Comparison of neural network models with GRU and LSTM layer for earthquake prediction⋆

Wiktoria Plechta

Faculty of Applied Mathematics, Silesian University of Technology, Kaszubska 23, 44100 Gliwice, POLAND

Abstract
The prediction task is an important element of research as it allows for forecasting future events. In the case of earthquakes, it is possible to make such a prediction based on specific attributes of historical data. In this paper, we compare two solutions in the field of neural networks that allow for such an analysis. More specifically, we model two recurrent networks by using the LSTM and GRU layers. The comparison is based on sequences of one-element and two-element data to determine which model is more accurate. Also, such methodology allows us to pay attention to the amount of sequence data needed to train the recurrent classifier for earthquake prediction tasks.

Keywords
rnn, lstm, gru, earthquake prediction

1. Introduction

Data prediction is an important tool that uses algorithms and mathematical models to predict a specific event. For this purpose, historical data is used, i.e. a record of information that recorded such phenomena. There is a great need to build such solutions because they enable their use in a wide range of applications. An example is weather prediction, which is based on the analysis of historical data and the current weather conditions. Please note that weather models also use other data such as solar irradiation [1]. Another example is the construction of models that predict the dynamics of the development of various diseases. An example is the analysis of COVID-19 solutions using the computational model [2], or Markov chains for epidemic simulations [3]. Medical solutions also show the potential to predict the occurrence of various diseases [4]. An interesting approach is also the use of various methods such as k nearest neighbors, and decision trees, to predict communication customer churn [5]. Huge technological development also allows for the automation of many tasks. One such example is predicting the throughput of autonomous guided vehicles [6].

The construction of predictive systems allows for the possibility of building solutions using artificial intelligence methods and other techniques indicating the possibilities of presentation and decision-making. An example is a hybrid approach to analyzing many different techniques [7]. There are also ranking methods such as weighting techniques [8]. Attention should also be paid to the time analysis of various objects, an example of which is the change in water level [9].

The authors proposed using segmentation tools and edge analysis through the use of graphics processing techniques.

The most commonly used prediction techniques include recurrent neural networks. An example is the construction of a model of such a network for time series prediction [10]. Recurrent neural networks are quite often built based on recurrent layers. Currently, there are two main ones: Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU). The difference between them is based on the goal structure, whereas GRU has fewer of them. An example of the use of GRU gates is the development of aircraft assembly technology [11], or coal heading temperatures [12]. Long-term prediction for energy consumption can be performed by LSTM layers with multi-attention mechanism [13], or photovoltaic energy forecasting [14].

In this paper, we want to compare the two most popular recursive layer architectures are LSTM and GRU. For this purpose, a publicly available database with significant earthquake predictions was used. Based on this database, a recurrent neural network model and one/two-element segmentation samples were built.

2. Neural network architecture

Recurrent neural networks are extremely effective and capable of learning complicated issues. However, simple RNN is known for having problems with vanishing and exploding gradients. That is why two recursive architectures are more common in this paper: LSTM and GRU. While there is a lot of research comparing these two methods, this work focuses on forecasting earthquakes, which are more random and, therefore not so commonly used with these networks. Predicting earthquakes is a complex challenge because seismic events are inherently unpredictable and influenced by many factors. While GRU and LSTM networks excel at capturing patterns in sequential data over time their comparison based on this database may bring riveting conclusions.

2.1. Long Short-Term Memory

Long Short-Term Memory cells consist of two states: cell state and hidden state. The first one is responsible for encoding information from all previous steps and extracting important features. The other one focuses on the latest time step, giving a prediction for the next time step. However, the prediction is encoded and therefore is not the same as the output state. Another important factor, that is a must to describe when it comes to LSTM are three gates: forget gate, input gate and output gate. Forget gate filters data from the cell state - the closer to 1, the more important the information is and in reverse - the closer to 0, the more of the information is being forgotten. Similar comes to the input gate, except it is used with tanh function to charge of adding information to the cell state. In turn, the output gate decides what the next hidden state will be.

The operation of an LSTM network can be described using several key equations.

