UNIVERSITY OF CAPE TOWN

DEPARTMENT OF COMPUTER SCIENCE

CS/IT Honours Final Paper 2021

Title: Deep Learning For Network Traffic Prediction

Author: Justin Myerson

Project Abbreviation: DL4NTP

Supervisor(s): Josiah Chavula

Category	*Min*	*Max*	Chosen
Requirement Analysis and Design	*0*	*20*	
Theoretical Analysis	*0*	*25*	5
Experiment Design and Execution	*0*	*20*	20
System Development and Implementation	*0*	*20*	
Results, Findings and Conclusions	*10*	*20*	20
Aim Formulation and Background Work	*10*	*15*	15
Quality of Paper Writing and Presentation	*10*		10
Quality of Deliverables	*10*		10
Overall General Project Evaluation (*this section allowed only with motivation letter from supervisor*)	*0*	*10*	
Total marks		**80**	

Deep Learning For Network Traffic Prediction

Using LSTMs on the South African Research and Education Network

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

Abstract

Network traffic prediction is an important tool for the South African Research and Education Network (SANREN) 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. Three different Long Short Term Memory (LSTM) models were developed for the SANREN: Bidirectional, Simple and Stacked. These models were evaluated both in terms of their prediction accuracy, and their computational complexity. Dataset size and prediction accuracy have a strong correlation, at the cost of increasing training time as more data are added. The results demonstrated that a Stacked LSTM was the most accurate prediction model, at the expense of using the most computational resources. Using a Simple LSTM greatly saved on computational resources, and was only slightly less accurate at predicting future traffic on the SANREN compared to a Stacked LSTM.

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

Keywords: Long-Short Term Memory, Mean Squared Error, Mean Absolute 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 COVID-19 has caused a significant surge in internet traffic [5], networks such as the SANREN are having to manage their limited bandwidth more effectively. Accurate network traffic prediction - provided by models that predict future traffic flows and fluctuation - would help the SANREN 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 [17]. 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 Long Short Term Memory models are now the gold standards for network traffic prediction. [13, 14, 16, 18, 20]. There has not been enough research done into using deep learning for short term network traffic prediction, and limited computational resources have also not been extensively considered.

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. Therefore, this paper also investigates the computational resources required for each LSTM implementation and concludes on the trade-offs between computation time, dataset size and prediction accuracy - specifically when considered for the SANREN use case, in order to more accurately predict future network traffic compared to current statistical methods used. Additionally, the SANREN needs to be able to determine how traffic flow on the network varies with the South African University Calendar.

1.1 Project Aims and Research Questions

The main aim of this project is to develop three LSTM models to test whether it is possible to accurately predict future network traffic on the SANREN, and to determine the computational complexity associated with training and making predictions using these models. Three research questions are also to be answered:

- How does the SANReN traffic data vary with time and day in relation to the South African university calendar?
- What is the computational cost of different LSTM architectures, given a required level of accuracy in predicting future traffic flows on the SANREN?

- Which of the LSTM models, Bidirectional, Simple or Stacked, provides the highest accuracy when predicting future SANREN traffic data, subject to network constraints?

Successful completion of this project will be achieved by answering these research questions, through the development and testing of the LSTM models, and the discussion generated from the experimental results obtained.

1.2 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 a trade off between accuracy and computational resources required to train the models arising. Existing literature on time series prediction has shown that LSTM derivative models, such as the Stacked and Bidirectional LSTM, have out-performed Simple LSTM models [6, 16].

The feasibility of LSTM and LSTM-derivative models to predict network traffic on the SANREN will be investigated. 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.

1.3 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 and limitations faced during the project.

While the overall objective of the research paper is to answer the research questions set out above, this paper will focus more on the LSTM architectures and their implementations, and discuss why certain results were observed. There will be a brief overview of the preprocessing and data analysis stages, but this will not be the main focus of this paper.

2 Background on Deep Learning and Related Work

Network traffic prediction approaches have been investigated in a multitude of past studies. Furthermore, the growth of the internet has accelerated research, with deep learning techniques emerging as the prevalent means 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 in their prediction accuracy. A Recurrent Neural Network can suffer from the vanishing gradient problem [4], 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. It is for this reason that Krishnaswamy et al. [16] propose that using an RNN is unsuitable for network traffic prediction.

An LSTM is a type of RNN, which is formed by adding a short and long term memory unit to an RNN [11]. 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 [21]. Krishnaswamy et al. [16] suggest that an LSTM should be used for time-series predictions, to eliminate the vanishing gradient problem.

2.1 Long Short Term Memory

LSTMs have cells in the hidden layers of the neural network, which have three gates: input, an output, and a forget gate [8]. These gates control the flow of information which is needed to predict the output in the network. The gates that are added to an LSTM cell allow LSTMs to learn long term dependencies, since they are able to retain information from multiple previous time-steps [11].

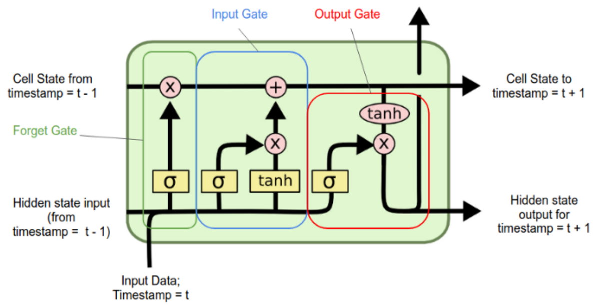

Input Gate
Output Gate
Cell State from timestamp = t - 1
Cell State to timestamp = t + 1
Forget Gate
tanh
σ
σ
tanh
σ
Hidden state input (from timestamp = t - 1)
Hidden state output for timestamp = t + 1
Input Data; Timestamp = t

Figure 1. An example of a LSTM Cell and its gates. Input is received from the previous cell and passes through the forget gates, where values deemed unimportant are dropped. Eventually new values are sent to the next cell through the output gate [12].

