Traffic Prediction of Real Network Traffic Data with LSTM Neural Networks

No Author Given

No Institute Given

Abstract. The aim of this work is to make time series predictions for real network traffic data by using long short-term memory neural networks (LSTMs). The data used for this work was captured in an office environment by a company from the field of network security and monitoring. The trained LSTMs are compared to baseline methods which have lower training costs. It is shown that the relative performance of LSTM-based predictions depends on the composition of the data. For aggregated traffic, the analyzed time series contains regular patterns with no big gaps and the trained LSTM outperformed baseline methods with an NRMSE of 0.072. In case of non-aggregated traffic of single clients/services with more gaps, jumps or steep trends within the time series, baseline methods can outperform LSTMs. Out of the used baseline methods, the ARIMA method shows the best performance in most experiments. Finally, it is also shown how the traffic predictions can be used to statistically detect anomalies within networks by considering the difference between the predicted and the actual values.

Keywords: Network Traffic Data, LSTM, ARIMA, Time Series Prediction, Anomaly Detection

1 Introduction

With the need to support a diverse set of services and products in modern communication networks, network administrators must monitor and analyze their networks to make suitable management decisions and guarantee security and quality of data transmission. Network traffic prediction can be highly valuable for various use cases including network monitoring, dynamic bandwidth allocation, network planning, congestion control and network security.

Dependent on the specific use case for traffic prediction and analysis, different types of network traffic data are used. Examples include the number of packets from a source to a destination or, at an even higher level, the number of connections. In this work the traffic in bytes within a specific time range is used as measure [1]. The recorded traffic data consists of time series on a per client basis and is further distinguished by specific types of services.

The idea of traffic prediction in this context is to learn from time series data in order to predict the future course of the time series. So, based on past

data, the goal is to find a model that describes the time series well enough to be able to predict how the time series will continue in future. Typically, time series are decomposed into trend, seasonal, and residual components. Only the trend and seasonal components are used for traffic prediction, since the residual component contains noise which is not predictable. A simple approach to approximate the smooth signal without noise is for example the use a simple moving average (SMA) filter. There are various approaches to create prediction models, which also differ in their complexity. Due to their ability to process temporal information and their performance shown in past works (cf. section 2), recurrent neural networks in form of LSTMs are considered for prediction in this work.

Finally, based on the resulting traffic predictions, anomalies can be identified in time series. Since anomalies are events in a time series that are neither normal nor explainable by past values, anomalies could simply be defined as points of high variation between the predicted time series and the actual values.

2 Related Work

Wide-area network traffic data is analyzed and different baseline prediction methods (i.e. historical mean/median and simple exponential smoothing) are compared in [2]. We have also used some of these prediction methods as baseline methods in our comparison. One work using LSTM neural networks for traffic prediction uses a combination of LSTMs with autocorrelation to predict real network traffic data [3]. It is shown that such an approach is efficient and suitable for predicting network traffic that behaves non-linearly. Another work compares different variants of recurrent neural networks for predicting real network traffic with LSTMs and GRUs (Gated Recurrent Units) identified as most suitable for traffic volume prediction [4].

A standard method often used in literature for traffic prediction not based on neural networks is ARIMA (Auto-Regressive Integrated Moving Average). The prediction is based on a linear combination of previous values and noise. Similar to LSTMs, ARIMA also has a set of hyperparameters, e.g. for the filter lengths, that need to be adapted to the data and optimized [5].

3 Long Short-Term Memory Neural Networks (LSTM)

An LSTM neural network is a type of recurrent neural network (RNN). Similar to a memory, it saves information of the past inputs, which is realized through loops within the neural network. Simple RNNs have the problem of the vanishing or exploding gradient in the backpropagation, because the network uses all inputs it has ever seen for predicting the next value. An LSTM is designed to overcome this problem by using a forget, input and output gate within one cell (visualized in Figure 1).

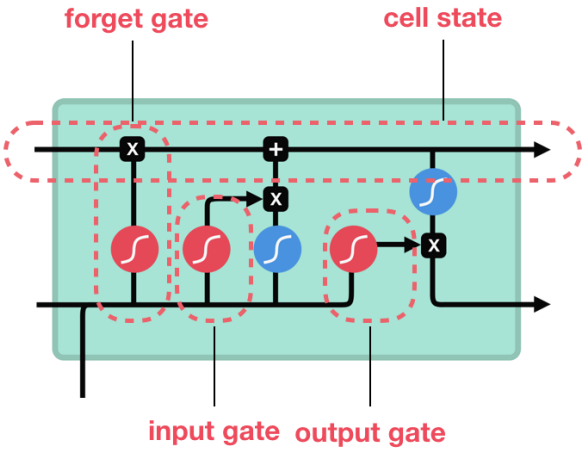

Fig. 1. Structure of a LSTM cell [6]

More specifically, the output of the cell state of an LSTM cell (memory) is the sum of the output of the previous cell filtered by the forget gate and the input of the current cell filtered by the input gate. This sum is further filtered by the output gate. The forget gate takes into account old and current information to decide whether to forget or remember information. The input gate also decides, based on the old and current information, which of the new values are relevant for the prediction. The cell's output is a combination of the generated cell state and the actual (old and current) values. This output is then passed to the next LSTM cell [7].