The forget gate similar to the update gate, f_t is also computed using a sigmoid activation function.

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \tag{1}$$

Where:

- x_t - input at time t,
- h_{t-1} - previous hidden state,
- W_{xf}, W_{hf}, and b_f - weights and biases for the forget gate,
- σ - sigmoid function.

The input gate is typically computed using a sigmoid activation function, resulting in values between 0 and 1.

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \tag{2}$$

Where:

- W_{xi}, W_{hi}, and b_i - weights and biases for the update gate.

Finally, the output gate (o_t) decides which information to pass on to the output. This is determined by the current hidden state (h_t), which is filtered through the output gate.

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \tag{3}$$

Where:

- W_{xo}, W_{ho}, and b_o - weights and biases for the output gate.

The cell state (C_t) update is then calculated. This involves updating the cell state C_{t-1} based on the result of the update gate i_t, and removing unnecessary information based on the result of the forget gate f_t.

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \tag{4}$$

Where:

- W_{xc}, W_{hc}, and b_c - weights and biases for the cell state update,
- $\tanh$ - hyperbolic tangent function.

The final hidden state h_t is calculated by updating the cell state C_t based on the output gate result o_t, resulting in the LSTM network output.

$$h_t = o_t \cdot \tanh(C_t) \tag{5}$$

2.2. Granted Recurrent Unit

Granted Recurrent Unit has only two gates and does not contain the Cell State. All of the data goes through the hidden state, where the reset gate determines which information from the previous hidden state time-stamp will be forgotten. The update gate settles how much of the current input will be used to update the hidden state. The Gated Recurrent Unit (GRU) is another type of recurrent neural network (RNN) designed to address the vanishing gradient

problem and improve learning on sequential data. The operation of a GRU network can be described using several key equations.

The reset gate is typically computed using a sigmoid activation function.

$$r_t = \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r) \tag{6}$$

Where:

- x_t - input at time t,
- h_{t-1} - previous hidden state,
- W_{xr}, W_{hr}, and b_r - weights and biases for the reset gate,
- σ - sigmoid function.

Similar to the reset gate, the update gate is typically computed using a sigmoid activation function.

$$z_t = \sigma(W_{xz}x_t + W_{hz}h_{t-1} + b_z) \tag{7}$$

Where:

- W_{xz}, W_{hz}, and b_z - weights and biases for the update gate.

The candidate hidden state ($\tilde{h}_t$) is calculated using the current input x_t and the reset gate r_t. This candidate's hidden state represents the new information that could be added to the current hidden state.

$$\tilde{h}_t = \tanh(W_{xh}x_t + W_{hr}(r_t \odot h_{t-1}) + b_h) \tag{8}$$

Where:

- W_{xh} and b_h - weights and biases for the candidate hidden state,
- $\odot$ - element-wise multiplication,
- tanh - hyperbolic tangent function.

Finally, the hidden state h_t is updated using the update gate z_t, which determines how much of the candidate hidden state $\tilde{h}_t$ to incorporate into the current hidden state.

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \tag{9}$$

The differences in RNN, LSTM and GRU are shown in the simplified visualization in Fig. 1.

