A Comparative Evaluation of Deep Learning Approaches to Online Network Traffic Classification for Community Networks

Project Proposal

Matthew Dicks
University of Cape Town
Cape Town, South Africa
dckmat004@myuct.ac.za

Jonathan Tooke
University of Cape Town
Cape Town, South Africa
tkxjon001@myuct.ac.za

Shane Weisz
University of Cape Town
Cape Town, South Africa
wszsha001@myuct.ac.za

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; **Supervised learning by classification**; • **Networks** → **Network management**.

KEYWORDS

Network traffic classification, deep learning, community networks

1 PROJECT DESCRIPTION

Community networks have emerged as a promising solution to providing internet connectivity in rural areas around the world. These networks typically involve a small group of people coming together to develop a network infrastructure in their local community. An internet gateway is then created to connect the local network infrastructure to the wider internet [8]. It is important that these community networks provide a positive user experience to the network users for the full benefits of the network to be experienced.

An approach commonly used to improve user experience on networks is Quality of Service (QoS). QoS engineering works by prioritizing the traffic from some applications over others in accordance with the network's requirements [2]. For example, a network administrator may decide that it is critical that video calls provide a "smooth" user experience. To reduce network jitter, latency, and packet loss for video calls, the network can be configured to forward video call packets before other packets. For the QoS mechanism to be able to prioritize packets of a certain class before others, it **first has to classify incoming packets into their respective application classes.**

Traffic classification used to be a fairly straightforward task, but with the increasing prevalence of encryption and other network security standards, this classification has become challenging, reducing the effectiveness of QoS. Researchers have investigated increasingly advanced methods of performing this classification. Most recently, studies have shown that deep learning algorithms can demonstrate significant success in the online traffic classification task. These algorithms usually work by using patterns within a packet or time series features, both of which would be incomprehensible to a human attempting the same classification [9].

It is the recent success of deep learning classifiers and the need for effective QoS in community networks that has inspired this project. In addressing the traffic classification problem for community networks, some variations of the deep learning classifiers discussed in related literature will be implemented. However, the critical difference to conventional literature is that the choice and evaluation of deep learning architectures will be informed by the computational limits imposed by hardware constraints of community networks. Furthermore, the classifier will be tested on traffic from the Ocean View community network [5], which may have a different distribution to data from networks that traffic classifiers have traditionally been trained on.

2 PROBLEM STATEMENT

Although research into deep learning network traffic classifiers has shown positive results in other studies, such studies have all focused on classifying traffic for specific datasets and cannot necessarily be generalized to traffic for all networks. Furthermore, the limitations of building a computationally inexpensive model, subject to community network resource constraints, has not yet been investigated.

2.1 Research Problems

Two main research problems will be targeted in the formation of this project. The first is to build three different traffic classifiers using different deep learning architectures. These deep learning models will consist of a 1D Convolutional Neural Network (1D-CNN), a 2D Convolutional Neural Network (2D-CNN) and a Long Short-Term Memory (LSTM) model, which will be compared on the basis of their classification accuracy. Furthermore, two benchmark models will be built against which the three deep learning frameworks can be further compared. The second research problem is to build these models such that they can perform the classification within the hardware constraints of community networks.

2.2 Research Questions

To address the two research problems outlined above, the following key research questions are proposed:

(1) Which of the chosen deep learning architectures provides the highest classification accuracy subject to community network resource constraints?
(2) What is the least computationally expensive model that can be built for each deep learning architecture such that it meets a minimum accuracy requirement?
(3) How much impact does the use of an advanced deep learning framework (such as a CNN or LSTM model) have on classification accuracy as compared to the more basic Multi-Layer Perceptron (MLP) deep learning framework, assuming the same computational requirements?

(4) How much impact does the use of an advanced deep learning framework (such as a CNN or LSTM model) have on classification accuracy as compared to the Support Vector Machine (SVM) traditional machine learning framework, assuming the same computational requirements?