An LSTM neural network is then constructed by combining several of these LSTM cells in a layer and then stacking these layers. The input of an LSTM is an observation of one time step, that can be a vector if the observation consists of several features.

4 Experimental Setup and used Methods

The computations were performed on a system containing 20 Intel Xeon Silver 4114 CPU cores, 64 GB RAM and an NVIDIA GeForce RTX 2080 Ti Rev. A graphics card hosting 11 GB of GDDR6 memory, and a system with 48 Intel Xeon Gold 6326 CPU cores and 78 GB of RAM. Both systems ran Ubuntu 20.04 LTS as operating systems. Additionally, the following software libraries were used:

- `python 3.10.6`
- `keras 2.12.0`
- `keras-tuner 1.3.5`

- `tensorflow 2.12.0`
- `tensorflow-gpu 2.6`

4.1 Network Traffic Data

To test the algorithms with a real dataset, a company from the field of network safety and monitoring kindly provided network traffic data. The data was recorded at a gateway located between an internal office network and the public Internet. For each 1-minute interval, a data record is produced. It can be distinguished whether the traffic flows from one device to another device within the internal network or if the traffic is coming from the Internet to an office-internal device (incoming traffic) or vice versa (outgoing traffic). Moreover, the volume of traffic exchanged between two specific IP addresses (of client and server) is quantifiable and the type of service associated with this traffic, such as Social Media or Cloud, can also be determined (deep packet inspection is used to detect the class of application).

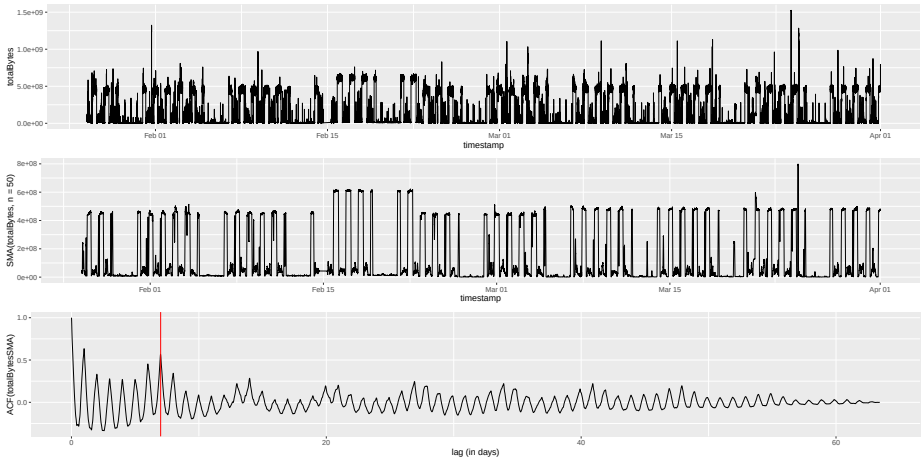

Fig. 2. Section of network traffic data, top: total packets from 26th Januar to 31st March 2022, in the middle: smoothed time series by SMA with filter length 5, kernel length 1, bottom: auto correlation function of time series

For this work, the data is considered in two different forms. First, the total number of bytes per 10 minutes over a time period of three months is used to test the traffic prediction. Since the data was recorded almost uninterruptedly every minute over three months, there are approximately 130000 data points in the raw time series. These are summed up in 10 min intervals resulting in 13000 data points. As it can be seen in Figure 2, there is a repeating pattern in the data, with high traffic during the day of working days and hardly any traffic

otherwise. This can be seen in the autocorrelation function (ACF) of the time series at the bottom in Figure 2 as well. To reduce the noise, the time series is filtered by a simple moving average (SMA) with filter length 5 and kernel length 1. It can be seen, that the difference between the value of high traffic and low traffic is very high ($7 \cdot 10^5$).

Since the company that provided the data is interested in using traffic prediction to find anomalies, and due to the assumption that it is more difficult to detect network attackers by looking at all traffic (since malicious behaviour can be lost in the sum of all bytes including noise and natural outliers), the data is additionally looked at from a more detailed perspective. More specifically, the total number of bytes per minute per client IP address and service group is considered over a period of three months. Over this period, there appeared 506 different client IP addresses and the traffic was grouped into 27 different service types. It should be noted that the data is not labeled w.r.t. possible anomalies and it may be that there are no anomalies at all.