2.3. Proposed neural architectures for earthquake prediction

To predict earthquakes, we propose a network architecture consisting of four layers: a recurrent layer and three dense layers. The networks will accept sequences of historical data: one or two elements. This action will allow the analysis of the solution in terms of predictions based on only one value, as well as two values indicating the time distance between these quakes. The modeled networks are presented in Tab. 1-2 with details about parameters number and as follows:
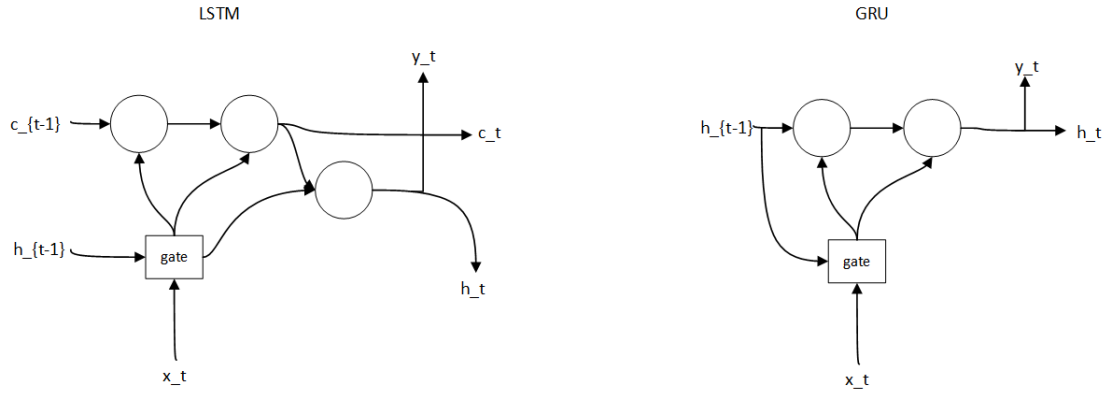
Figure 1: Visualization of neurons' architecture: LSTM and GRU

1. Input data: the shape was defined as $(1, 3)$ or $(2, 3)$ which meant that each input sequence has one/two-time step and three features (Latitude, Longitude, Depth),
2. LSTM/GRU Layer with 10 units,
3. Dense Layers:
   - Following the LSTM layer, there are three dense (fully connected) layers.
   - The first dense layer has 30 units and uses the ReLU activation function.
   - The second dense layer has 10 units with ReLU activation and dropout regularization technique with a threshold equal to 0.4.
   - The third dense layer has 1 unit that corresponds to the output classes which means 'Magnitude'.

Layer (type)	Output Shape	Param #
LSTM	(None, 10)	560
dense	(None, 30)	330
dense_1	(None, 10)	310
dense_2	(None, 2)	22
Total params		**1,222**
Trainable params		**1,222**
Non-trainable params		**0**

Table 1
Summary of LSTM model without distinction on the entered input

Both models were trained using Adam optimizer and the following loss function:

$$L\left(y_{true}, y_{pred}\right) = \frac{1}{n} \sum_{i=1}^{n} \left(y_{true} - y_{pred}\right)^2 \tag{10}$$

Layer (type)	Output Shape	Param #
GRU	(None, 10)	450
dense	(None, 30)	330
dense_1	(None, 10)	310
dense_2	(None, 2)	22
Total params		**1,122**
Trainable params		**1,122**
Non-trainable params		**0**

Table 2
Summary of GRU model without distinction on the entered input

where y_{true} is a true label and y_{pred} is predicted value returned by network. The choice of using the presented loss function is motivated by the regression nature of the prediction task. In regression tasks where the goal is to predict continuous numerical values (such as Magnitude), MSE is a standard and effective loss function. MSE calculates the average squared difference between predicted values and true values, providing a measure of how well the model's predictions align with the actual target values. The network architecture is composed of 52 neurons, which is small for a prediction task. However, the network accepts three numerical values defining the most important information about a given phenomenon, which should enable obtaining correct results. It is worth noting that the use of a layer with recurrent neurons contributes to making predictive decisions based on the context analyzed through the possibility of sequential analysis. Additionally, the recursive layer allows us to pay attention to long-term dependencies between values. Consequently, the modeled architecture consisting of only 52 neurons arranged in four layers and two types of neurons (LSTM/GRU and classic one in dense layer) is a model prepared for predictive analysis.

