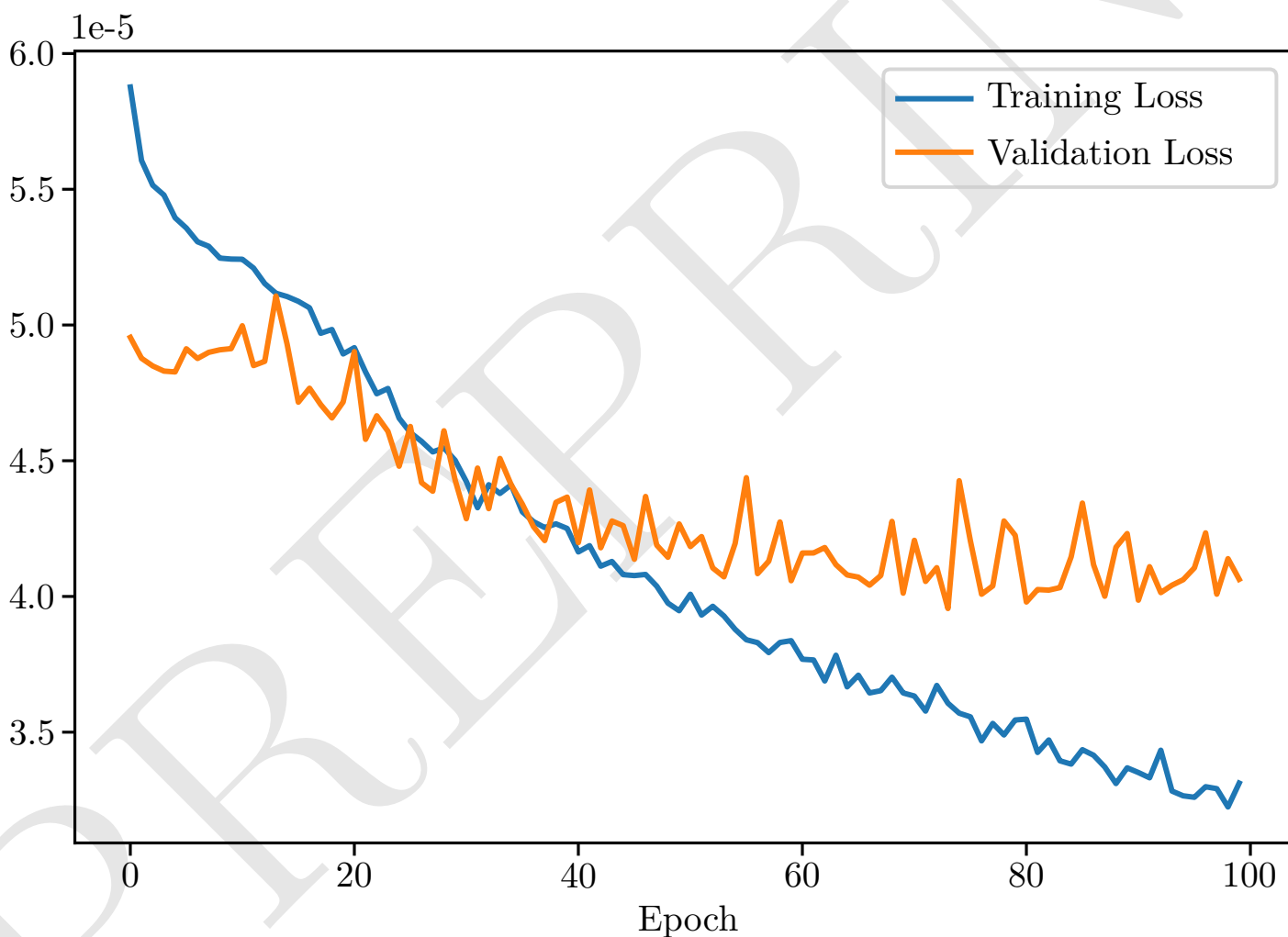


Figure 4: Training and validation loss during the training process.