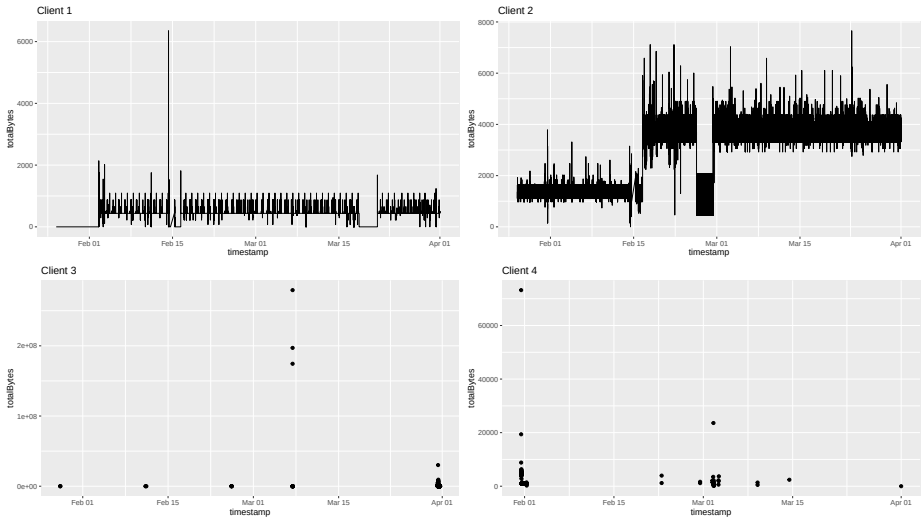

Fig. 3. Examples of time series from network traffic of specific client IP addresses (top: more regular patterns, bottom: time series with large gaps and high jumps)

It is worth noting that there are very different types of clients (examples can be seen in Figure 3). Some clients represent servers doing backups or other programs which produce nearly perfect regular traffic with the same repeating pattern. There are clients which represent typical employees in an office which produce traffic during the day on working days and zero traffic otherwise. And 60% of the clients have zero traffic most of the time (98% samples are 0, which corresponds to one value higher than 0 per day in average) and irregularly occurring peaks of high traffic otherwise. Furthermore, the data per client was

separated into service groups. All services have been divided into 27 different types of services by the company that provided the data. In Figure 4, it can be seen that different service groups have specific patterns of traffic in the course of one day.

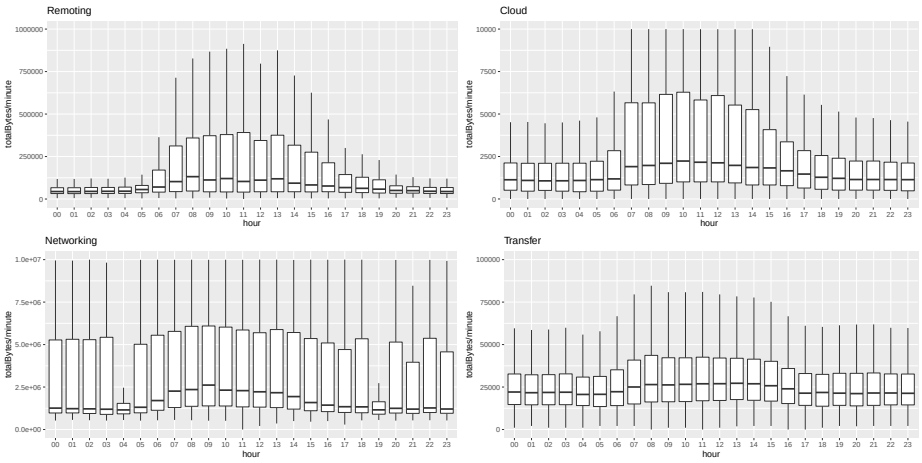

Fig. 4. Traffic patterns of service groups in the course of one day, illustrated by boxplots of Bytes per minute for each hour of the day (outliers are removed for better visualization).

4.2 Baseline Methods for Traffic Prediction

The following baseline methods are easy to implement and are more straight forward approaches to traffic prediction. They can therefore serve as a basis for comparison whether a more complex model like a LSTM is reasonable.

- **Previous Value (prevVal)**
  One simple idea is to make a prediction for the next value of a time series by simply taking the last seen value of the time series

- **Historical Median (histMed)**
  Take the Median of several last values as forecast for the next step

- **Historical Mean (histMean)**
  Take the Mean of several last values as forecast for the next step

- **Linear Regression (linear)**
  Linear regression also takes several last values. It computes a weighted sum of these values for the prediction [8]

– **Auto-Regressive Integrated Moving Average (ARIMA)**
(cf. section 2)

4.3 LSTM with keras

LSTM neural networks are implemented in the libraries tensorflow[1] and keras[2]. To design an LSTM, first the number of steps of the past, the number of steps into the future and the selection of features (in case of a multivariate time series) that are relevant for the prediction have to be specified. As described in section 3 there are many options for designing an LSTM, those for neural networks in general plus some extra hyperparameters. In this work, the LSTM was designed with some LSTM layers, followed by one dense layer. The last LSTM layer provides as output the final prediction after seeing all input time steps (keras: return sequences = FALSE in the last LSTM layer). For the first case (predicting the traffic in total Bytes over 10 minute intervals) the number of epochs was limited to 1000 without an early stopping criterion. In the second case (predicting the traffic per client and service group summarized in 1 minute) the input was grouped into batches of size 50, the number of epochs is 70 and there is also no early stopping criterion. For the loss function, MSE was chosen in both cases. All other hyperparameters were optimized with the help of the keras-tuner[3]. A detailed list of the hyperparameters used can be found in section 5.

