Deep Learning For Network Traffic Prediction

An Investigation into Long Short Term Memory Models

Justin Myerson
myrjus002@myuct.ac.za
University of Cape Town
Cape Town, South Africa

Abstract

Network traffic prediction is an important tool in managing network congestion, resources and security. It predicts future network traffic flows based on previous data, using either statistical time-series or machine learning approaches. Efficient prediction can improve the quality of service and lower operating costs for network service providers. Existing literature shows that deep learning models learn network traffic patterns more efficiently and predict future traffic more accurately than traditional prediction models. The purpose of this paper is to develop and test Long Short Term Memory (LSTM) models to learn and predict future network traffic on the South African Research and Education Network (SANREN). These models were evaluated both in terms of their prediction accuracy, and their computational complexity. The results demonstrated that a Stacked LSTM was the most accurate prediction model, at the expense of using the most computational resources to make predictions.

CCS Concepts: • **Computing methodologies** → **Unsupervised learning**; **Neural networks**; • **Networks** → **Network performance analysis**.

Keywords: Long-Short Term Memory, Mean Squared Error, Network Traffic Prediction, Stacked Long-Short Term Memory, Bidirectional Long-Short Term Memory, Computational Complexity

1 Introduction

In today's world, the internet and its applications have become a vital tool for all types of users. As more individuals are gaining access to the internet for the first time, and others increasing their usage, networks are having to manage their limited bandwidth effectively. COVID-19 has caused a significant surge in internet traffic [5]. Accurate network traffic prediction - provided by models that predict future traffic flows and fluctuation - would help network providers to alleviate congestion and load management issues. Predicting network traffic in the short term aids in dynamic resource allocation, while longer-term prediction provides insight into how a service provider may improve their network capacity and performance [12]. Deep learning models have become a popular approach to time series forecasting, which includes network traffic data. An analysis of existing literature shows deep learning models consistently outperform traditional statistical and machine learning methods, and that Recurrent Neural Networks and their subsets are now the gold standards for network traffic prediction [8, 9, 11, 13, 15].

SANReN is an organised network of education and research institutions within South Africa [3]. Within the network, there are time-series traffic data flows that can be too large to be adequately monitored by traditional data analysis techniques. Hence, a deep learning approach is proposed as an alternative method to perform network traffic analysis and prediction for SANReN. The objective of this paper is to critically evaluate deep learning approaches to determine the best model for network traffic prediction on the SANReN. This paper also investigates the computational resources required for each implementation and concludes on the trade-offs between computation time and prediction accuracy - specifically when considered for the SANReN use case.

1.1 Structure of Report

The paper begins with a brief overview of the South African Research and Education Network, which is the network on which this project is based. Following this, the paper will discuss Deep Learning and LSTMs in detail and the previous work that has been done in network traffic prediction. Experiment design and results will come next. Closing off, the key findings and limitations of the project will be visited, as well as future work that can be conducted.

2 South African Research and Education Network

SANREN is a high-speed network which connects South African tertiary education institutions, research centers, academic hospitals and museums. It has a network current capacity of 3557Gbps and was rolled out starting in 2007 [3].

2.1 Need for Network Traffic Prediction

It is important to consider the constraints and resources of a network when evaluating a candidate model for network traffic prediction. Both computational complexity and run time can be a limiting factor for less-resourced networks such as SANReN, which may result in different network traffic prediction models being better suited for them. Existing literature on network traffic volume predictors has shown that neural networks - particularly LSTMs - provide

improvements in accuracy and performance over traditional statistical prediction methods. Furthermore, LSTM derivative models, such as the stacked LSTM and bidirectional LSTM, have out-performed baseline LSTMs [6, 11].

The computational feasibility of LSTM and LSTM-derivative prediction models will be investigated. These models will be replicated and trained on new SANReN data sets, to further assess their performance against traditional statistical models and conventional LSTM. This project is a departure from previous work done in the field, since this is specifically geared towards creating an optimal model for predicting traffic on the SANReN.

3 Background on Deep Learning

Network traffic prediction approaches have been formalized in a multitude of past studies. Furthermore, the growth of the internet and its networks have accelerated research, with deep learning techniques emerging as the prevalent tool for network traffic prediction. Historically, researchers used statistical prediction techniques such as ARIMA and Holt-Winters models, but deep learning models - particularly the LSTM - have shown to out-perform those. A Recurrent Neural Network can suffer from the vanishing gradient problem, which occurs when the network is unable to send back useful gradient information from the output layers to those layers that are more shallow. If this occurs, the RNN loses its ability to consider long term dependencies in calculations [4]. It is for this reason that Krishnaswamy et al. [11] propose that using an RNN is unsuitable for network traffic prediction, suggesting that an LSTM should be used for time-series predictions, to eliminate the vanishing gradient problem.

