UDC 621.382.2

*L.V. Sibruk Dr. of Engineering Science, I.V. Zakutynskyi*
*(National Aviation University, Ukraine)*

# Recurrent neural networks for passengers traffic data forecasting. Architecture overview and comparison.

*This work will consider the use of recurrent neural networks for time-series forecasting. Will be compared the most popular RNN architectures – LSTM and GRU with full network realization. Based on the above tests will be made a conclusion regarding the best architecture for passenger flow forecasting.*

## Introducing.

Recurrent neural networks (RNN) are a class of artificial neural networks in which connections between nodes form a time-oriented graph. This creates an internal state of the network that allows it to exhibit dynamic behavior over time. Unlike feed forward neural networks, RNN's can use their internal memory to process arbitrary sequences of inputs. We'll consider a comparison of the efficiency of the two most popular architectures of recurrent neural networks. LSTM (long short-term memory) and GRU (gated recurrent units, controlled recurrent neurons).

The purpose of the work is the development of the neural network based on LSTM and GRU architectures, using Python and Keras library with the comparison of results, and the choice of the best architecture for passengers flow forecasting.

## Problem statement.

The neural network will be implemented to predict passengers of international airlines, based on the dataset "Airline passengers January 1949 to December 1960", which is publicly available. The data cover the period from January 1949 to December 1960, or 12 years, with 144 observations. As a result, the developed neural network should predict the number of passengers of international airlines for years that are not in this sample.

*Table 1.*

Input dataset example

| Year/Month | Number of passengers |
|---|---|
| 1960/07 | 622 |
| 1960/08 | 606 |
| 1960/09 | 509 |

## Model implementation.

Model must perform the international airline passengers prediction. This is a problem where given a year and a month, the task is to predict the number of international airline passengers in units of 1,000.

The data ranges from January 1949 to December 1960 or 12 years, with 144 observations (Fig. 1). There is a certain periodicity that corresponds to the holiday period, and the vacation period.

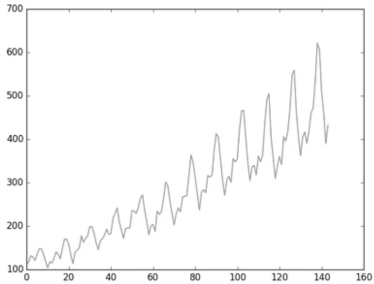


Fig. 1. Airline passengers January 1949 to December 1960

LSTM and GRU networks are quite sensitive to the scale of the input data. Especially when sigmoid or tanh activation functions are used. So, we need normalize data to the range [0, 1]. In Python, we can normalize a dataset using the *MinMaxScaler* preprocessing class from the scikit-learn library.

Also, we need to divide the input data into training and testing samples. In this case, the data is divided as follows - 67% initial sample (train), 33% test (test). For the network training, we create a new dataset where X is the number of passengers at a certain time (t) and Y is the number of passengers at the next time (t + 1).

At the next stage, data for learning and test sets are prepared. An LSTM network expects the input data (X) to be provided with some array structure in the form of [samples, time steps, features].

A neural network consists of: a visible layer with 1 input, 2 hidden layers with 64 blocks of LSTM neurons, and an output layer that makes a prediction of a single value. For LSTM blocks, we use default activation function – Relu (1).

$$f(u) = \max(0, u) \quad (1)$$

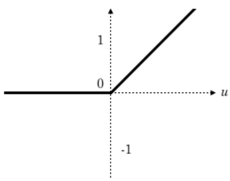


Fig. 2. Relu activation function

The learning process until the network can predict the correct output value based on the input (training sample).

For both networks (LSTM and GRU) we provide similar training process (Fig. 3).

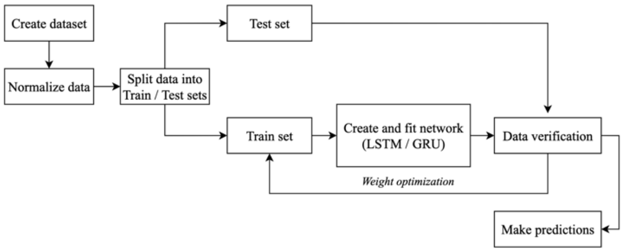


Fig. 3. Model training process schema

The network is trained with different epochs, and hidden layers count with package size = 1. Also we use Adam algorithm for model optimization. Adam optimization is a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments.

For the network quality check, a loss function is used, which determines the difference between the original (predicted values) and the previously known values. In this case, we use mean squared error (MSE) as loss function (2). The mean squared error tells how close a regression line is to a set of points. It does this by taking the distances from the points to the regression line (these distances are the "errors") and squaring them. The squaring is necessary to remove any negative signs. It also gives more weight to larger differences.

$$cost = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2 \quad (2)$$

The basis of neural network learning is the minimization of this value. We show model accuracy according to epochs count at the Tab. 2.