The sigmoid function, outputs a value between $0 < x < 1$. It is used in the forget gate, in order to decide what information from the previous time step should be retained and what should be discarded in the next cell state. A value of 0 means that no information will pass through to the next cell, while 1 means that all information is passed through. The tanh function, outputs a value between -1 < x < 1. It forms the input gate, and receives information from the forget gate. When the input gate receives information from the previous time step C_{t-1}, it then determines whether the input will contribute to the cell state.

$$g_t = \sigma(W(h_{t-1}, X_t) + b_i \tag{1}$$

$$C_t = tanh(W(h_{t-1}, X_t) + b_C \tag{2}$$

$$h_t = o_t * tanh(C_t) \tag{3}$$

g_t indicates whether the information will pass to through to the output gate, while C_t calculates the value of the new cell state. Lastly, the result of the output from the cell unit is calculated. h_t represents the cell output, and is calculated by multiplying the current cell state (C_t) by the important features from the input (o_t) [7].

2.1.1 Bidirectional.

A Bidirectional LSTM runs 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 [19]. Cui et al. [6] investigated the use of a Bidirectional LSTM for forecasting network traffic. They also found that a Stacked + Bidirectional LSTM achieved a more accurate prediction than a basic Bidirectional LSTM. Importantly, the training times that they observed indicate that a Stacked Bidirectional LSTM has nearly double the training time of a regular LSTM.

2.1.2 Stacked.

A Stacked LSTM puts multiple LSTM cells on top of each other, which forms multiple LSTM layers [7]. These layers provide a sequence of outputs to the next layer, rather than a single value. Krishnaswamy et al. [16] discussed the differences between using a Simple and Stacked LSTM learning approach for network traffic prediction. Stacked LSTM's were more accurate in predicting future traffic flows compared to the Simple LSTM architecture. This comes at the cost of a higher computational complexity, as a result of added LSTM layers that the Simple LSTM does not have. 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. Krishnaswamy et al. [16] noted that adding extra LSTM layers did not cause a noticeable change in accuracy. The Simple LSTM had the lowest Mean Squared Error overall. One limitation that is investigated further in this project is whether these LSTM algorithms perform as well for smaller traffic volumes such as those on SANREN, compared to the traffic links that process over 100GB/s.

3 Experiment Design and Execution

3.1 Test Bench

The code was developed and tested on a 2016 MacBook Pro with the following specifications:

- 8GB of 1866MHz LPDDR3 RAM
- Intel I5-6360U 2.0GHz
- Intel Iris Graphics 540

3.2 Obtaining SANREN Data

The SANREN data was extracted from the SANREN server in a pcap file format. In order to make the data easily accessible for our research purposes, it was converted into a text-file. Traffic flows from 5 consecutive days (5 - 10 July 2020) were extracted. Having multiple days of data allows for the comparison of how traffic flow changes based on day, and may impact on the training and prediction of models if the data is correlated. Additionally, it follows our hypothesis that more data would lead to an increase in prediction accuracy.

3.3 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. Additionally, there were summaries of traffic flows extracted that needed to be removed.

3.3.1 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. Other variables, such as flags, flows and Tos were dropped. The remaining variables, source and destination IP addresses were formatted to make them suitable for use in an LSTM model. The South African University calendar for 2020 was added as well, in order to visualise the change in network traffic over this period.

3.3.2 Datasets.

The data was split using an 80:20 ratio. 80% of the data for the training set, and 20% for the testing set. Of the 80% used for the training set, another 80:20 split was applied. 80% of the split remained as a training set, while the other 20% was used to create a validation set [9]. Using less training data would mean that the parameter estimates would have higher variance, while less test data means that performance metrics would have higher variance. The final predictions will be made on the test dataset once the models have been trained.

A validation set forms an important part of ensuring that the models do not suffer from over fitting, as it shows how well the model is generalizing during training. These data are separate from the training set, and should have the same distribution as the training set. While the model will train on the data from the training set, it will be evaluated on the predictions it makes using validation set in order to help fine tune the hyperparamters. The model does does not update weights based on errors it encounters during the validation process, only those encountered on the training set.

3.4 Evaluating Performance of LSTM Models

The LSTM models were evaluated during and after training. During training, the training and validation losses were measured. When predicting future traffic flows, they were then assessed on their prediction accuracy. Both accuracy and computational resources required were assessed, in order to answer the research questions.

During training, the training and validation loss metric for each LSTM was specified to be Mean Squared Error (MSE). MSE as well as R^2 were used to evaluate the final predictions the LSTMs generated on the SANREN test data.

3.4.1 Prediction Metrics. Mean Squared Error (MSE), Mean Absolute Error (MAE) and the Coefficient of Determination were the measurements used to evaluate the predictive accuracy 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.

$$MSE = (\frac{1}{n})\sum_{i=1}^{n}(y_i - x_i)^2 \tag{4}$$

Equation 4 shows that the test MSE is an average of the errors observed when the LSTM makes predictions on unseen data.

MAE measures the average magnitude of each prediction error for all predictions in the test set. It is more robust to outliers than MSE, since it does not punish large errors by squaring them.

$$MAE = (\frac{1}{n})\sum_{i=1}^{n}|y_i - x_i| \tag{5}$$

Equation 5 shows that the test MAE is the average of the absolute errors observed when the LSTM makes predictions on unseen data.