An LSTM is a type of RNN, which is formed by adding a short and long term memory unit to an RNN [7]. The addition of memory units allows the network to deal with the correlation of time series in the short and long term, and store dependencies that it deems important from earlier epochs of training [16]. Additionally, to control the use of historical information, the model uses an in, forget and out gate.

Krishnaswamy et al. [11] discussed the differences between using a Simple and Stacked LSTM learning approach. Stacked LSTM's were more accurate in predicting future traffic flows compared to traditional LSTM architectures, but at the cost of a higher computational complexity as a result of added LSTM layers. There is a a trade off here, but the choice between accuracy and computational efficiency allows network operators to decide what is more practical for their needs. In their training, Krishnaswamy et al. [11] noted that adding extra LSTM layers did not cause a noticeable change in accuracy. In fact, the baseline LSTM had the lowest Mean Squared Error overall. One limitation that will be investigated further in this project is whether these LSTM algorithms perform as well for smaller traffic volumes that one may see on an education network, as the results above were for links of capacity of over 100GB/s.

Cui et al. [6] investigated the use of a bidirectional LSTM for forecasting network traffic. A bidirectional LSTM (BDLSTM) is one that runs the input from both past to future, and future to past. This approach preserves information from the future and, using two hidden states combined, it is is able in any point in time to preserve information from both past and future [14]. Cui et al. [6] found that a stacked BDLSTM achieved a more accurate prediction. Importantly, the training times that they observed indicate that a BDLSTM is nearly double the training time of a regular LSTM. To our knowledge there was no mention of prediction times for a stacked BDLSTM, so we will investigate whether this increased prediction accuracy comes at the cost of a higher computational time compared to a stacked LSTM and a BDLSTM.

3.1 Long Short Term Memory

LSTMs are a type of Recurrent Neural Network architecture, which were introduced as a solution to vanishing gradient problem. Hochreiter and Schmidhuber worked to address the problem of long-term dependencies. That is, if the previous state that is influencing the current prediction is not in the recent past, the RNN model may not be able to accurately predict the current state. To remedy this, LSTMs have "cells" in the hidden layers of the neural network, which have three gates: input, an output, and a forget gate. These gates control the flow of information which is needed to predict the output in the network.

These improvements mean that LSTMs are capable of learning long term dependencies, since they are able to retain information from multiple previous time-steps. The forget gate within the cell dictates what information should be retained and what should be discarded. This is achieved through a sigmoid layer, which returns a value $0 < x < 1$. A value of 0 means that no information will pass through to the next cell, while 1 means that all information is passed through.

INPUT LSTM CELL

The input gate layer, also a sigmoid layer decides which values to update. A tanh layer then creates a vector of new values we could add to the state. The old state is then multiplied by the forget gate in order to drop the values we don't want, added then by the tanh vector values scaled by the sigmoid layer which decides how much to update each state value. Lastly, the output is a filtered version of the cell state. A sigmoid layer determines which part of the cell state will be outputted, cell state is then pushed through tanh to get values between -1 and 1 and multiplied by output of sigmoid gate to only output the desired values.

3.1.1 Bidirectional.

Not sure if required since there is a related work section which covers their application and explains them - Josiah please advise

3.1.2 Stacked.

Not sure if required since there is a related work section which covers their application and explains them - Josiah please advise

4 Related Work

Network traffic prediction approaches have been formalized in a multitude of past studies. Furthermore, the growth of the internet and its networks have accelerated research, with deep learning techniques emerging as the prevalent tool for network traffic prediction. Historically, researchers used statistical prediction techniques such as ARIMA and Holt-Winters models, but deep learning models - particularly the LSTM - have shown to out-perform those. A Recurrent Neural Network can suffer from the vanishing gradient problem, which occurs when the network is unable to send back useful gradient information from the output layers to those layers that are more shallow. If this occurs, the RNN loses its ability to consider long term dependencies in calculations [4]. It is for this reason that Krishnaswamy et al. [11] propose that using an RNN is unsuitable for network traffic prediction, suggesting that an LSTM should be used for time-series predictions, to eliminate the vanishing gradient problem.