4.4 Configuration of Traffic Prediction Scenario

There are many possible ways to configure the traffic prediction scenario. As part of a pre-test, the number of steps in the past used for prediction was varied. Interestingly, the prediction results were better if fewer steps were used. Since the computation time is also shorter when using only a few time steps for prediction, we finally decided to use 5 steps of the past. In some cases, it might be reasonable to increase the number of time steps of the past to capture one period of a periodic time series. However, for the real data used in this work, it was not necessary to capture a full period to get a better prediction. Another parameter that can be varied is the number of steps into the future. The choice of this parameter highly depends on the application and the requirements. For the case considered here and w.r.t. the anomaly detection, it is sufficient to predict one step ahead. Furthermore, some of the basic methods are not able to predict more than one step without additional implementation effort. With respect to both, the data and the prediction scenario, it is also an option to consider several features e.g. the week day as additional information next to the total amount

[1] https://www.tensorflow.org/tutorials/structured_data/time_series
[2] https://keras.io/api/layers/recurrent_layers/lstm/
[3] https://keras.io/keras_tuner/

of bytes in the time range. But for this first application of traffic prediction, we focus on the single feature (sum of bytes over the specified time range).

As final configuration of the traffic prediction case considered in this work five steps from the past are used to predict one step ahead for the prediction of the total amount of bytes in the traffic data. Since it was desired to apply the traffic prediction to the filtered time series without noise, the time series are smoothed for convenience only by a simple moving average (length of kernel equals 1 and filter length equals 5). Then, the model is trained by using the first 80% of the time series as training data and the remaining 20% as test data.

4.5 Evaluation Measures

To measure the accuracy of a time series prediction, the differences between the prediction $p = [p_1, ..., p_n]$ and the actual values $s = [s_1, ...s_n]$ is considered in the following ways.

- **Normalized Mean Absolute Error (NMAE):**
  $\frac{1}{|\max_{i\in[n]} s_i - \min_{i\in[n]} s_i|} \frac{\sum_{i\in[n]} |s_i - p_i|}{n}$
- **Normalized Root Mean Squared Error (NRMSE):**
  $\frac{1}{|\max_{i\in[n]} s_i - \min_{i\in[n]} s_i|} \sqrt{\frac{\sum_{i\in[n]} (s_i - p_i)^2}{n}}$

4.6 Anomaly Detection

For finding the global outliers in a time series, the Z-Score can be directly applied to the values of the time series. To detect contextual outliers, the Z-Score can be applied to the difference between the original time series and a smoothed version of the time series. Here, the result of the traffic prediction can be used as well. Since the predicted time series represents the expected course of values over the time, contextual outliers might be found by looking at the difference between the predicted time series and the actual time series. In this work, a threshold of five times the variance in the data is used. It should be noted that mathematical outliers in a time series as identified here do not always match to actual anomalies in the traffic and, vice versa, actual anomalies could exist without being visible in the traffic time series.

5 Results and Discussion

In the following, the results are discussed separately for the two cases considered. In section 5.1 the traffic prediction of the total bytes in 10 minute intervals is considered and section 5.2 deals with the traffic prediction for the combinations of clients and service groups.

5.1 Traffic in total Bytes (summarized in 10 minutes)

First, the total traffic in Bytes per 10 minutes is considered. For the LSTM, the following hyperparameters were found by the keras-tuner:

- layer LSTM, units=250, return_sequences=TRUE, dropout=0.0, recurrent_dropout=0.0
- layer LSTM, units=10, return_sequences=FALSE, dropout=0.0, recurrent_dropout=0.0
- layer Dense, units=1
- optimizer=adam, loss=MSE
- batchSize = 16

In Figure 5, the resulting NRMSEs achieved by the different traffic prediction methods are visualized. It can be seen, that the LSTM achieved a NRMSE of 0.072 for the test data set, which is the smallest error compared to the baseline methods. The best baseline method was the ARIMA model, where the NRMSE for the test data set was 0.074. The linear regression was slightly less successful than the previous value method with an NRMSE of 0.093 for the test data set. Especially the methods histMean and histMed were significantly worse than LSTM.

