Deep Learning For Network Traffic Prediction

Project Proposal

Antony Fleischer
flsant005@myuct.ac.za
University of Cape Town
Cape Town, South Africa

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 more accurately than traditional prediction models. Of these approaches, Long Short Term Memory (LSTM) appears to be the most accurate method. However, there is little research on the computational resources required to run deep learning models, and whether they can be optimised for large networks. This paper aims to determine whether a standard LSTM is an appropriate architecture for network traffic prediction on the South African Research and Education Network (SANReN), or whether applying a stacked or bidirectional LSTM yields more accurate results. Furthermore, this paper will evaluate whether these evolutionary approaches are viable for less-resourced systems and whether they perform well on less intensive network traffic.

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 [7]. 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 [14]. 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.

SANReN is an organised network of education and research institutions within South Africa [1]. 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.

2 Problem Statement

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, 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 Long Short Term Models (LSTM) - provide improvements in accuracy and performance over traditional statistical prediction methods. Furthermore, LSTM derivative models, such as the stacked LSTM and bilateral LSTM, have out-performed baseline LSTMs.

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.

2.1 Research Questions

1. How does the SANReN traffic data vary with time and day in relation to the South African university calendar?
2. Which of the LSTM architectures, baseline, bilateral or stacked, provides the highest prediction accuracy, subject to network constraints?
3. What is the computational cost of different LSTM architectures given a required level of accuracy?

3 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. [12] 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 [9]. 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 [19]. Additionally, to control the use of historical information, the model uses an in, forget and out gate.

Krishnaswamy et al. [12] 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. [12] 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. [8] 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 [17]. Cui et al. [8] 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.

4 Procedures and Methods

4.1 Network Traffic Data

The first step in conducting our research is to obtain time-series data from SANReN. This data will be used to train, validate and test the neural networks that this paper evaluates. The data set consists of a multitude of files, each describing traffic flows on the network over a period of time. The SANReN data has been placed on a data store at the University of Cape Town, which we will access remotely.

4.2 Data Preprocessing

To use the SANReN dataset, the raw data has to be extracted and cleaned for a neural network model [2]. This preprocessing step will transform raw data into input for each of the neural network models. This section describes the preprocessing methods that will be applied.

4.3 Data Engineering

The network traffic received from SANReN is already stored in CSV format, with labels. The data will be checked for missing values, if any exist, these values will be removed. Sometimes categorical features are represented numerically, however, this is not recommended [3] and one hot encoding will be used to represent such data correctly. Additionally, to capture the relationship between university schedules and network traffic data, day of the week fields will be added to the data set. These fields will be encoded so that all of the flows on a Monday have a Monday value equal to 1, whilst flows not on Monday will have a Monday field equal to 0. This will be applied so that all days of the week are represented.

Additionally, some of the features in the provided data may also be highly correlated. This will be examined using a correlation matrix, and some fields may be removed to avoid multicollinearity. Multicolinearaity may affect how the LSTM models weigh different inputs, and so it is advisable to remove highly correlated variables [3]. The data will also be examined for outliers. However, since we want the LSTM

to be able to predict burst flows, those that do not follow the long-term temporal pattern, outliers will not be removed.

4.4 Preliminary Statistical Analysis

To provide insight into the underlying patterns, correlations and composition of the SANReN data, a preliminary statistical analysis will be performed. This will include a data description of the SANReN data, relationships of interest such as volume versus hour and volume versus hour, and a descriptive statistics table. Linear regression models will also be implemented to further describe the patterns in the data. This will be done using pandas - a Python library for data analysis - and NumPy - a Python library for high-level mathematics.

4.5 Establishing an LSTM Baseline

A minimum level of prediction performance must be defined against which other models can be assessed. To do this, we will implement a traditional LSTM model as the deep learning performance baseline. Existing literature provides evidence that LSTM models are more accurate general recurrent neural networks [10, 18] than traditional statistical prediction models - such as ARIMA [15]. Therefore an initial baseline will not be implemented for the traditional LSTM. However, evidence to support traditional LSTM's superior performance will be presented. The traditional LSTM itself will set the bar for the stacked-LSTM and bilateral LSTM models. Furthermore, because more sophisticated LSTM architectures - the stacked and bilateral LSTM - are being implemented, we have decided to evaluate them against their simpler counterpart. The stacked and bilateral models will have to exhibit a statically significant improvement in prediction accuracy or computational cost to justify their additional complexity.