Both MAE and MSE are ordinal values, and therefore cannot provide meaningful information on their own [1]. Rather, the training or test metrics need to be compared with the metrics produced by the other LSTM models. A low validation MAE and MSE would indicate that the model has generalised well to the data, while a low test MAE and MSE would indicate that the model made accurate predictions.

$$R^2 = 1 - RSS/TSS \tag{6}$$

R^2, the Coefficient of Determination is a metric which measures the proportion of the variation in the dependent variable that is predictable from the independent variables [2]. In this case, it measures how well the model captures variation in the data. The closer the R^2 value is to 1 the better, since a value of 1 implies the model has fitted the data perfectly. RSS refers to the Residual Sum of Squares which is the sum of the errors squared, while TSS refers to the Total Sum of Squares which is the sum of the squared differences between actual and average value of y.

3.4.2 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. In order to track how long each LSTM takes to train and make predictions, the Python *time* module was used. The number of seconds each model took to train with different hyperparamters was then easily comparable. A long training time can be a hindrance for the SANREN, especially if they want to make short term predictions and have a limited time frame and resources.

3.5 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 and validation loss during training is shown for each LSTM, which allowed to see where the validation loss reached a minimum. Once the lowest validation loss had been found, the hyperparamters of that LSTM were noted. This LSTM is then used to make predictions on the unseen SANREN test set. Their predictions against expected values are plotted in order to visualise their accuracy. During both training and testing, accuracy and computational resource metrics are also calculated in order to compare the different models and are then written to an external file. Post training and testing, the results are analysed and plots are generated in order to visually compare performance of the LSTMs.

3.5.1 Implementation of LSTMs. Previous work discussed in Section 2 suggests that of the models that are investigated in this paper, the Stacked LSTM is the most accurate predictor of future network traffic. This however comes at the cost of having the highest computational complexity as well. We therefore expect the findings of this paper to be reflective of the previous work discussed. Our two results, computational complexity and prediction accuracy will allow for a network service provider to decide what LSTM model is more suited to their needs. Therefore, the impact of the product will fall primarily on National Research and Education Networks (NRENs). If the LSTM models developed are found to be an accurate means of predicting future network traffic, and are computationally viable for the SANREN use case, then NRENs will be able to implement LSTMs as network network traffic predictors in order to manage network load, security and resource use. Additionally, if LSTMs are found to be sufficiently computationally cheap, then the application of them as a network traffic predictor could be applied to additional low-resource networks.

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. Choosing a Simple LSTM to be the baseline predictor was a decision made from the work discussed in Section 2, as it was found that the Simple model had the least computational complexity of the LSTMs developed and still outperformed the prediction accuracy of other deep learning methods. For these reasons, the Simple LSTM was chosen to be the baseline predictor with which to compare the Bidirectional and Stacked 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. They were developed using Keras, and were a sequential model which means each layer has one input and output Tensor. Their validation and training loss function *MeanSquaredError*. 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 [15]. An activation function is required between matrix multiplications to afford a neural network, or in this case an LSTM the ability to model nonlinear processes. Internally, LSTMs have activation functions built into them that occur within the cells, as can be seen in Figure 1. Therefore, it was deemed unnecessary to add any new activation functions.

During training, history of the training and validation loss per epoch is stored and then plotted. This allows for the models to be evaluated on training and validation loss. Multiple metrics, including *MAE*, *MAE* and R^2 were calculated at the end of the training and prediction phases too, in order to gain more insights into how the models were performing during the process and to be able to conclude on the most accurate model.

3.6 Hyper-Parameter Tuning

Using a validation set helps to determine how each model will behave on data which it has not been trained on. When the validation error reaches a minimum, this is when training will be stopped, since the model has trained as well as it can without starting to overfit to the training data. Continuing to train the data may lead to decrease in training loss, however, this would mean the data is not generalizing well, but rather is fitting to the noise of the data [10]. In order to compare the performance of the different models, they each needed to be trained and evaluated with different hyper-parameter in order to find the lowest possible validation loss. Once found, the hyper-parameters that led to the model reaching a minimum validation loss in training will be used to fit the model. This model will then be used to make predictions.

Different dataset sizes allowed us to determine whether more data increases accuracy and it's effect on training and prediction time. The inclination is that more data would increase prediction accuracy, since the models have more data to train on, but would also lead to an increase in training and prediction times. Larger amounts of data would mean outliers, traffic bursts in this case, would have less of an effect on the models, since their variation would be averaged out more and should not reduce accuracy.

Epochs signify how many times the model cycles through the full dataset. Increasing the number of epochs should then help the model to generalise better to unseen test data during the testing phase. However, too many epochs can cause the model to start over fitting, fitting to the noise rather than the underlying pattern of the data. The number of epochs which lead to the lowest validation loss will be used.

Neurons represents the dimension of the output. They are not independent, and communicate with each other. Increasing the number of neurons increases the number of parameters that are passed onto the next LSTM cell, which will increase computational complexity and should increase accuracy.

4 Results

4.1 Optimum Parameters

Since the *MSE* values were so small, the accuracy metrics were also calculated for in terms of Root Mean Squared Error (RMSE) in order to make the results more interpretable.