3 RELATED WORK

Due to its many important applications, various approaches to traffic classification have been studied extensively in literature. In recent years, much of the research into traffic classification has been around applying deep learning techniques to overcome the difficulties of encryption of traffic. However, many such approaches perform flow-based classification based on inter-packet features, which is less suited to the real-time classification task necessary for QoS. As such, the key works that are particularly relevant to our project are rather those that use deep learning techniques for packet-based classification (that is, classification based solely on each individual packet).

One such study is *deep packet* by Lotfollahi et al. [7] who use 1D-CNNs for packet-based classification. Lotfollahi et al. suggested that 1D-CNNs are ideal architectures for the traffic classification task using packet data, since they are able to recognize dependencies between successive bytes in order to learn key patterns that enable successful classification. The paper explains that their deep learning models are thus able to learn the distinguishable patterns within the encrypted data that characterize applications, despite the content itself being inaccessible due to encryption. Their results testify to this end, with their 1D-CNN model attaining a highly impressive F1 score of 0.95, outperforming all prior similar works in literature that perform classification on the same public dataset. This highlights the success that deep learning can have on traffic classification tasks.

A similar recent study performed by Wang et al [10] used the same public dataset and achieved F1 scores of 0.96 and 0.98 for their MLP and 2D-CNN networks respectively. The study also performed experiments on a smaller, balanced version of the same dataset and attained another impressive set of results, with the MLP reaching an F1 score of 0.93 and the 2D-CNN achieving an F1 score of 0.96. The slight drop in performance can be attributed to the reduced number of data points on which they had to train and test the networks. This is another example of how deep learning networks can learn valuable representations from the raw high dimensional data that comes from individual packets. The paper thus provides further evidence for using deep learning as a tool in packet classification.

Whilst packet-based classification using deep learning has been shown to be successful, to the best of our knowledge there are no studies yet that consider computational resource utilization constraints on deep learning models.

4 PROCEDURES AND METHODS

This section will be used to explain and justify the procedures and methods that will be implemented in order to answer the research questions.

4.1 Obtaining Network Traffic Data

The first step for the project involves accessing the community network traffic data that will be used for training and testing our deep learning models. To this end, we will be using a dataset that was collected from the Ocean View community network in Cape Town [5]. The data consists of numerous PCAP files that were collected at the gateway of the network, capturing all traffic flowing between the network and the Internet from February 2019 onwards. The PCAP files have been copied to a data repository at the University of Cape Town, through which we will accessing the data.

4.2 Preprocessing

The preprocessing stage will take as input a set of PCAP files, and will label each packet and extract the raw data that will be used as features. The stage will then transform those features so that they can be used as input to the neural networks. This section will describe the methods that will enable these tasks.

4.2.1 Packet labeling. The goal of this project is to use deep learning to perform traffic classification. Classification is a supervised learning task. This means that each example must have a label associated with it, which will allow the neural networks to learn from their errors and adjust their parameters accordingly. The packets in PCAP files do not have labels associated with them already, so we will have to provide these labels. This will be done by splitting each PCAP file into its respective flows using a tool called *pkt2flow*[1]. An open-source deep packet inspection library called *nDPI* will then be used to label each flow. Each packet associated with a given flow will receive that flow's label.

4.2.2 Feature extraction and transformation. The features that will be used as input into the deep learning networks will be the bytes extracted from the IP payload for each packet. This method was chosen because it allows us to use both encrypted and unencrypted packets. There is also evidence to suggest that using this data as input into deep learning models in the context of traffic classification can yield high prediction accuracy [7, 10].

Python has an open source library called *scapy* that will enable us to process packets and flows found in PCAP files. The library has methods that can extract the IP payload from packets and convert it into bytes. These methods will be used to obtain the features needed for each packet.

Once the raw data has been extracted from the packets, it has to be transformed into an appropriate format so that a given model can learn from it. For the 1D-CNN and the LSTM models, the one-dimensional byte stream is an appropriate feature vector — the only transformation that is necessary is to pad the vectors with zeros to ensure that they are all the same length. The 2D-CNN network accepts a two-dimensional image as input, therefore the byte stream will have to be transformed into an image. This can be done by creating an NxN matrix from the bytes in a stream. Once the data has been transformed into the correct format, it will be normalized to increase the learning algorithm's stability and

[1] *pkt2flow* is a simple utility that classifies packets into flows. It takes a single PCAP file as input and returns a set of PCAP files where each file contains a single flow. It is available at: https://github.com/caesar0301/pkt2flow.

decrease the training time, which could save valuable resources for community networks.