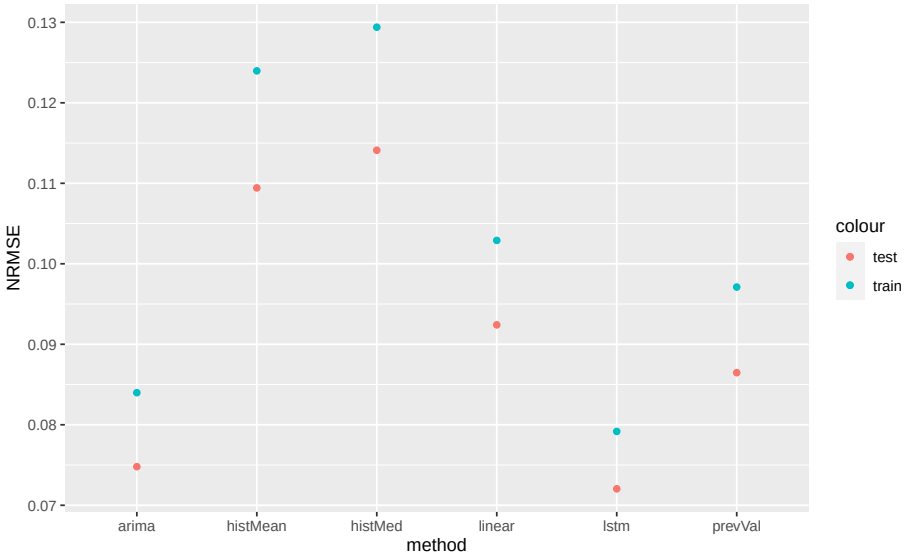

Fig. 5. NRMSE of traffic prediction of total Bytes for different prediction methods (comparison of test (red) and training (blue) data).

Interestingly, the errors are around 10% higher for the training data set compared to the test data. This might be due to the fact, that there is a period

of higher traffic in the mid of February, whereas the rest of the data (also in the test data) remains at a constant level of traffic (see Figure 2). The results of the best prediction methods for the test data set are depicted in Figure 6. On closer examination, it can be seen, that the method linear overshoot or undershoot the level of traffic values after jumps within the time series. For this reason, the NRMSE might be larger than for the other methods.

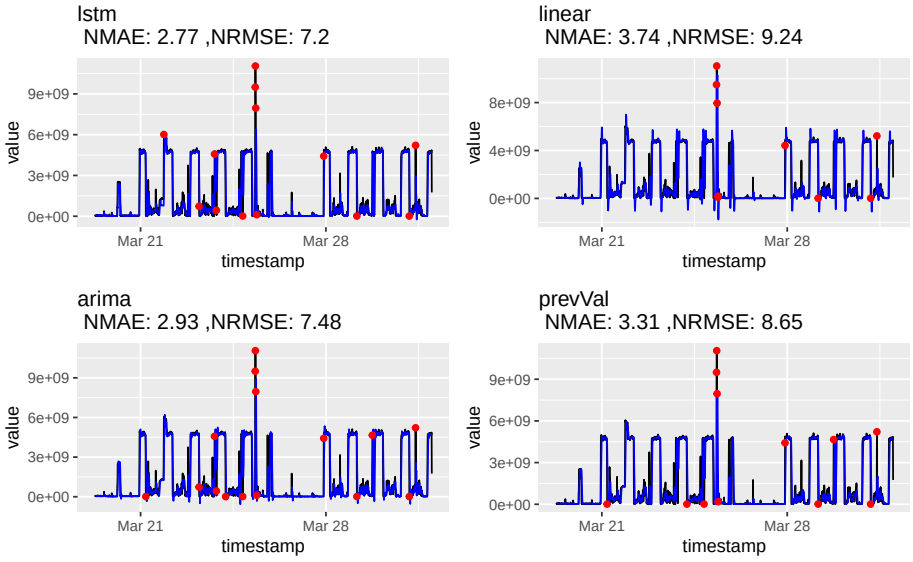

Fig. 6. Traffic prediction with different methods (black: original, blue: prediction) and anomaly detection based on traffic prediction (actual: NA, detected: red) for time series of total Bytes. (Evaluation measures are multiplied with 100 for readability.)

With respect to the anomaly detection, it is not possible to calculate numerical results because of the missing labels for anomalies. However, anomalies at high traffic peaks are detected. All other suspected anomalies are identified when there was a jump in the time series.

5.2 Traffic per client and service group (summarized in 1 minute)

Prior to the traffic prediction for specific clients and service groups, the time series are preselected. There are many clients which have zero traffic most of the time, when looking at the traffic of single service groups. We do not want to learn to predict zero traffic, but we want to concentrate on time series with more interesting traffic. Therefore, only parts of time series (1285 minutes) that contain less than 10% zeros are considered. This results in remaining 82 different time series of client - service group combinations. Due to the smaller number of

samples for each client - service group combination and the preselection made, the data volume of the individual time series is significantly smaller than in the first experiment. Therefore, in this case, 1-minute intervals are used instead of 10-minute intervals.