3. Experiments

The database that was used contained information about various earthquakes with magnitude of 4.0 or higher that happened in Turkey since 1900. It consisted of a total of 14 columns. For this paper, we used four columns. We involved localization of the earthquake's epicenter ("Longitude" and "Latitude"), hypo-center depth("Depth") and measure of the amount of energy released during an earthquake ("Magnitude"). A significant portion of the remaining columns contained NaN (missing) values or string data. Furthermore, in the other columns, the data was identical for the majority or all entries, thereby limiting their relevance for analysis. The database is available online on Kaggle[1]. Building the model did not involve scaling or normalizing data. The decision aimed to examine raw relationships and patterns in the dataset without changing the original feature distribution. Skipping normalization and scaling lets the model's performance
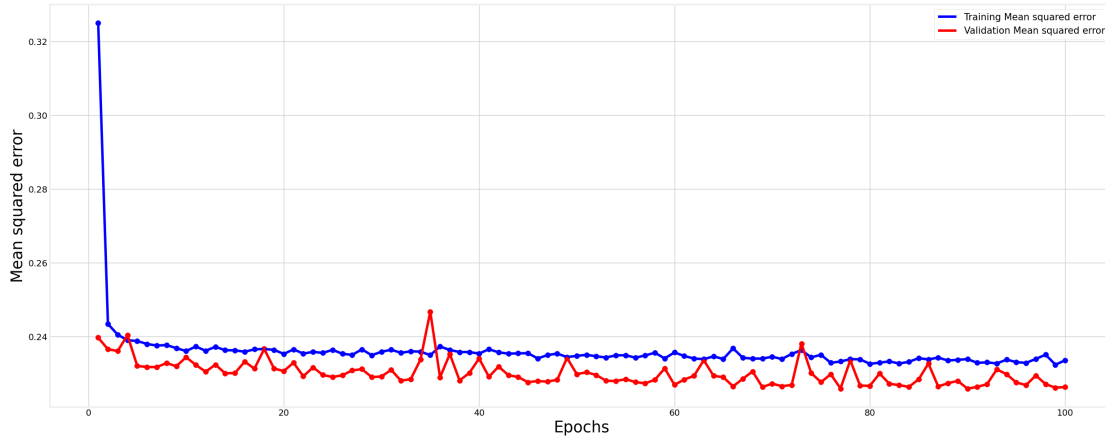
---

[1]https://www.kaggle.com/datasets/kmlyldrn/earthquakes

Figure 2: Network model with an LSTM layer for single-element sequences
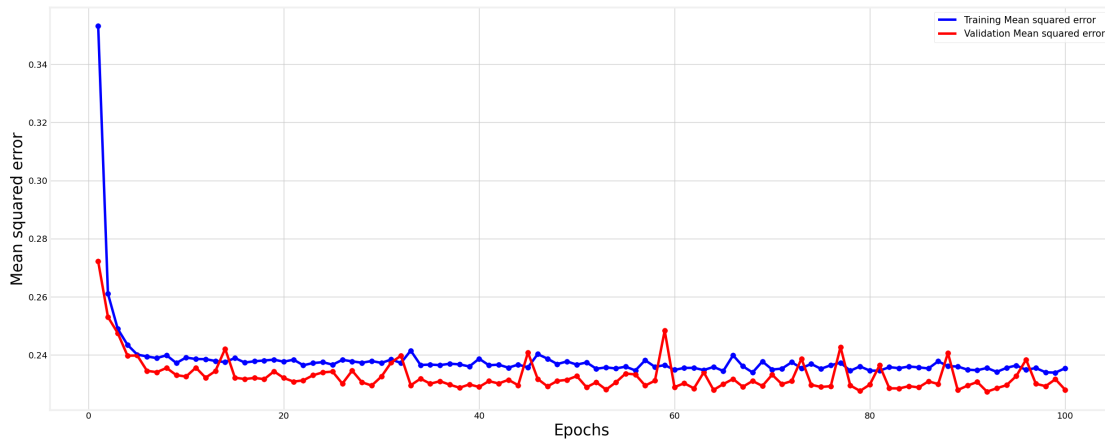

Figure 3: Network model with a GRU layer for single-element sequences

provide insights into how it handles varying data ranges and variances. This clear view reveals inherent strengths and weaknesses. However, it's important to note that in practical applications or for improved generalization, preprocessing steps like normalization and scaling are typically recommended to enhance model performance and stability across varying datasets. The data set was divided into two subsets: training and validation. For this purpose, a 70:30 split was used, i.e. 70% of the data was allocated to the training set and the remaining 30% to the validation set.