4.2.3 Balanced dataset. Machine learning classification models perform better when the number of examples per class are the same [10]. If the classes are not balanced, then the model can learn the unequal distribution between the classes and skew its predictions in favor of the majority class – causing poor predictive accuracy in the minor classes. To mitigate this, steps will be taken to ensure that the dataset is balanced. Due to the large number of packets collected from the Ocean View community network, there will be enough examples even in the most underrepresented classes. To create a balanced dataset we will thus randomly remove examples from the majority classes until all the classes have the same size.

4.3 Establishment of Benchmarks

To determine the minimum level of performance that would justify using deep neural networks, we have chosen to implement a traditional machine learning model to be used as a benchmark. The model that will be used as the benchmark will be the traditionally highly successful Support Vector Machine (SVM). Besides the SVM's predictive capabilities, this model was chosen because it will be able to learn and make predictions based on the same input data format as the deep learning models. Other machine learning methods such as Random Forests and Decision Trees would not be considered viable options with such high dimensional data. Deep learning networks have a large number of parameters and are computationally expensive to train and deploy – so, if our neural networks do not significantly outperform the SVM, then the recommendation will be against using deep learning models in community networks.

Since we will be implementing sophisticated neural network architectures in the form of CNNs and an LSTM, we have decided to compare these sophisticated models with the simpler MLP. The CNNs and LSTM networks will have to have significantly greater predictive accuracy to justify their additional features. The benchmark is also set up to help answer our third research question regarding the impact of advanced deep learning frameworks compared to the more basic MLP.

4.4 Learning the Sequential Patterns in the Data

The byte stream obtained from each packet is inherently sequential, because the order of bytes matters. Long Short-Term Memory (LSTM) networks and one dimensional Convolutional Neural Networks (1D-CNN) are built to learn from sequential data and have consequently been chosen to detect the sequential patterns in these byte streams.

The 1D-CNN has been chosen because it has two important characteristics – namely sparse interactions and parameter sharing – that enable it to be more computationally efficient. Sparse interactions limit the amount of dependencies between neurons and therefore reduces the training time, and parameter sharing means that parameters can be reused which will decrease the memory requirements needed by the network. This neural network architecture also showed great predictive performance in a previous study [7].

LSTM networks are a type of recurrent neural network that can learn long term dependencies between inputs. Due to the success of 1D-CNNs in previous studies, it is expected that a more powerful sequence model would increase the accuracy. However, the increase in accuracy of such a model does come at a price – because of the inherently sequential structure of the LSTM model, it cannot be trained or run in parallel, thereby increasing the training time and its requirement for computational resources. Therefore, the 1D-CNN may still prove to be the better model for community network purposes, even if its accuracy is lower than the LSTM. As far as we are aware, the LSTM network and the 1D-CNN have not been directly compared in packet-based classification, and therefore this project will yield novel results in this regard.

4.5 Learning the Spatial Patterns in the Data

There is also evidence to suggest that 2D-CNNs can be effective for packet-based classification due to their ability to learn spatial patterns in the data. Both the sparsity of information to be found in the packet data, and the noisiness of the data in terms of variability amongst packets, provide justification for the suitability of 2D-CNNs for the packet classification task. This is supported by a study done by Wang et al [10], who built a system called *Datanet* that used a 2D-CNN to classify individual packets. They achieved an F1 score of 0.96 and 0.98 on a balanced and an unbalanced dataset respectively. Furthermore, the 2D-CNN has also achieved high accuracy and F1 scores compared to other machine learning models in classifying flows [1, 6, 11], which suggests that the network architecture is able to learn meaningful patterns in sequential data. The 2D-CNN also takes advantage of sparse interactions and parameter sharing, which will make the network's forward propagation pass and the training time more efficient. This is important in the context of a community network. Whilst the LSTM and 1D-CNN neural networks are designed to detect sequential patterns in data, the 2D-CNN architecture is a specialized kind of neural network for learning spatial dependencies in data that has a known grid-like topology, such as an image [4]. As a result, comparing the 2D-CNN, the 1D-CNN and the LSTM networks will also allow us to determine whether the patterns within packet data are purely sequential, or if there also exist useful spatial patterns that can be uncovered through a grid-like topology in the data.