For this data, the following hyperparameters were found by the keras-tuner for the LSTM:

- layer LSTM, units=250, return_sequences=TRUE, dropout=0.005, recurrent_dropout=0.0
- layer LSTM, units=35, return_sequences=FALSE, dropout=0.0, recurrent_dropout=0.0
- layer Dense, units=1
- optimizer=adam, loss=MSE
- batchSize = 50

In Figure 7 and 8, the resulting NRMSEs achieved by the different traffic prediction methods are visualized. It can be seen, that all methods achieved similar results. For the test data set, the median of the NRMSEs is the lowest for the methods ARIMA and prevVal, followed by LSTM and histMean. The variance of the NRMSE is higher for LSTM compared to all other baseline methods. There is one outlier at NRMSE of 58.4 when using LSTM, this outlier is eliminated from the plot to have a better overview.

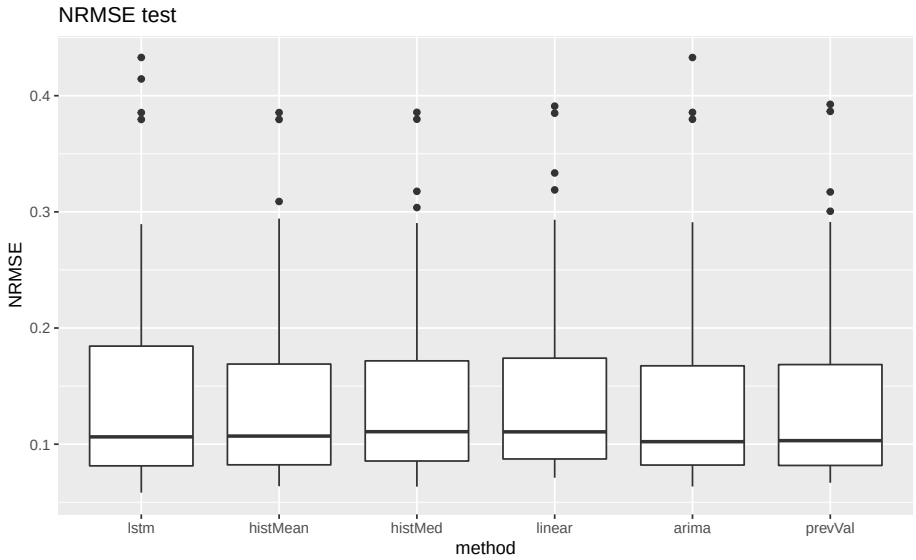

Fig. 7. Boxplots of NRMSE of traffic prediction of time series per client and service group, compared for different methods. Results for test data set.

For the training data, the median of NRMSEs was 15% lower when using LSTM or ARIMA, the best methods, compared to histMed, the worst method. Here, the distribution of values is very similar for all methods, except for the higher variance of NRMSEs for the linear regression.

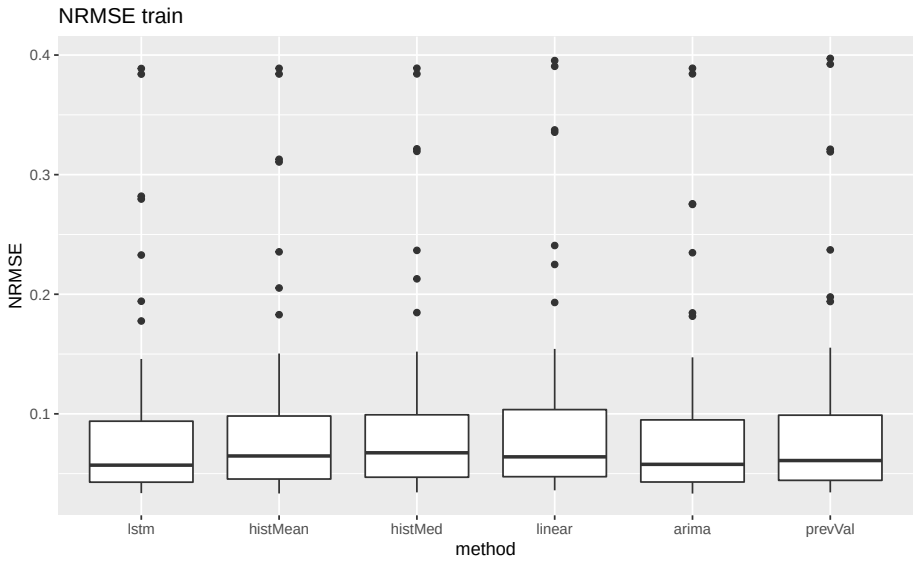

Fig. 8. Boxplots of NRMSE of traffic prediction of time series per client and service group, compared for different methods. Results for training data set.