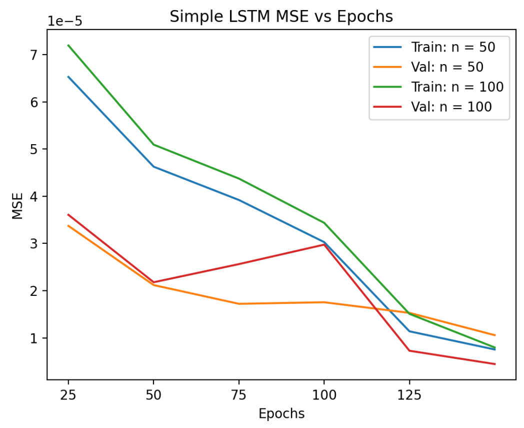


Figure 2. Simple LSTM Training and Validation Loss

Table 1. Simple LSTM Validation Results

Epochs	**Neurons**	R^2	*RMSE*
150	100	0.98	0.00632

The Simple LSTM reached a maximum validation R^2 and a minimum validation loss at 150 epochs and 100 neurons.

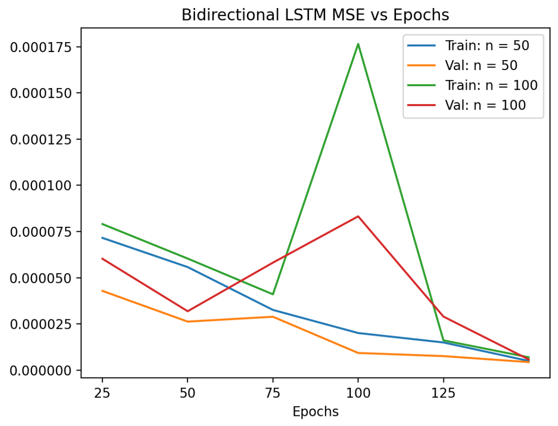


Figure 3. Bidirectional LSTM Training and Validation Loss

Table 2. Bidirectional LSTM Validation Results

Epochs	**Neurons**	R^2	*RMSE*
150	50	0.98	0.00223

The Bidirectional LSTM reached a maximum validation R^2 and a minimum validation loss at 150 epochs and 50 neurons.

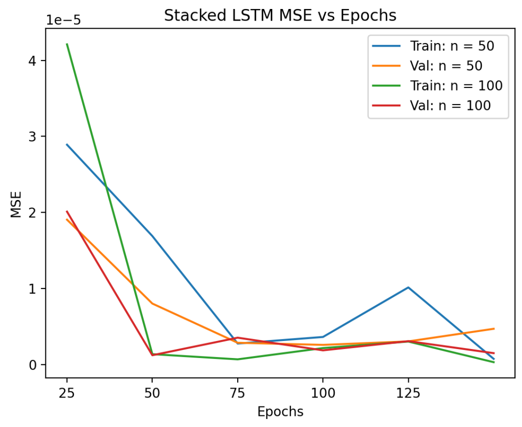


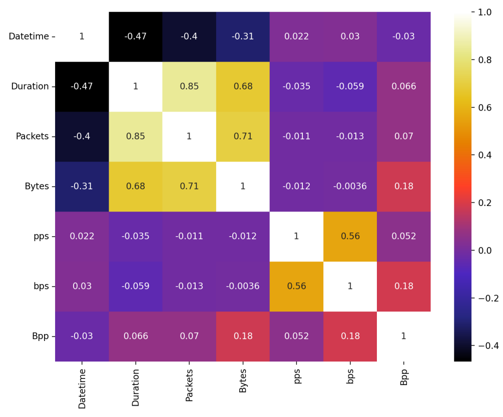
Figure 4. Stacked LSTM Training and Validation Loss

Table 3. Stacked LSTM Validation Results

Epochs	**Neurons**	R^2	*RMSE*
100	50	0.99	0.00948

The Stacked LSTM reached a maximum validation R^2 and a minimum validation loss at 100 epochs and 50 neurons.

4.2 Preliminary Analysis



	Datetime	Duration	Packets	Bytes	pps	bps	Bpp
Datetime	1	-0.47	-0.4	-0.31	0.022	0.03	-0.03
Duration	-0.47	1	0.85	0.68	-0.035	-0.059	0.066
Packets	-0.4	0.85	1	0.71	-0.011	-0.013	0.07
Bytes	-0.31	0.68	0.71	1	-0.012	-0.0036	0.18
pps	0.022	-0.035	-0.011	-0.012	1	0.56	0.052
bps	0.03	-0.059	-0.013	-0.0036	0.56	1	0.18
Bpp	-0.03	0.066	0.07	0.18	0.052	0.18	1

Figure 5. Correlation Matrix of Input Features

Figure 5 above shows the correlation between input features that were used. This was a useful step in order to visualise the data, and understand what dependencies variables had.

While not investigated here, removing variables with low correlations may reduce computational complexity. Notable correlations were between duration and packets, duration and bytes and packets and bytes. Intuitively this makes sense, since we expect a traffic flow to last longer if it contains more packets or bytes. Bytes and packets also correlate, since more bytes would require more packets to transport the load.