4.5.1 Long-term Temporal Patterns. Time series data exhibits sequential pattern behaviour. Essentially, a new data observation is dependent on past values in the time series. The LSTM model is designed to capture these long-term temporal dependencies [15], and has therefore been selected to predict sequential patterns in the SANReN data. LSTM models are ideal for this task because of the structure of their memory units. Each unit gives the model the ability to consider its input, and then either keep existing memory or overwrite it with new information [10]. This capacity allows extremely long-term temporal dependencies to be captured by the model, whereas simpler RNNs are unable to capture these trends [15]. Additionally, LSTMs have become the high-performing standard for network traffic prediction [10, 11, 15, 18] and so it is sensible to implement a traditional LSTM for this project. Bidirectional and Stacked LSTM architectures are more suited to dealing with long-term temporal patterns [5] [6], since they both include more layers than a traditional LSTM.

Network traffic data may also be provided to a neural network to predict when a burst in traffic volume will occur, rather than to forecast a sequential pattern in the time-series data [13]. Burst traffic is defined as a prolonged, uninterrupted transfer of data from one device to another. When the sequential pattern of the data is non-linear, and burst traffic is also present, neural network models have shown to be 78% more accurate than traditional statistical prediction methods [18]. Therefore, this paper will aim to use a large sample of non-linear SANReN data with burst flow activity.

4.6 Hyper-parameters

Hyper-parameters in deep learning are parameters that are defined before a model is trained. Therefore, these parameters cannot be adjusted by the model as it learns more about the dataset. Examples of hyper-parameters in an LSTM include the memory unit topology, the train-test split, the learning rate of the model and the number of training epochs it is given to train. In a neural network, a training epoch is defined as one full cycle of data through the model. Since hyper-parameters such as the training epochs are predefined, there is motivation to alter these values to optimise the performance of each model.

4.7 Model Feasibility Evaluation

In order to evaluate the accuracy of the LSTM models that will be developed, a training and test dataset will be created. The LSTM architectures will be trained using the training dataset, and subsequently evaluated by testing them using the test dataset. The results of testing the models will enable the comparison between the models, and provide insight into how well the model can make a prediction on new data. The metric that will be used for predictive accuracy is Mean Squared Error (MSE), shown below:

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

The second metric by which the models will be evaluated is their computational complexity. This will be measured in terms of time taken and the amount of memory used to predict once the model has been trained. Python provides a library, *tracemalloc*, which allows a section of code to be evaluated in terms of its memory use. Since these models may be deployed on different systems, for real-time predictions, minimizing the time a prediction takes is important.

4.8 Implementing LSTM and its Derivatives

This paper will use the Keras API running over Tensorflow to implement a LSTM, an stacked-LSTM and bilateral LSTM. Keras has been selected as it is a high-level Python API for Tensorflow that provides scalability and accessible resources and documentation. Scalability is particularly important for the SANReN use-case, as the LSTM models will only be

tested on subsets of SANReN's network traffic data. Therefore, the implemented LSTM models should be able to scale and be trained on larger subsets of NeN data everywhere. By building models and data pipelines using existing libraries, the complexity of model implementation is reduced, and more focus can be placed on hyper-parameter turning and selecting a model that fits within the SANReN constraints.

4.8.1 Baseline LSTM. A baseline LSTM will be developed and implemented using Keras. The number of layers, and other hyper-parameters will be changed in order to produce the highest accuracy.

4.8.2 Stacked-LSTM. A stacked LSTM will be developed and implemented using Keras. The baseline LSTM will be built upon, by adding additional LSTM layers to the model. An LSTM layer above provides a sequence output rather than a single value output to the LSTM layer below, and therefore we will need to adjust the adjust the output of the previous layer to output a 3-D array for input in the next layer [5].

5 Ethical, Professional and Legal Issues

This research uses network data traffic that is not linked to any individual user. The ethical considerations of this project are therefore minimal, and there is no requirement to appear in front of the Ethics Committee. There are no legal issues relating to this project since the data is anonymised. The implementation done will be using TensorFlow, an open-source Machine Learning library created and maintained by Google. Any code developed will be open-source and the final paper is published it will be under the creative commons licence used by the University of Cape Town.