By looking at the traffic prediction of specific time series (Figures 9, 10 and 11), we can learn when LSTM are successful in prediction and what the challenges for this method are compared to the baseline methods. For time series with high noise and many outliers (Figure 9), the prediction of the LSTM was more stable than with all other methods. Therefore, a smaller error could be achieved. Due to the large number of outliers and noise, the overall results for all methods are worse than for other time series.

In some cases (see figure 10), it happened that the prediction of the LSTM failed for the test data set. The level of the values was significantly above or below the actual values. This might be because of a change of level from training data to test data. But it is not clear, why the LSTM did not succeed in adapting to the new level, since it has a constant behavior (and thus also easy to correct). In this case histMean provides the best results, followed by ARIMA and prevVal with the same result-values. Overall, the results here are also generally worse compared to other time series.

In Figure 11, it can be seen, that the methods show different behavior at sudden jumps. The LSTM and histMean methods need around 15 minutes to follow

the new level of values (the downside of an outlier-robust prediction method). The ARIMA method is able to make a sudden jump faster. Although the prediction slightly overshoots at the end of the jump, the method provides the best results with regard to the error criteria. The prevVal method proves to be second best in this case, but this is also due to the definition of the method itself, as only the last value is taken, so that sudden jumps can be followed relatively quickly and there effects like overshooting do not occur.

To summarize, the success of the prediction method depends on the shape of the time series. If the time series prediction does not contain sudden jumps, LSTM outperforms the baseline methods. Furthermore, it should also be noted that although the ARIMA method does not always yield the best result, it consistently performs relatively well in all cases.

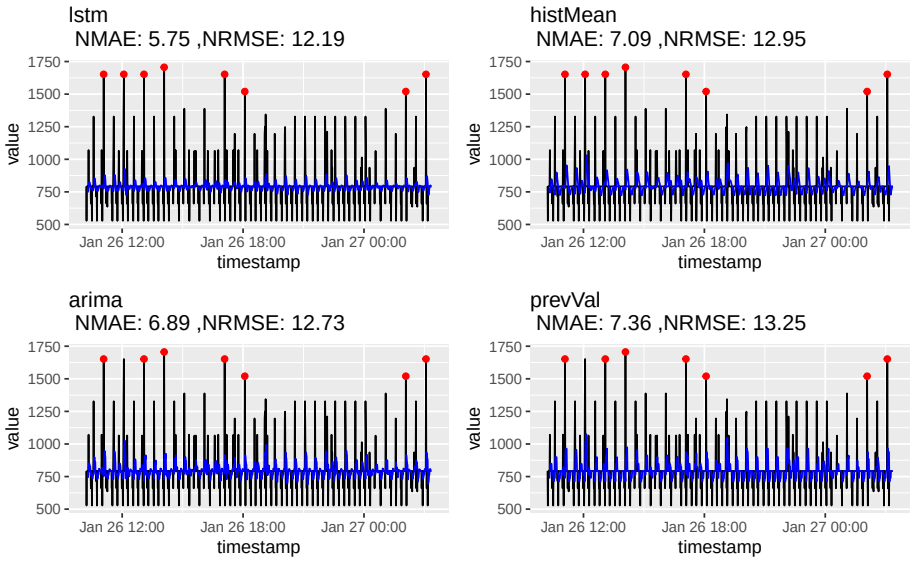

Fig. 9. Traffic prediction with different methods (black: original, blue: prediction) and anomaly detection based on traffic prediction (actual: NA, detected: red) for a specific client - service group time series. Example, where LSTM is more successful than baseline strategies due to more stability in prediction. (Evaluation measures are multiplied with 100 for readability.)

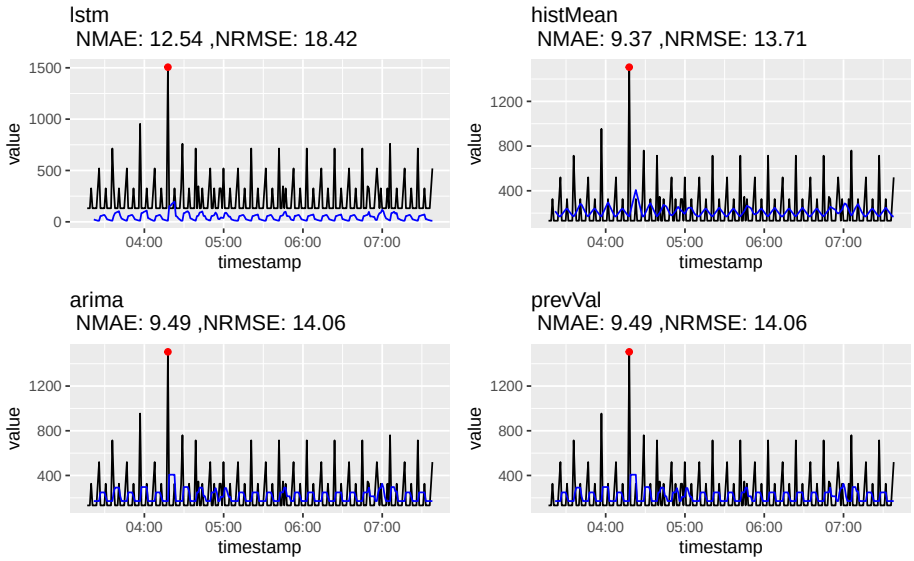

Fig. 10. Traffic prediction with different methods (black: original, blue: prediction) and anomaly detection based on traffic prediction (actual: NA, detected: red) for a specific client - service group time series. Example, where the LSTM prediction fails because of an additional offset. (Evaluation measures are multiplied with 100 for readability.)