*Table 2.*

LSTM and GRU models accuracy

| Model | Accuracy | Error rate |
|---|---|---|
| LSTM (10 Epochs) | 71.18% | 28.82% |
| GRU (10 Epochs) | 67.11% | 32.89% |
| LSTM (30 Epochs) | 73.64% | 26.36% |
| GRU (30 Epochs) | 61.02% | 38.98% |
| LSTM (50 Epochs) | 86.83% | 13.17% |
| GRU (50 Epochs) | 67.12% | 32.88% |
| LSTM (100 Epochs) | 86.1% | 13.9% |
| GRU (100 Epochs) | 67.18% | 32.82% |
| LSTM (1000 Epochs) | 71.18% | 28.82% |
| GRU (1000 Epochs) | 67.11% | 32.89% |
| LSTM (10k Epochs) | 81.39% | 18.61% |
| GRU (10k Epochs) | 67.05% | 32.95% |
| LSTM (50k Epochs) | 76.82% | 23.18% |
| GRU (50k Epochs) | 66.22% | 33.78% |

| | | |
|---|---|---|
| LSTM (100k Epochs) | 53.68% | 46.32% |
| GRU (100k Epochs) | 66.16% | 33.84% |

According to the above data (Tab. 2) the best result (86.83% accuracy) got with LSTM architecture and 50 ”learning” epochs.

Also, we can notice the decline model performance point (LSTM > 50k Epochs). . This is the moment of saturation, or network retraining moment.

In the result graphs (Fig. 4, Fig. 5, Fig. 6, Fig. 7) shows the best results of both networks (LSTM - 100 Epochs, GRU - 10 Epochs).

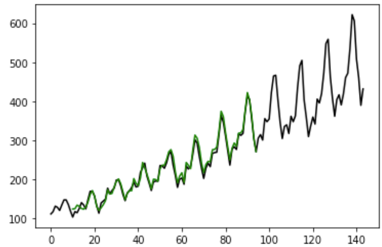


Fig. 4. LSTM model results – train data

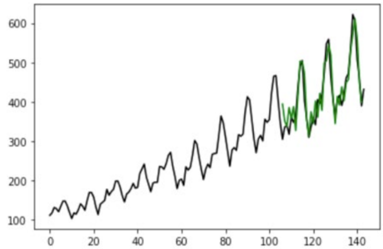


Fig. 5. LSTM model results – test data

The X axis is the month number, Y axis is the passengers count. The main line shows the input data, the auxiliary one - the result of the model forecast.

Fig. 4 and Fig. 6 shows network accuracy with train data. It this case both networks, show very similar good results. But real network performance shows at Fig. 5 and Fig. 7 (network prediction based on the test data).

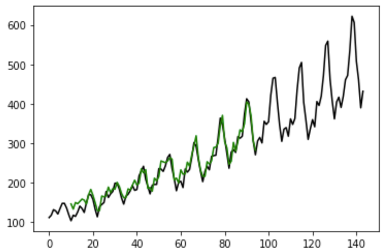


Fig. 6. GRU model results – train data

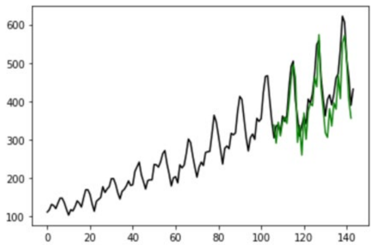


Fig. 7. GRU model results – test data

## Conclusions

The results show that RNNs are well suited for forecasting time series data (like passengers flow), but have some problems with shot-memory. In this study, we developed an RNN model for airline passenger prediction. We have selected network parameters (architecture, train/test set, hidden layers count, optimization function, epoch count) to generate a model with the highest accuracy.

LSTM network in practice, showed better results - 86.83% accuracy (GRU - 67.12%). According to the small size of the input dataset, both results can be considered good. As has been said in the "Abstract" chapter, further improvement must be related to the increase in input data.

So, neural networks based on LSTM architecture can be used for nonlinear, long-term data prediction with high accuracy. RNN based on GRU architecture is less accurate than the LSTM but has the advantage of being relatively simple, and time-consuming, with relatively good performance, and low computing time.

## References


1. Nam, K., & Schaefer, T. Forecasting international airline passenger traffic using neural networks, 239-252, 1995.
2. Keras.io – Developer library https://keras.io/api/optimizers/adam
3. W. Min and L. Wynter, "Real-time road traffic prediction with spatiotemporal correlations," Transp. Res. C, Emerg. Technol., vol. 19, no. 4, pp. 606–616, 2011.

A. Abadi, T. Rajabioun, and P. A. Ioannou, "Traffic flow prediction for road transportation networks with limited traffic data," IEEE Trans. Intell. Transp. Syst., vol. 16, no. 2, pp. 653–662, 2015.

4. Quang Hung Do et al. / Journal of Computer Science 16 (7), 2020.
5. Simon Haykin – Neural networks. A comprehensive Foundation. Second edition, 919-925, 2020.