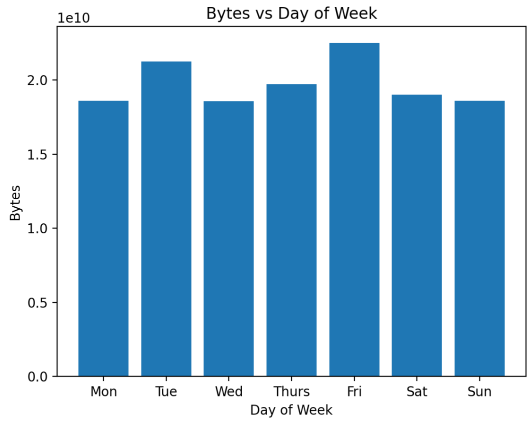


Figure 6. Total Traffic Flow vs Day of Week

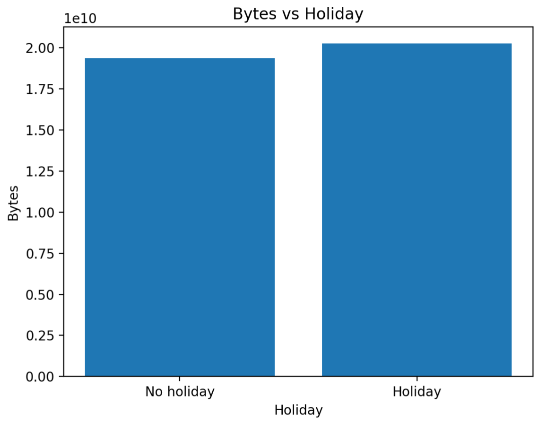


Figure 7. Total Traffic Flow vs Holiday / University Term

In Figure 6, Friday and Tuesday have the highest amount of network traffic. However, all of the days seem to have similar amounts of total network traffic that they experience. Additionally, there was 5% more network traffic observed during the holiday than during a regular university time as is seen in Figure 7.

4.3 Training and Prediction Time

The results in this section illustrate the effect of increasing the dataset size on training and prediction time.

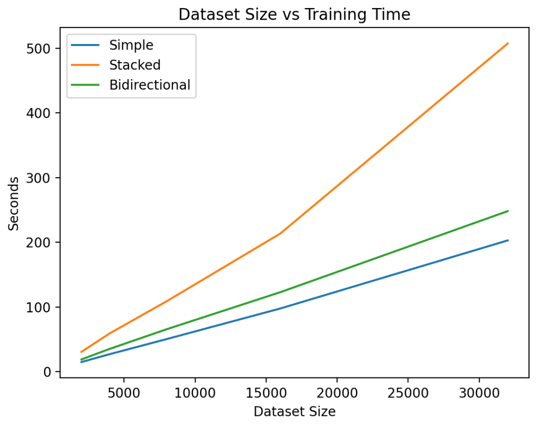


Figure 8. Dataset Size vs Training Time

Figure 8 illustrates how increasing the size of the training set, by including more traffic flows, increases the training times of all three LSTM models. The Simple and Bidirectional LSTMs are fairly close in their training time which grows linearly, and are consistently quicker to train than the Stacked LSTM. Furthermore, the Stacked LSTM training time grows faster after 16000 traffic flows, taking over 8 minutes to train on the largest dataset.

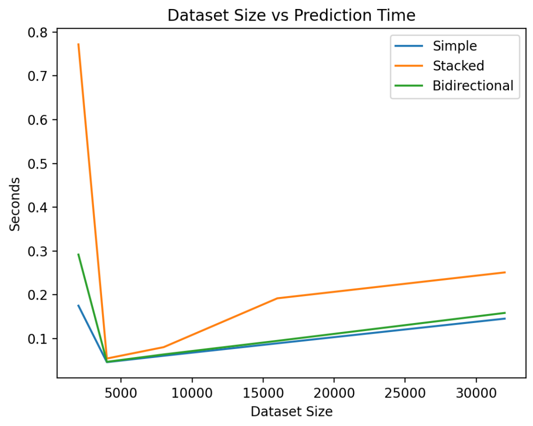


Figure 9. Dataset Size vs Prediction Time

Figure 9 illustrates the effect of increasing the dataset size on the time taken for the models to make predictions. The Stacked LSTM consistently takes the longest to make predictions while the Bidirectional and Simple predictors are almost indistinguishable in their prediction times for all dataset sizes. When dealing with dataset sizes of this magnitude, it appears the prediction times in practice would be

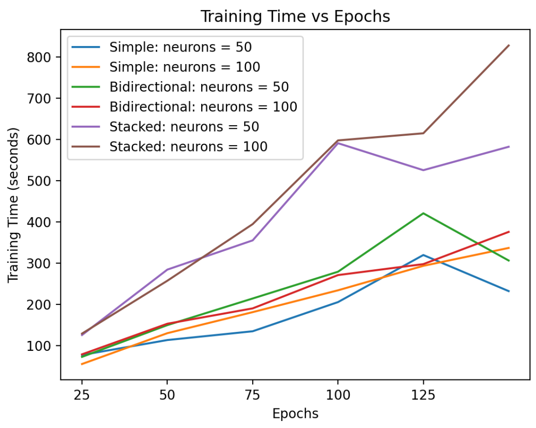


Figure 10. Training Time vs Epochs

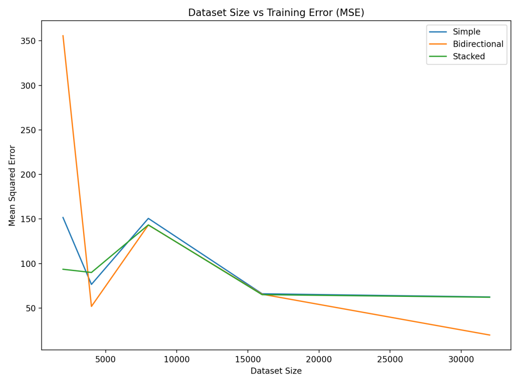


Figure 11. Dataset Size vs Training Accuracy

negligible.

In Figure 10, the training time for all three models is compared with 50 and 100 neurons, with a dataset size of 48000 traffic flows. The Stacked LSTM consistently takes longer to train than the Simple and Bidirectional models, which is consistent with other results we have seen. Between 25 to 100 epochs, training time increases linearly for all of the LSTMs. With more than 100 epochs, the Stacked LSTM with 100 neurons starts taking on order of magnitude minutes longer to train than model with 50 neurons.