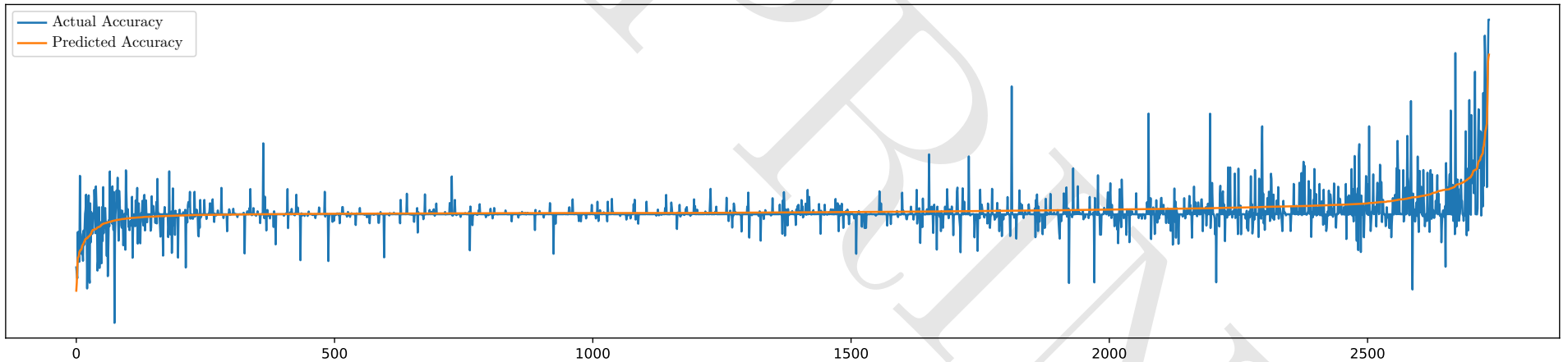
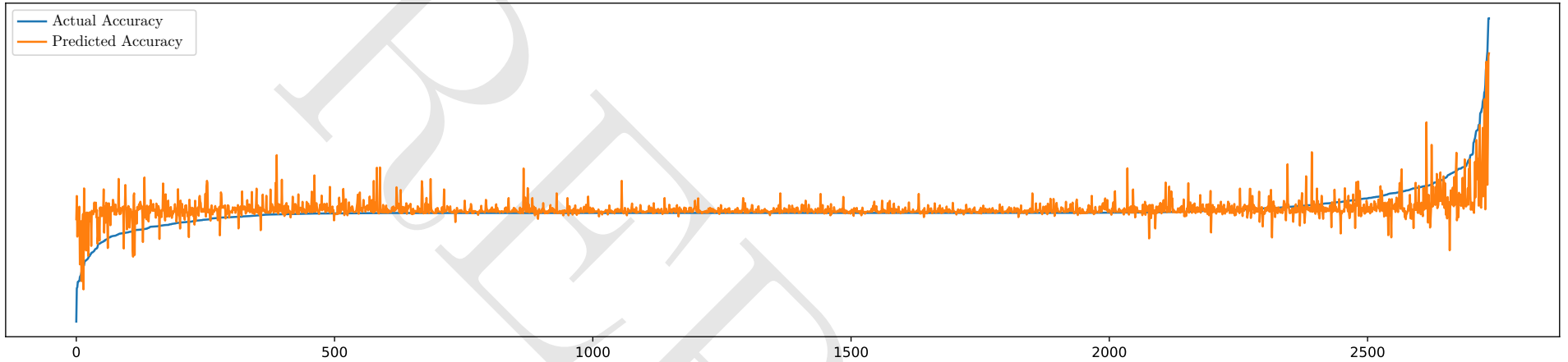


Figure 5: Results on the *test set* ordered by actual accuracy (above) and by predicted accuracy (below).

5 Evaluation

Plots in Figure 5 show results on the test set by comparing the actual accuracy (in blue) with the accuracy predicted by the model (in orange). The same results are ordered by actual accuracy and by predicted accuracy. In a nutshell, where the orange points coincide with the blue ones that means our trained model was able to perfectly predict the accuracy of the magnitude estimates by Early-est (i.e. the actual accuracy). These plots provide us with information on how our trained neural network model performs on never-before-seen data.

Results show the model exhibits the tendency to correctly predict errors whenever the magnitudes are under- or overestimated (left-hand and right-hand side in the plots, respectively).

6 Conclusion

In spite of the limited amount of historical data in this initial analysis, results show the designed network exhibits the tendency to correctly predict errors whenever the magnitudes are under- or overestimated. This fact indicates that the historical data are suitable for the automatic training of the neural network presented in this report.

Acknowledgments

This work would not have been possible without the help of Dr. Franco Mele, Dr. Stefano Lorito and the Tsunami Alert Center group at INGV.

References

- [1] Abadi M., Barham P., Chen J., Chen Z., Davis A., Dean J., Devin M., Ghemawat S., Irving G., Isard M. and others, (2016). *Tensorflow: A system for large-scale machine learning*. In 12th USENIX Symposium on Operating Systems Design and Implementation, OSDI 16, pp. 265-283
- [2] Bernardi F., Lomax A., Michelini A., Lauciani V., Piatanesi A. and Lorito S., (2015). *Appraising the Early-est earthquake monitoring system for tsunami alerting at the Italian Candidate Tsunami Service Provider*. Nat. Hazards Earth Syst. Sci., 15, pp. 2019-2036, <https://doi.org/10.5194/nhess-15-2019-2015>
- [3] Bondár I., Myers S.C., Engdahl E.R. and Bergman E.A., (2004). *Epicentre accuracy based on seismic network criteria*. Geophysical Journal International, Volume 156, Issue 3, March 2004, pp. 483-496, <https://doi.org/10.1111/j.1365-246X.2004.02070.x>
- [4] Chollet F. and others, (2015). *Keras*. <https://keras.io>.
- [5] Hochreiter S., (1991). *Untersuchungen zu dynamischen neuronalen Netzen*. Diploma Thesis, Technical University Munich, Institute of Computer Science.
- [6] Hochreiter S., (1998). *The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions*. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 06 (2), pp. 107-116, doi:10.1142/S0218488598000094

- [7] Hochreiter S. and Schmidhuber J., (1997). *Long short-term memory*. Neural Computation, 1997 Nov 15, 9(8), pp. 1735-80, doi:10.1162/neco.1997.9.8.1735
- [8] Kong Q., Trugman D.T., Ross Z.E., Bianco M.J., Meade B.J. and Gerstoft P., (2018). *Machine Learning in Seismology: Turning Data into Insights*. Seismological Research Letters, 90 (1), pp. 3-14, doi: <https://doi.org/10.1785/0220180259>
- [9] Lomax A. and Michelini A., (2009). Tsunami early warning using earthquake rupture duration. Geophysical Research Letters, 36, L09306, doi:10.1029/2009GL037223
- [10] Pedregosa F., Varoquaux G., Gramfort A., Michel V., Thirion B., Grisel O., Blondel M., Prettenhofer P., Weiss R., Dubourg V., Vanderplas J., Passos D., Brucher M., Perrot M. and Duchesnay E., (2011). *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 12, pp. 2825-2830