An LSTM is a type of RNN, which is formed by adding a short and long term memory unit to an RNN [7]. The addition of memory units allows the network to deal with the correlation of time series in the short and long term, and store dependencies that it deems important from earlier epochs of training [16]. Additionally, to control the use of historical information, the model uses an in, forget and out gate.

Krishnaswamy et al. [11] discussed the differences between using a Simple and Stacked LSTM learning approach. Stacked LSTM's were more accurate in predicting future traffic flows compared to traditional LSTM architectures, but at the cost of a higher computational complexity as a result of added LSTM layers. There is a a trade off here, but the choice between accuracy and computational efficiency allows network operators to decide what is more practical for their needs. In their training, Krishnaswamy et al.

[11] noted that adding extra LSTM layers did not cause a noticeable change in accuracy. In fact, the baseline LSTM had the lowest Mean Squared Error overall. One limitation that will be investigated further in this project is whether these LSTM algorithms perform as well for smaller traffic volumes that one may see on an education network, as the results above were for links of capacity of over 100GB/s.

Cui et al. [6] investigated the use of a bidirectional LSTM for forecasting network traffic. A bidirectional LSTM (BDLSTM) is one that runs the input from both past to future, and future to past. This approach preserves information from the future and, using two hidden states combined, it is is able in any point in time to preserve information from both past and future [14]. Cui et al. [6] found that a stacked BDLSTM achieved a more accurate prediction. Importantly, the training times that they observed indicate that a BDLSTM is nearly double the training time of a regular LSTM. To our knowledge there was no mention of prediction times for a stacked BDLSTM, so we will investigate whether this increased prediction accuracy comes at the cost of a higher computational time compared to a stacked LSTM and a BDLSTM.

5 Experiment Design and Execution

5.1 Obtaining SANREN Data

The SANREN data was extracted from the SANREN server, in a pcap file format. This file format was likely generated by Wire-Shark which monitors network traffic and provides insights which can be used. Therefore in order to make the data easily accessible for our research purposes, it was converted into a text-file.

For our experiment, over 1GB of data was extracted. This data contains over 36500 flows and spans over 35 minutes on Tuesday the 4th of August 2020, between 7:50 and 8:25pm.

5.2 Preprocessing

The format of the data that was extracted was not suitable to be used without preprocessing. Some columns had null values, and different suffixes attached which made them difficult to read into the program. A pandas data frame was created to house the data, which also made it easier to work with.

5.2.1 Feature Extraction and Transformation.

In order to standardise the data, all traffic flows were converted into bytes to create a single unit of measurement. Week and weekend days were converted into a binary variable (week days corresponding to 0, weekend to 1), in order to be processed in the LSTM models.

The intention was to add South African University calendar to the preprocessing step, in order to visualise the change in network traffic over this period. However, this was not achieved and can be left for future work.

Other variables which were not suitable for use in an LSTM model were encoded using one-hot encoding.

5.2.2 Data sets.

The data was split using an 80/20 ratio, with 80 percent being the training set, and 20 percent for testing. This decision was made as it is inline with the recommendation of the Google Machine Learning Crash Course [1].

5.3 Evaluating Performance of LSTM Models

The LSTM models were evaluated during and after training, as well as on their performance predicting future traffic flows. Predictive accuracy and run time were assessed, in order to gauge how many resources are required to run these predictive models.

During training, the loss metric for each LSTM was specified to be 'MSE' (Mean Squared Error'). Test loss however is important, since it will determine whether the model has suffered from over fitting.

5.3.1 Prediction Metrics. Mean Squared Error (MSE) and Mean Absolute Error (MAE) were the measurements used to evaluate the predictive performance of the model.

MSE is a measurement of how close the models predictions are to the actual values, and is an ordinal value which allows the prediction accuracy of the different LSTM models to be compared.

$$MSE = (\frac{1}{n}) \sum_{i=1}^{n} (y_i - x_i)^2 \quad (1)$$

Figure 1 shows that the test MSE is an average of the errors observed when the LSTM makes predictions on unseen data. A low MSE would indicate that the model has learned the patterns of the data well, and is able to accurately predict what network traffic to expect in the future.

MAE measures the average magnitude of each prediction error for all predictions in the test set. Unlike MSE, it does not punish larger errors by squaring them.

$$MAE = (\frac{1}{n}) \sum_{i=1}^{n} |y_i - x_i| \quad (2)$$

Figure 2 shows how the Mean Absolute Error is calculated.