6 Anticipated Outcomes

In this section, the anticipated outcomes are discussed. Major results, including the expected impact and key success factors will also be looked at.

6.1 System

A data processing pipeline will be programmed to automate the preparation of data for the models. The data pipeline will be used to prepare SANReN data for training and testing LSTM models, and will also implement the preprocessing steps described earlier. The pipeline will take SANReN data and clean it, before processing it into engineered data. The engineered data is then ready to be split into training, validating and testing sets for each LSTM model.

The system will also include a programmed preliminary data analysis, as described in the Preliminary Statistical Analysis section. This will be automated in Python, and will output graphs and tables that assist in describing the nature of the data.

By the end of this project, three LSTM models will have been trained, tested and evaluated for the SANReN use-case. Given inputs from the data pipeline, each model will be able to predict network traffic data. The program will be modular, so as to allow a user of the system to predict network traffic for any network, provided that the initial input is correctly formatted.

We will also determine the relative performance of these models based on accuracy and computational efficiency, which will allow us to determine which LSTM model or models will be best suited for SANReN, and NeNs in general.

6.2 Expected Impact of Project

Literature suggests that LSTM is a more accurate predictor of future network traffic when compared to an RNN. We therefore expect our findings to be the same. Our two results, computational complexity and accuracy will allow for a network service provider to decide what prediction approach is more suited to their needs. Therefore, the impact of the product will fall primarily on NeNs. If an LSTM is found to be an accurate, computationally viable predictor for the SANReN use case, then NeNs will be able to implement LSTMs as 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.

6.3 Key Success Factors

In order to determine whether the deep learning approaches are an accurate method for predicting future network traffic, they will need to be evaluated on a test dataset. TensorFlow has built in accuracy metrics, which will allow for the comparison between approaches.

- Successful implementation of an LSTM, Stacked LSTM and BDLSTM to predict network traffic
- Preliminary statistical analysis of dataset to provide general overview of dataset
- Definitive results that allow conclusion on most accurate LSTM approach to network traffic prediction
- Recommendation that can be made based on requirements by network: Highest accuracy, lowest computational complexity

7 Project Plan

7.1 Required Resources

The data for this project will be provided by the South African National Research and Education Network. There is no specific computational requirement for this project, but using a Graphics Processing Unit for training the deep learning models will be beneficial [16]. The code will be run on Google Colaboratory, and will make use of TensorFlow library for the deep learning algorithms.

Table 1. Deliverables of this project and their associated due dates

Due Date	Deliverable
4 June	Literature Review
24 June	Project Proposal
9 July	Proposal Presentation Uploaded
30 July	Revised Project Proposal (after staff feedback)
10-13 August	Initial Software Feasibility Demonstration
6 September	Final Complete Draft of Paper
17 September	Final Submission of Project Paper
20 September	Final Submission of Project Code
4-8 October	Final Project Demonstration
11 October	Poster Due
18 October	Web Page
TBA	Open Evening

7.2 Risks

A risk matrix has been developed to capture the issues that could arise during the project life cycle. These risks have mitigation strategies and ways to monitor them, as well as ways to manage the risks should they be realised. The full risk matrix table can be seen in Appendix B.

7.3 Timeline

The timeline for this project begins with the project proposal, from the 1st of June onward, and culminates with a completed project web page - due on the 18th of October. As shown in the Gantt Chart, preprocessing and baselining will be completed as a team, thereafter each team member will work concurrently on an individual LSTM architecture. Work on the project papers will begin early in the development process, to allow for ample time for feedback, revisions and developments. The full detailed breakdown of the project timeline is in the Gantt Chart in Appendix A.

7.4 Milestones and Deliverables

These milestones are set out by the Computer Science Department of the University of Cape Town.