The first step was to check how the classifier is learned for a single data sequence. The results for both networks are shown in Fig. 2 and 3. In both cases, the MSE quickly decreases to a low value. It should be noted that there are small fluctuations in the obtained values. This is especially visible in the network with GRU neurons. Analysis of the loss function graphs indicates very small values and a slow decline. A larger decrease in the value of the loss function is visible from the LSTM analysis. Despite this, for both architectures, the accuracy values drop below 0.24, but for networks with GRU layers, spikes are increasing this value in the validation
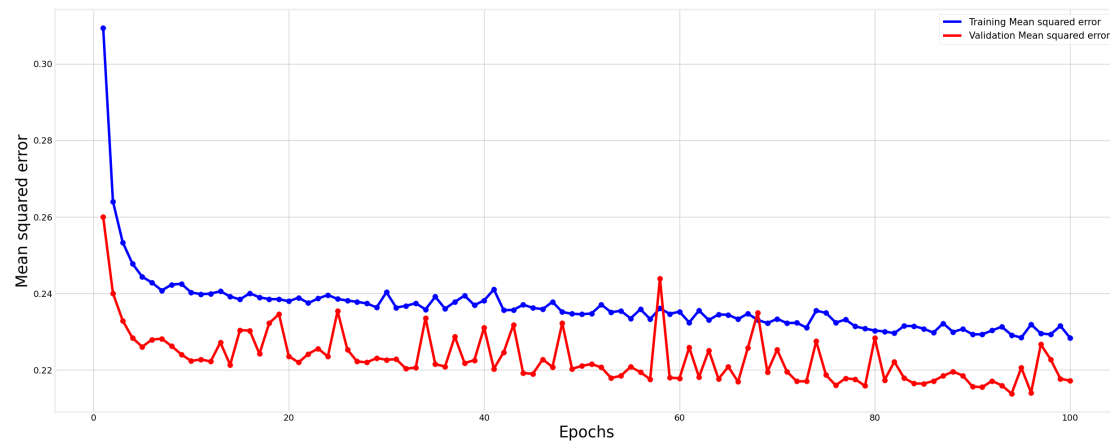
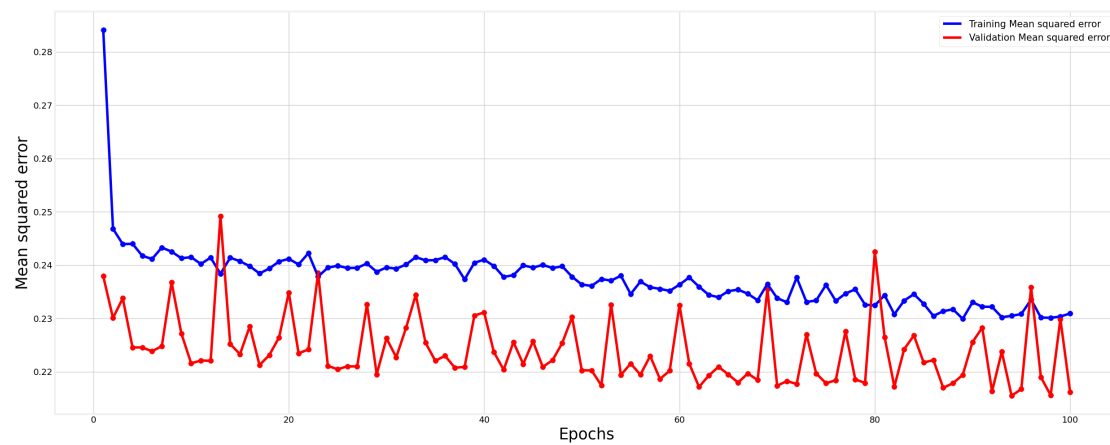**Figure 4:** Network model with an LSTM layer for two-element sequences

Figure 5: Network model with a GRU layer for two-element sequences

set.

In the case of a sequence consisting of two vectors, the learning results are not more stable, as we can see in Fig. 4-5. For the model with LSTM neurons, MSE decreases for both sets. However, for the validation set, the loss value drops below 0.22 after more than 50 training iterations. A similar situation was noticed for the second network consisting of the GRU layer. However, the decline in value for this architecture is faster. The main differences that have been noticed are that for networks with LSTM there are no large jumps as for GRU. However, in a network with a GRU layer, the values decrease faster than in LSTM. Moreover, for the training set, the GRU classifier achieved a lower value after 100 iterations.