Both MSE and MAE are ordinal values, and therefore cannot provide meaningful information on their own [2]. Rather, it needs to be compared to the training MSE / MAE and compared with the test MSEs / MAEs of the other LSTM models.

5.3.2 Computational Efficiency Metrics. For a problem such as network traffic prediction, being able to quickly predict future traffic flows can be important, especially in the short term to aid dynamic resource allocation. Google Colab has a timer built in, which allows for the measure of how long the LSTMs take to train and to predict. Prediction times will likely be more important, since a long training time may provide predictions well into the future.

5.4 Overview of Experiment Design

Once the data has been pre-processed, it is ready to be used within a LSTM. The Bidirectional, Simple and Stacked LSTM models were all developed in Python. These were built with Keras API, which is an implementation of TensorFlow. This choice was made based on the ease of use and development of building deep learning models with these libraries. Once the models have been specified and compiled, they are then fitted to the data in the training stage. A graph showing the training loss per epoch is shown for each LSTM, which allows us by inspection to see when a models training loss is starting to plateau. Lastly, the LSTM models are tested on the test set, and their predictions against expected values are plotted. The predictions are then converted into megabytes, in order to be able to interpret them more easily. Accuracy metrics are also calculated in order to compare the different models.

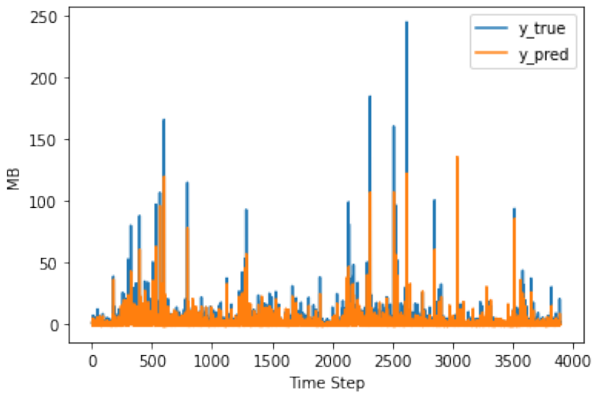
5.4.1 Implementation of LSTMs. The simple LSTM served as a benchmark with which to compare our Bidirectional and Stacked LSTMs. The training and prediction times, as well as the prediction accuracy served as a comparison with the more complex models.

All of the LSTMs trained and predicted using the same training and test datasets, to ensure that they were being compared as accurately as possible. Their activation functions were sigmoid, and their loss function was mean squared error. The Adam optimizer was also used for all the models, a stochastic gradient descent method which is suitable for network traffic prediction where we are concerned with memory constraints and have a large amount of data [10].

5.5 Hyper-parameter Tuning

6 Results

Figure 1. Simple LSTM Predictions



By visual inspection alone, it appears difficult to differentiate between the predictions made by the Bidirectional LSTM in and Simple LSTM in .

Figure 2. Bidirectional LSTM Predictions

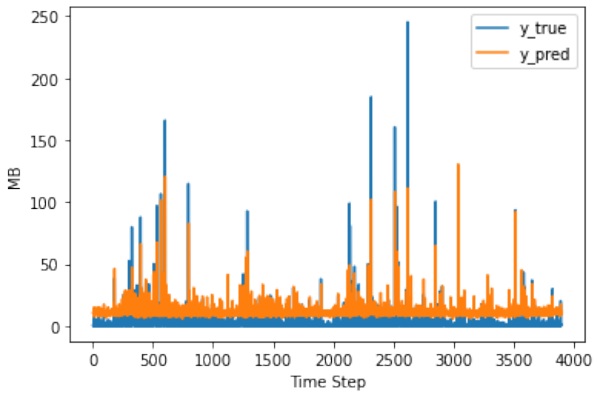


Figure 3. Stacked LSTM Predictions

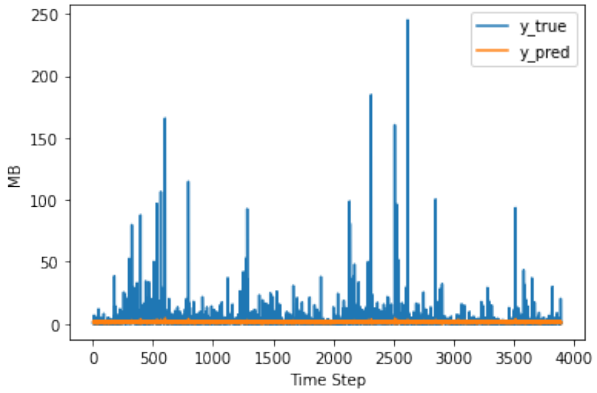


Table 1. Test Mean Absolute and Squared Errors for the LSTM Models (MB)