Lastly, it can be seen in Figure 11 that training loss (MSE) tends to decrease with the dataset size. This was measured as an average across 20-100 epochs at each dataset size. There is a small uptick in training loss between 4000 and 8000 data points, which will be discussed later on.

4.4 Prediction Accuracy

In order to evaluate the predictive accuracy of the Bidirectional, Simple and Stacked LSTMs, they were each initially trained using the following hyper-parameters which led to the lowest validation loss, as seen in Tables 1, 2 and 3.

Table 4. LSTM Hyper-Parameters

LSTM	Epochs	Neurons
Bidirectional	150	50
Simple	150	100
Stacked	100	50

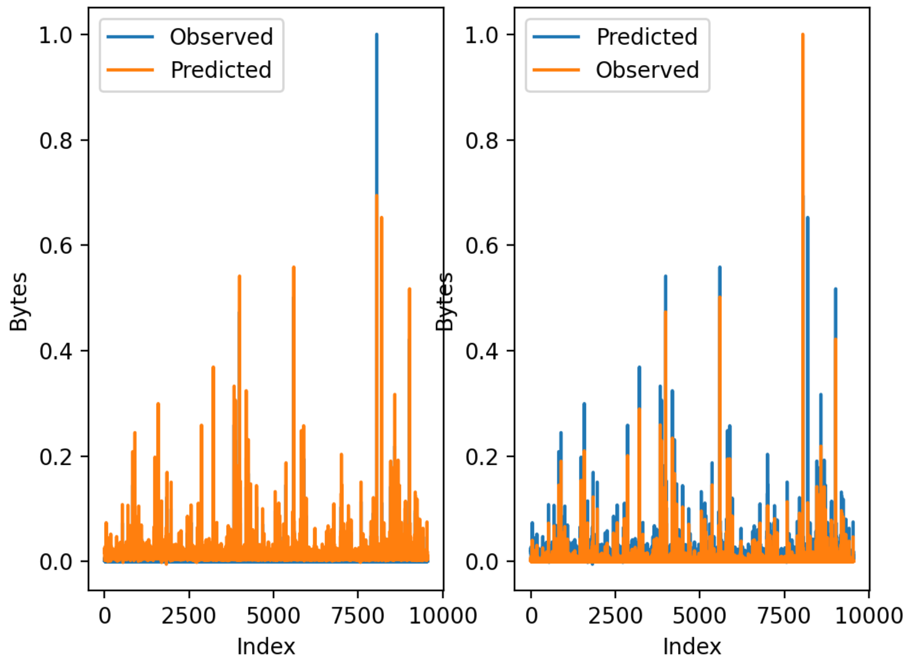


Figure 12. Simple LSTM Predictions

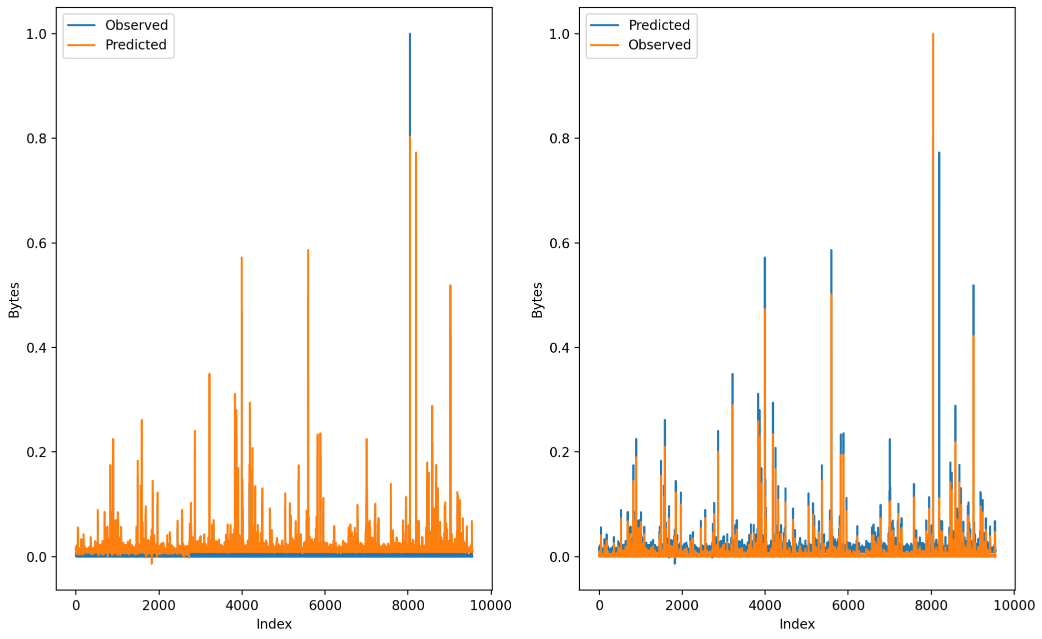


Figure 13. Bidirectional LSTM Predictions

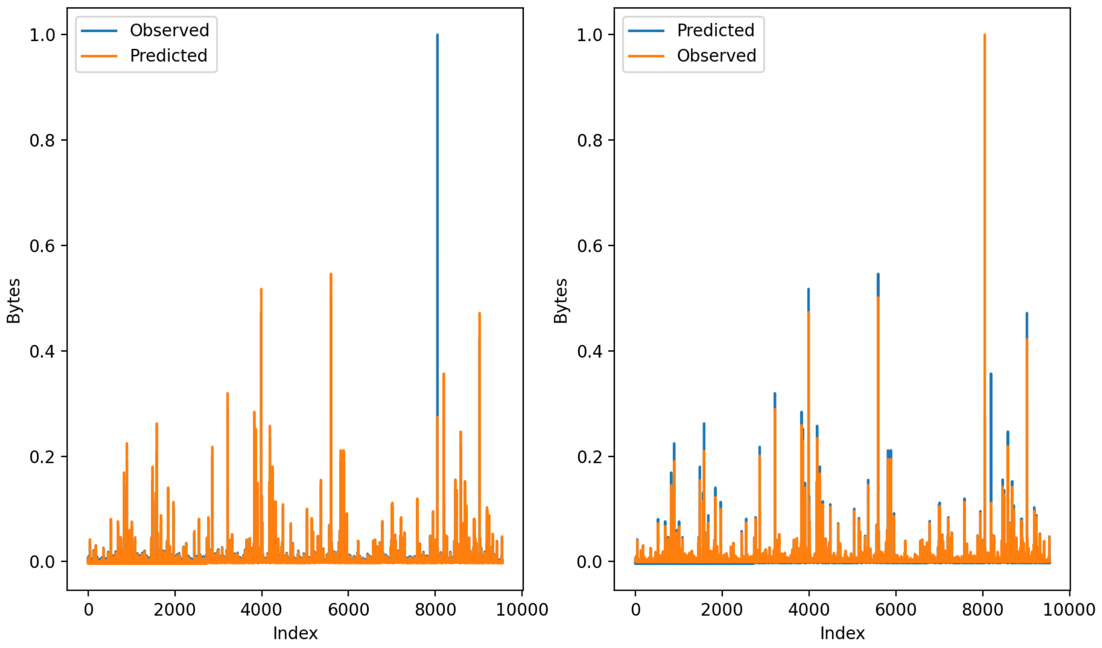


Figure 14. Stacked LSTM Predictions

Figures 12, 13 and 14 illustrate the predictions made by each optimised LSTM, with hyper-parameters observed during training. The predictions on the left and right are inverted, in order to more easily be able to visualise when the models over or under predict.

The Simple LSTM captures 65 percent of the variance of the data, as seen below in Table 5. It appears that the model is over predicting, but has managed to capture the general pattern of the time series data. In terms of the Bidirectional LSTM, it less accurate in learning the characteristics of the network traffic data, since it had the lowest R^2 value. For both the Simple and Bidirectional models, seeing a large traffic burst at time step 8000 led to the models both massively over predicting for time step t_{i+1}. While the Stacked model also suffered this over prediction, it was far less severe.

Model	MAE	MSE	R^2	Training (s)
Bidirectional	0.0078	0.00013	0.55	306.60
Simple	0.0063	0.00010	0.65	**293.67**
Stacked	**0.0029**	**0.00006**	**0.80**	591.20

Table 5. Model Performance Using Optimum Hyper-Parameters - Test Set

The highlighted values in Table 5 illustrate the best result. For prediction accuracy, the Stacked LSTM outperformed the Simple and Bidirectional LSTMs respectively. It had the lowest Mean Absolute and Squared Error, as well as the highest R^2 value. However, it also had the highest training time, taking close to 2x longer to train than the Bidirectional LSTM. The Simple LSTM had the lowest training time, and made more accurate predictions than the Bidirectional LSTM, measured in both MAE and MSE. Additionally, it captured more of the variance of the model with an R^2 score of 0.65 compared to the Bidirectional R^2 of 0.55.

5 Discussion