- Literature Review - A critical analysis of past work in the field, deep learning and time series approaches to network traffic prediction
- Project Proposal - Outlining project plan, deliverables and aims
- Proposal Presentation - Presentation to department motivating importance of project and feasibility
- Initial Software Feasibility Demonstration - Proof of concept, basic workings of project
- Final Project Paper and Project Code - Upload final paper including results and conclusions comparing the LSTM approaches to network traffic prediction, code including implementation of project
- Final Project Demonstration - Demonstration to department that project works, and results observed are consistent with final paper
- Poster - Overview of project and findings
- Web Page - Web page detailing project components and all deliverables

7.5 Work Allocation

This project work will be distributed between Antony and Justin. Together, the two will apply data preprocessing to the raw PCAP data, to convert it into a format that is suitable for training deep learning models. Antony will carry out the preliminary statistical analysis on the data, which has been discussed in Section 4.4, as well as developing the stacked-LSTM implementation. Justin will develop the LSTM which will be used as a benchmark to compare different LSTM approaches, as well as the bidirectional LSTM implementation. The associated model building, training, hyperparameter tuning, testing and evaluation of each LSTM implementation will thus be done individually. While the test and training data will be the same for both Antony and Justin, each one will be individually evaluating the LSTM variants that they develop using the same metrics. Once complete, they will work together, comparing results in order to conduct the comparisons between the architectures in terms of performance and resource utilization.

References

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

[2] 2021. URL https://cloud.google.com/architecture/data-preprocessing-for-ml-with-tf-transform-pt1.

[3] Basil Alothman. Raw network traffic data preprocessing and preparation for automatic analysis. In *2019 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, pages 1–5. IEEE, 2019.

[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] Jason Brownlee. Stacked long short-term memory networks, Aug 2019. URL https://machinelearningmastery.com/stacked-long-short-term-memory-networks/.

[6] Jason Brownlee. How to develop a bidirectional lstm for sequence classification in python with keras, Jan 2021. URL https://machinelearningmastery.com/develop-bidirectional-lstm-sequence-classification-python-keras/.

[7] 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.

[8] 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.

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

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

[11] 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.

[12] 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.

[13] Huang Lin, Wang Diangang, Liu Xiao, Zhuo Yongning, and Zeng Yong. A predictor based on parallel lstm for burst network traffic flow. In *Proceedings of the 2020 6th International Conference on Computing and Artificial Intelligence*, pages 476–480, 2020.

[14] 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.

[15] 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.

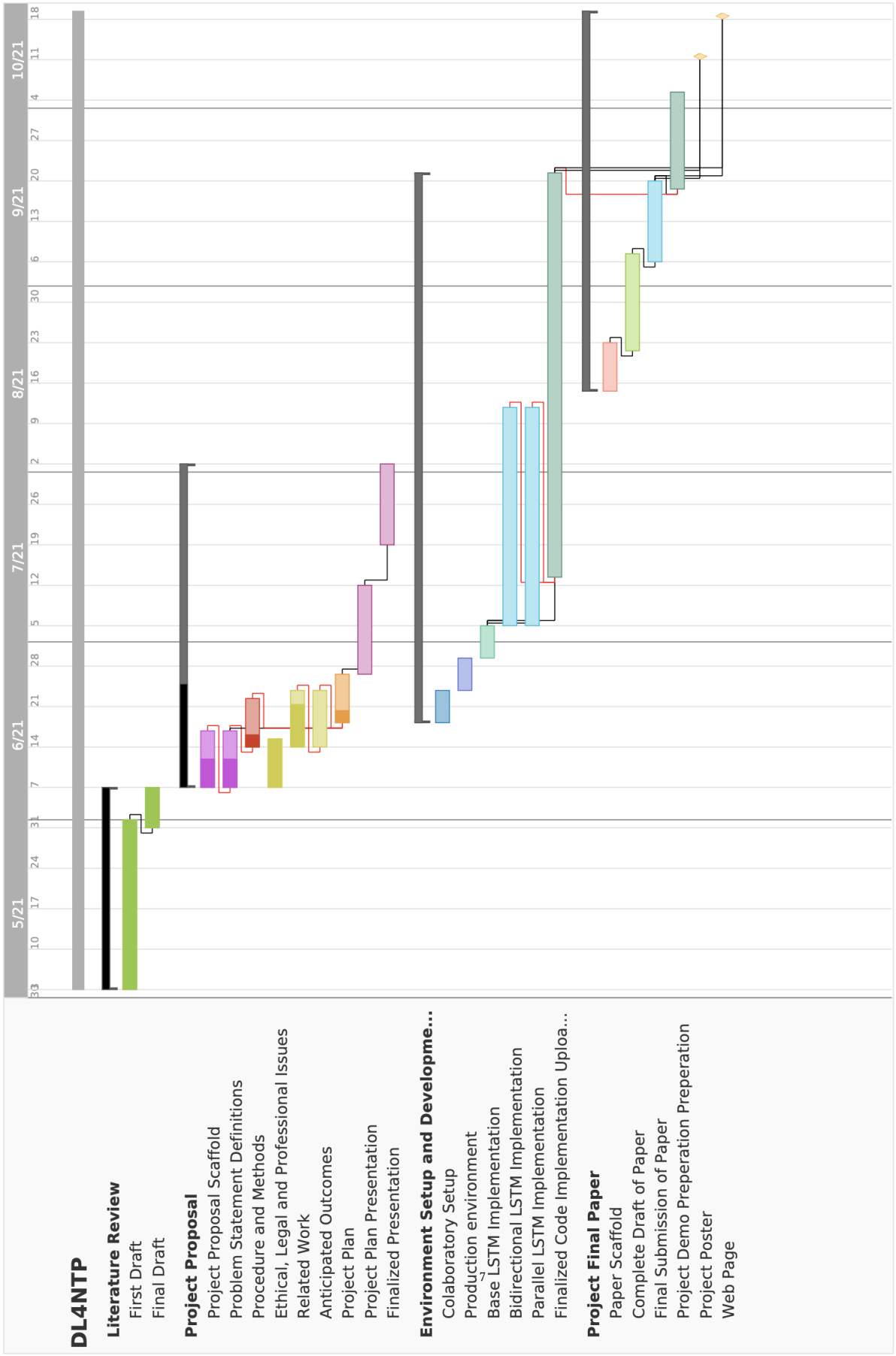
[16] Daniel Schlegel. Deep machine learning on gpu. *University of Heidelberg-Ziti*, 12, 2015.