4.6 Implementation of the Machine Learning Models and Deep Learning Networks

The deep learning networks will be built in Python using Tensorflow's implementation of the Keras API. Keras was chosen because it provides easy ways of building networks without sacrificing flexibility. It also supports eager execution and input pipelines built in Tensorflow, which will decrease the memory requirements when training the models. The inputs will need to be padded in order to be used as input for the models. Hence, another reason for using Tensorflow is that it supports automated padding. Removing the complexity of building networks and data pipelines by using these libraries will allow us to focus on hyperparameter tuning and experimentation with different network structures.

The Support Vector Machine (SVM) will be built with Python's scikit-learn library. The decision to use this library was on the basis

that scikit-learn includes the ability to make multi-class classifications with an SVM model. This is important because we will be classifying traffic into multiple application classes.

4.7 Hyperparameter Tuning

Hyperparameters are values that are set before the training of the networks and therefore are not learned during the training process. Some examples of hyperparameters in neural networks include the network topology, the learning rate of the optimization algorithm, as well as the training batch size and the number of training epochs. Since these parameters have to be defined prior to training, the best way to find the parameters will be to systematically try out different combinations and then pick the best set. To do this, we will be implementing randomized grid search together with k-fold cross-validation. In this regard, for each neural network architecture, we will enumerate the set of hyperparameters that need to be tuned, including for example the filter size for the CNNs and the number of LSTM cells for the LSTM model. For each hyperparameter in the set, we will then list a suitable representative set of values that we wish to explore, for example for the learning rate using values of 0.1, 0.05, 0.01 and 0.001. These possible values will then be used to randomly choose combinations of the hyperparameters, each of which will be used for retraining the model and evaluating its performance using k-fold cross-validation. The combination that produces the lowest cross-validation error will be an approximation to the optimal set of hyperparameters and hence will be used as the hyperparameters for the final model.

4.8 Evaluating the Models

To evaluate the performance of the models, we will be creating training and test datasets. Networks will be trained and evaluated on the training set using k-fold cross-validation. After this process, the best models will be chosen and will then be evaluated on the test set. The scores that are achieved on the test set will give us an indication of each network's ability to generalize to new examples. The scores obtained on the test set will thus be used to compare each model's predictive power.

The models will be evaluated based on two characteristics, namely predictive performance and computational efficiency. The metric we will use for predictive performance is accuracy, and the formula is as follows:

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions} \times 100$$

The computational efficiency of the algorithms will be measured in two ways – the time it takes to make a prediction, and the amount of memory needed to make a prediction. Since the algorithms are going to be deployed to produce real-time classifications, the speed of the predictions is key. Therefore, we will be using the average time it takes a model to make a prediction as a metric to analyze computational efficiency. To estimate the amount of memory needed to make a prediction, we will use a python library called *tracemalloc* that can calculate the amount of memory used in a section of code. A prediction function will be implemented and *tracemalloc*'s methods will be invoked to measure the memory used by this function. The average amount of memory needed by a given architecture to make a prediction will then be used as the second metric to measure computational efficiency.

5 ANTICIPATED OUTCOMES

This section is going to describe what we expect to build and accomplish by the end of the project. It lays out the key aspects of the system, the predicted impact that our project will have, and how we expect to evaluate the success of our project.

5.1 System

The following subsections will describe the key software systems that will have been built by the end of this project.

5.1.1 Data pipeline. The data pipeline will be used to implement the preprocessing steps described in the preprocessing section (Section 4.2). The pipeline will be the part of a system that takes in raw traffic data and transforms it into an appropriate structure on which networks can be trained and tested. It will be able to take in a set of PCAP files, and for each packet, it will label it and extract the data from it to provide the feature label pairs. It will then take these feature label pairs, place them into batches, and pad them so that efficient training can be accomplished.