The first research question we aimed to answer was how network traffic observed varies based on the day of the week and with university holidays. The results showed that there is little change in the traffic based on day, or whether is a university term day. More data would be required to create a strong argument as to why this is the case, however it is possible that students and lecturers were indifferent with when they interacted on the network. The high amount of traffic during the holiday could also be explained by students having deadlines during the holiday, or those who were taking the time to catch up work.

One of the reasons the Stacked LSTM has much higher training times for all dataset sizes compared to the Simple and Bidirectional LSTMs, taking 500 seconds to train with 100 epochs on 32000 traffic flows compared to 200 seconds for the other models, is that the Stacked model has multiple LSTM layers. This adds an overhead, since it is essentially sequentially training Simple LSTMs. It is likely that increasing the dataset size beyond 32000 traffic flows will continue to increase the training time, inferring from the previous result seen in Figure 8, and this is something that network providers need to be cognisant of. The low prediction times seen across all the models, stems from the fact that we are essentially feeding an algorithm a set of values, and it is applying a complex mathematical function that has already been generated during training. However, inferring from the trend seen in Figure 9, it is likely that the Stacked model will consistently take the longest to make predictions. Datasets of larger magnitudes could increase the gap in prediction times between the Stacked and the other LSTMs as well.

Increasing the amount of data available to the LSTM models had the effect of increasing prediction accuracy. In a time series approach, such as network traffic prediction, more data means more historical knowledge that can be used to infer future events. Additionally, more data means that the models will see more fluctuations which it can add to its understanding during training. This is what was observed in figure 11, where the general pattern is that the training and validation error decreased as more data was fed to the models. However, it appears between 4000 and 8000 traffic flows that training loss increased. One reason for this may be that there was a large amount of burst flows during that time, which caused the Mean Squared Error value to increase, since it heavily punishes outliers due to its squared nature. The initial high *MSE* values observed are again attributable to the fact that with a small amount of data, outliers can have a large impact on the error rate.