4. Conclusion

Prediction is a task based on an appropriate database with historical values. A very large number of records may contribute to more accurate results. This work uses a publicly available earthquake database. A neural network model with recurrent neurons was proposed as a classifier. The study analyzed LSTM and GRU neurons. Analysis of the solutions showed that both solutions can achieve very good results. The advantage of using GRU neuron models is a smaller number of training parameters. However, it should be noted that the database used allowed for rapid achievement of high results, which may result in the possibility of overfitting. Based on the experiments performed, it can be concluded that for networks with the LSTM layer, the drop in error values is more stable, while for networks with the GRU layer it contains larger jumps. However, with two-element sequences, the network with the GRU layer achieved better prediction results.

In future works, we plan to use other tools and create an ensemble model with solutions like Monte Carlo, and Markov chains.

Acknowledgments

References

[1] S. Pereira, P. Canhoto, R. Salgado, Development and assessment of artificial neural network models for direct normal solar irradiance forecasting using operational numerical weather prediction data, Energy and AI 15 (2024) 100314.

[2] A. Kloczkowski, J. L. Fernández-Martínez, Z. Fernández-Muñiz, Computational models for covid-19 dynamics prediction, in: International Conference on Artificial Intelligence and Soft Computing, Springer, 2023, pp. 228–238.

[3] K. Kesik, Markov chains as a simulation technique for epidemic growth., in: Proceedings of the International Conference on Information Society and University Studies (IVUS 2019), 2019, pp. 1–4.

[4] J. Rashid, S. Batool, J. Kim, S. Juneja, An augmented artificial intelligence approach for chronic diseases prediction, Frontiers in Public Health 10 (2022) 860396.

[5] M. Zdanavičiūtė, R. Juozaitienė, T. Krilavičius, Telecommunication customer churn prediction using machine learning methods, Proceedings of the 27th International Conference on Information Society and University Studies (IVUS 2022) (2022).

[6] K. Prokop, D. Połap, G. Srivastava, Agv quality of service throughput prediction via neural networks, in: 2023 IEEE International Conference on Big Data (BigData), IEEE, 2023, pp. 2493–2498.

[7] P. Mahajan, S. Uddin, F. Hajati, M. A. Moni, Ensemble learning for disease prediction: A review, in: Healthcare, volume 11, MDPI, 2023, p. 1808.

[8] A. Jaszcz, The impact of entropy weighting technique on mcdm-based rankings on patients using ambiguous medical data, in: International Conference on Information and Software Technologies, Springer, 2023, pp. 329–340.

[9] K. Prokop, K. Połap, M. Włodarczyk-Sielicka, A. Jaszcz, End-to-end system for monitoring the state of rivers using a drone, Frontiers in Environmental Science (2023).

[10] J. Siłka, M. Wieczorek, M. Woźniak, Recurrent neural network model for high-speed train vibration prediction from time series, Neural Computing and Applications 34 (2022) 13305–13318.

[11] H. Zhang, L. Feng, J. Wang, N. Gao, Development of technology predicting based on eemd-gru: An empirical study of aircraft assembly technology, Expert Systems with Applications 246 (2024) 123208.

[12] J. Guo, C. Chen, H. Wen, G. Cai, Y. Liu, Prediction model of goaf coal temperature based on pso-gru deep neural network, Case Studies in Thermal Engineering 53 (2024) 103813.

[13] D. Połap, G. Srivastava, A. Jaszcz, Energy consumption prediction model for smart homes via decentralized federated learning with lstm, IEEE Transactions on Consumer Electronics (2023).

[14] C. Xu, J. Yu, W. Chen, J. Xiong, Deep learning in photovoltaic power generation forecasting: Cnn-lstm hybrid neural network exploration and research, in: The 3rd International scientific and practical conference "Technologies in education in schools and universities"(January 23-26, 2024) Athens, Greece. International Science Group. 2024. 363 p., 2024, p. 295.