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

[18] 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.

[19] 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.

Appendix A - Gantt Chart



Risk	Probability	Impact	Consequence	Mitigation	Monitoring	Management
Difficulty collaborating between team members and supervisor, due to online work	High	Medium	Miscommunication slowing down progress of project.	Use Monday as a project management tool, Microsoft Teams for constant communication and ensure that GitHub is constantly updated to reflect changes being made.	Regular meetings with supervisor, constant communication between project members	Emphasize stronger communication and allocate more time each week towards collaborating with team and ensuring mutual understanding of progress.
Insufficient experience in deep learning implementations	Medium	Medium	Delays to project as time needs to be dedicated to researching implementation and structure of LSTMs.	Spend time researching LSTMs and how to implement, use tutorials to practice on other examples and ask lecturers for help or access to resources.	Use tutorials to practice and learn how to implement LSTMs and to understand the theory behind the architectures	Seek additional sources for technical information and allocate more time to the project to cement LSTM. Speak to UCT lecturers in Computer Science department to request a private lesson on LSTMs.
Difficulty cleaning data for use in learning models	Low	High	Unable to begin work on the core part of the project the deep learning classification until data has been cleaned for training deep learning models.	Learn how to use Pandas and Numpy in order to manipulate data frames.	Use the PCAP data in small samples to test data cleaning methods ahead of time.	Consider using smaller data set, in order to reduce cleaning requirements. Potentially change data fields used to mitigate the cleaning requirement.
Insufficient hardware for training deep learning models	Low	High	Inability to train LSTM models in reasonable time, or having to reduce size of training dataset.	Investigate use of Google Colab or UCT HPC.	Test Google Colab or UCT HPC on training or testing deep learning models of varying size.	Consider requesting access to UCT's HPC cluster to improve training time.
Team member dropping out	Low	High	Changing the scope of the project, or having another member have to take on additional work to cover their portion of project.	Constantly communicate any mental or physical health struggles and offer help to each other where possible. Don't allow other team member to get too stressed by leaving deliverables to last minute.	Communicate regularly with team members to monitor each other's progress and well-being.	Work would have to be redistributed amongst remaining team members, and they will have to allocate more time to the project to ensure remaining research objectives are fulfilled.
Not meeting final deadline	Medium	High	Late penalties being applied, reducing the potential mark that can be achieved. Failing the project as a whole.	Ensure all members are aware of project timeline, as well as coursework deadlines and exam schedules.	Follow Gantt chart closely to ensure that milestones and deadlines are met on time, communicate any concerns ahead of time to project supervisor.	Consult with project supervisor to change project scope, or request deadline extension where possible.