Changing the number of epochs the models trained on had the effect of dropping the accuracy observed in the prediction phase. it could have been caused by outliers that occurred in the extra data since the model may have started to suffer from over-fitting. This occurs when the model learns the data too well, and starts to fit to the noise of the data, not the underlying pattern. This is likely what happened in the case of the bidirectional model, which had the lowest training error by far compared with the simple and stacked models. However, it produced the least accurate predictions on the test set, indicating that it had suffered from over-fitting. This can cause lower prediction accuracy in the test set. One way to verify if this was the case would have been to create a validation set as well, but this will be left for future work.

The Bidirectional model seemed to fit the training data very well, and had the lowest validation loss of all the LSTM models. One possible reason for this, is since information is considered from both the past and the future, it may have considered future flows which the other models were not aware of yet. Compared to the Simple LSTM, it took slightly longer to train, yet this did not translate to a more accurate prediction. During the prediction phase, the Bidirectional LSTM had a higher error rate, and a lower R^2 compared to the Simple LSTM. Since it is seeing data from the future as well, the model is likely over-predicting fluctuations in the network traffic which causes the decreased accuracy. The results show that there is no benefit to using the Bidirectional model for training and predicting network traffic on the SANREN, as its complexity does not yield an improvement in prediction accuracy compared to the Simple LSTM.

By default, the number of time steps the LSTM models considered was one. This means that only one lagged observation is considered. Ideally, adding more time steps means that the model would have more context, and would therefore provide a more accurate set of predictions. However, when implementing time steps in the SANREN use case, the predictions became far less accurate. A possible reason for this decrease in accuracy is that the models miss out on subtle time dependencies that are not considered as time steps increase.

The Stacked LSTM model has the highest R^2 prediction value, 0.8, indicating that the model captures the most variance of all three models. It also had the lowest prediction error, which shows that the model generalised the best to new and unseen data, learning the overall pattern and not the noise. Figure 14 also shows how close the observed and the predicted values are. Since the Stacked LSTM has more layers of LSTM cells compared to the Bidirectional and Simple models, it can learn and pass on dependencies better to the next layer of the LSTM, further solidifying it's understanding of the data.

6 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. A potential future fix is by clustering the traffic flows into intervals, either hours or days which would allow for easier predictions in the long term since training is so expensive, especially in the case of the Stacked. Another limitation was that the time-step parameter which we unable to effectively implement. When the time step was changed, it drastically increased training time and the prediction accuracy dropped significantly. Ultimately, it was removed from the code since it made running the models extremely expensive and strained our resources. Changing this may have helped with prediction accuracy since the model could have additional context from previous observations. Removing some input features that do not impact the prediction accuracy may also be useful, since it could reduce the computational complexity of the models.

7 Conclusions

Traffic flow on the SANREN does not vary in any meaningful way on holiday or regular university days. Additionally, there was minimal change in the total traffic that flowed on the SANREN on different days of the week.

The results show that the Stacked LSTM is the most accurate method of future traffic prediction on the SANREN use case. However, subject to network constraints, the Simple LSTM is the best choice for a network provider, since it was a least complex model in terms of training and prediction time. The Stacked LSTM had an R^2 of 0.8 compared to 0.65 and 0.55 of the Simple and Bidirectional LSTMs respectively. Furthermore, the Stacked LSTM had the lowest prediction *MAE* and *MSE*. The Bidirectional LSTM is the least accurate predictor, even though it was a more complex model, taking longer to train and make predictions with compared to the Simple LSTM. Therefore, it is not recommended in any situation to be used.

Prediction time was not impacted by dataset size in any meaningful way, since the measurement was in milliseconds. However training time was consistently higher for the Stacked LSTM compared to the other two models, for all dataset sizes and number of epochs. An increase in dataset size led to an increase in training and validation accuracy for all three models.

If accuracy is most important without considering computational complexity, then a Stacked LSTM will be chosen since it has the lowest prediction error and highest R^2. The Simple LSTM will be a better choice if limited computational resources are involved, since it has a 2x lower training time and still makes more accurate predictions than the Bidirectional LSTM.

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. Additionally, a special mention to my partner Antony Fleischer who was instrumental in the development of this project.

References

[1] What is the difference between categorical, ordinal and interval variables? URL https://stats.idre.ucla.edu/other/mult-pkg/whatstat/what-is-the-difference-between-categorical-ordinal-and-interval-variables/.

[2] 2.8 - r-squared cautions. URL https://online.stat.psu.edu/stat462/node/98/.

[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] Xunsheng Du, Huaqing Zhang, Hien Van Nguyen, and Zhu Han. Stacked lstm deep learning model for traffic prediction in vehicle-to-vehicle communication. In *2017 IEEE 86th Vehicular Technology Conference (VTC-Fall)*, pages 1–5. IEEE, 2017.

[8] Alex Graves. *Long Short-Term Memory*, pages 37–45. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-24797-2. doi: 10.1007/978-3-642-24797-2_4. URL https://doi.org/10.1007/978-3-642-24797-2_4.

[9] Isabelle Guyon et al. A scaling law for the validation-set training-set size ratio. *AT&T Bell Laboratories*, 1(11), 1997.

[10] Jeffrey Heaton. Ian goodfellow, yoshua bengio, and aaron courville: Deep learning: The mit press, 2016, 800 pp, isbn: 0262035618. *Genetic Programming and Evolvable Machines*, 19, 10 2017. doi: 10.1007/s10710-017-9314-z.

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

[12] Ryan T. J. J. Lstms explained: A complete, technically accurate, conceptual guide with keras, Sep 2021. URL https://medium.com/analytics-vidhya/lstms-explained-a-complete-technically-accurate-conceptual-guide-with-keras-2a650327e8f2.

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

[14] 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.

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

[16] 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.

[17] 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.

[18] 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.

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

[20] 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.

[21] 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.