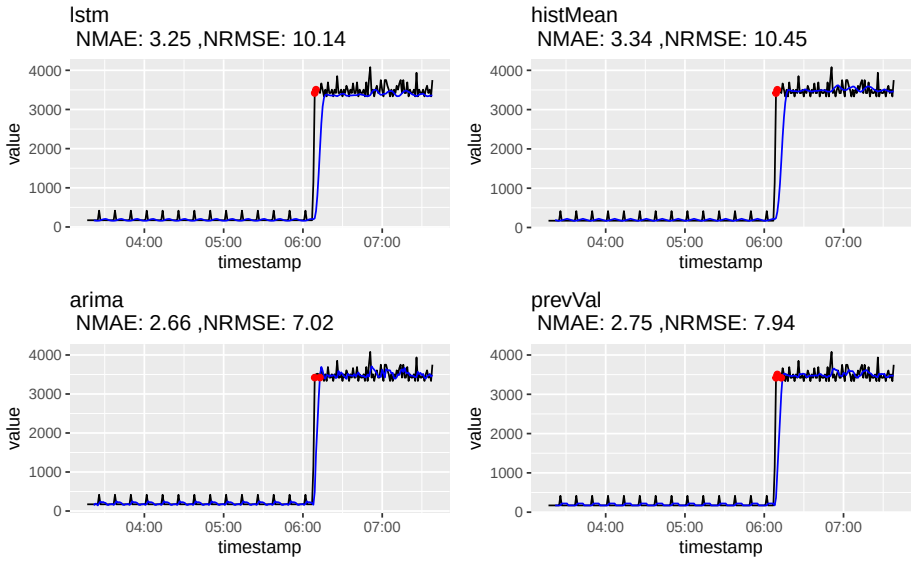

Fig. 11. Traffic prediction with different methods (black: original, blue: prediction) and anomaly detection based on traffic prediction (actual: NA, detected: red) for a specific client - service group time series. Example, where LSTM is slower in following jumps in a time series. (Evaluation measures are multiplied with 100 for readability.)

6 Conclusion

We have seen that for data without high jumps or permanent changes of the level of the values, LSTM outperforms all baseline methods, since LSTM detects periodicity and is robust against noise. However, sometimes the prediction of the LSTM was too strongly influenced by the training data and the results for the test data was significantly worse, e.g. there was a change in the level of values from training data to test data, so that the prediction failed because of an additional offset. In comparison, ARIMA was often the best baseline method and never the worst. In many cases it delivers similarly good results as LSTM, sometimes even better ones. The other baseline methods showed more variance in performance. However, it should be noted that ARIMA is significantly more complex compared to the other baseline methods and, similar to the LSTM, also has parameters that need to be adjusted according to the data.

7 Acknowledgements

The presented research has been performed as part of the project () supported by the Austrian Federal Ministry for Climate Action, Environment, Energy, Mobility, Innovation and Technology (BMK) and the Austrian Research Promotion

Agency (FFG) within "ICT of the Future - Cooperative R&D Projects between Austria, FFG and China, CAS" research programme.

References

1. Aleksandar Lazarevic, Vipin Kumar, and Jaideep Srivastava, "Intrusion detection: A survey," in *Managing cyber threats*, pp. 19–78. Springer, 2005.
2. Zhaohua Wang, Zhenyu Li, Guangming Liu, Yunfei Chen, Qinghua Wu, and Gang Cheng, "Examination of wan traffic characteristics in a large-scale data center network," in *Proceedings of the 21st ACM Internet Measurement Conference*, 2021, pp. 1–14.
3. Shihao Wang, Qinzheng Zhuo, Han Yan, Qianmu Li, and Yong Qi, "A network traffic prediction method based on lstm," *ZTE Communications*, vol. 17, no. 2, pp. 19–25, 2019.
4. Nipun Ramakrishnan and Tarun Soni, "Network traffic prediction using recurrent neural networks," in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2018, pp. 187–193.
5. David J Bartholomew, "Time series analysis forecasting and control.," 1971.
6. Michael Phi, "Illustrated guide to lestm's and gru's: A step by step explanation," https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21, 2018, Accessed 3rd August 2022.
7. Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, (1997).
8. David A Freedman, *Statistical models: theory and practice*, cambridge university press, 2009.