	Simple	Stacked	Bidirectional
MAE	4.5	2.5	7
MSE	116	87	145

Table 2. Training and Prediction Times (Seconds)

	Simple	Stacked	Bidirectional
Training	60.5	167.3	66.5
Prediction	2.2	4.7	2.9

7 Discussion

It appears from looking at the Stacked predictions in Figure 3, that the model does not seem to capture any of the network traffic bursts. However, the results show that in fact the Stacked LSTM is the most accurate predictor of our LSTM models. The caveat is that it is also the most computationally expensive of the models.

The bidirectional LSTM predictions made in Figure 2 all seem to be slightly higher than the true values, as if the graph has been shifted up. One possible reason for this, is since information is considered from both the past and the future, the model may be considering future flows which the Stacked and Simple LSTMs are not aware of yet.

8 Conclusions

In this paper, we set out to investigate using three different LSTM models for network traffic prediction on the SANREN. By comparing both prediction accuracy and computational efficiency metrics, we found that the Stacked LSTM was the most accurate method of future traffic prediction. The bidirectional LSTM appeared to be the least accurate, with a test MSE double that of the Stacked LSTM.

9 Limitations and Future Work

Due to the large volumes of traffic flows, predicting over long time periods was difficult since it would require analysing multiple gigabytes of data. This meant that the predictions made were only in the short term. A potential future fix is by clustering the traffic flows into intervals, which would allow for easier predictions in the long term.

10 Acknowledgements

I would like to acknowledge Prof. Chavula who guided us through this project and provided the necessary understanding and ideas behind how this project would be implemented.

References

[1] Training and test sets: Splitting data | machine learning crash course. URL https://developers.google.com/machine-learning/crash-course/training-and-test-sets/splitting-data.

[2]

[3] The south african nren. URL https://sanren.ac.za/south-african-nren/.

[4] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[5] Ibrahim G. Vallisn B. Böttger, T. How the Internet reacted to Covid-19 – A perspective from Facebook's Edge Network. *Proceedings of the ACM Internet Measurement Conference*, October 2020. doi: 10.1145/3419394.3423621.

[6] Zhiyong Cui, Ruimin Ke, Ziyuan Pu, and Yinhai Wang. Stacked bidirectional and unidirectional lstm recurrent neural network for forecasting network-wide traffic state with missing values. *Transportation Research Part C: Emerging Technologies*, 118:102674, 2020.

[7] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[8] Shan Jaffry. Cellular traffic prediction with recurrent neural network. *arXiv preprint arXiv:2003.02807*, 2020.

[9] Nan Jiang, Yansha Deng, Osvaldo Simeone, and Arumugam Nallanathan. Online supervised learning for traffic load prediction in framed-aloha networks. *IEEE Communications Letters*, 23(10):1778–1782, 2019.

[10] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[11] Nandini Krishnaswamy, Mariam Kiran, Kunal Singh, and Bashir Mohammed. Data-driven learning to predict wan network traffic. In *Proceedings of the 3rd International Workshop on Systems and Network Telemetry and Analytics*, pages 11–18, 2020.

[12] Rishabh Madan and Partha Sarathi Mangipudi. Predicting computer network traffic: a time series forecasting approach using dwt, arima and rnn. In *2018 Eleventh International Conference on Contemporary Computing (IC3)*, pages 1–5. IEEE, 2018.

[13] Nipun Ramakrishnan and Tarun Soni. Network traffic prediction using recurrent neural networks. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 187–193. IEEE, 2018.

[14] Mike Schuster and Kuldip Paliwal. Bidirectional recurrent neural networks. *Signal Processing, IEEE Transactions on*, 45:2673 – 2681, 12 1997. doi: 10.1109/78.650093.

[15] R Vinayakumar, KP Soman, and Prabaharan Poornachandran. Applying deep learning approaches for network traffic prediction. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 2353–2358. IEEE, 2017.

[16] Zheng Zhao, Weihai Chen, Xingming Wu, Peter CY Chen, and Jingmeng Liu. Lstm network: a deep learning approach for short-term traffic forecast. *IET Intelligent Transport Systems*, 11(2):68–75, 2017.