5.1.2 Deep Learning Models. At the end of this project, we will have trained and tested four deep learning models and one machine learning model. It is expected that these models will be able to classify traffic based on raw byte streams obtained from a given packet's IP payload. These models will be coupled with the data pipeline previously described, and using these two systems together, a user will be able to train and classify both encrypted and unencrypted traffic. We will also determine the relative performance of these models based on accuracy and computational efficiency, which will allow us to conclude which model or models will be best suited for deployment in a low resource environment.

5.2 Impact of Project

Community networks host a wide range of application services such as VoIP, content distribution, on-demand and live streaming media, instant messaging, as well as back-ups and updates. The challenge is that they have to provide these services on links and servers with limited capacity. They also have to manage the diverse and volatile resources in the network [3]. Therefore to be able to provide successful QoS to the users, network administrators should be able to know how their resources are being used – which our project will help with. The traffic classification will provide a way for the citizens and non-profit organizations who are running the network to view which applications are using which resources, and then network managers can analyse these patterns to ensure that resources are being used in the most efficient way. Our systems will also be able to perform near real-time classification that can be used to adjust traffic engineering schemes, by giving certain traffic types higher priority and deprioritizing others. This will enable the networks' QoS algorithms to provide a better experience for the users in the network.

5.3 Key Success Factors

To classify our project as a success, the following points will have to be satisfied. Firstly, the data pipeline will have to be implemented such that it can perform all of our preprocessing. The data pipeline will have to be integrated with each of our models so that they can be trained and tested on the preprocessed data. Furthermore, we will have to correctly implement our machine learning model and our deep learning networks so that they can yield valuable results. We will have to perform rigorous experiments so that we can definitively answer all of our research questions. Lastly, we will need to determine whether deep learning is a suitable approach for real time packet classification in a resource constrained environment.

6 ETHICAL, PROFESSIONAL AND LEGAL ISSUES

The project faces few ethical, professional, and legal issues. This is largely due to the fact that the team is not interacting with individual people, or sensitive data that can be traced to individuals. The most important ethical issue to consider is the confidentiality of the aggregated network data that has been provided by the Ocean View community network in South Africa for the purposes of training and testing our models. The network has granted permission for the team to use the data for research purposes. However, it is still important that the privacy of the network is protected, so the team will ensure that the data is only accessed by the team and not distributed publicly. There are no legal issues faced by the project.

Upon completion of the project, the source code will be made public for future researchers to use and build upon. If the research paper is published, it will be published under the creative commons license as per the University of Cape Town's policy.

7 PROJECT PLAN

In this section we provide a detailed account of our proposed project plan, including key required resources, anticipated risks and associated management strategies, a project timeline along with key deliverables and milestones, as well as a specification of how the work will be distributed amongst the team members.

7.1 Required Resources

The below categories contain our identification of the key data, software and hardware resources necessary for the success of the project.

7.1.1 Data. The primary data resource required for our project is the iNethi [5] data logs, containing numerous PCAP files with traffic traces collected at the gateway of the OceanView community network. The packets in these traces, after being labelled, will form the training and testing sets essential to the development of successful deep learning classifiers.

7.1.2 Software. In terms of software, we will make use of the AnyConnect VPN for off-campus login to UCT's network in order to access the iNethi data logs. The tools and libraries we plan to use for preprocessing include *pkt2flow* for splitting packets into flows, *nDPI* for labelling, and the Python library *scapy* for processing packet data. We plan on using Python 3 as our primary programming language for development of our deep learning models, and as such key libraries we expect to use are *numpy*, *pandas*, *matplotlib*, *scikit-learn* and *tensorflow*.

7.1.3 Hardware. We will be making use of our personal computers for the majority of the development process. However, when training our deep learning models, we plan on utilizing a GPU to speed up the training process, either through Google Colab or AWS. Additionally, we may consider requesting access to UCT's HPC cluster for model training in order to further reduce training time.

7.2 Risks

We have identified a number of key risks that could be responsible for significant project disruption if not appropriately anticipated and managed. These have been detailed in the risk matrix in Appendix A, where each risk is outlined along with our estimation of its likelihood, its impact on the project, the risk consequences, and associated mitigation, monitoring and management strategies.

7.3 Timeline

The timeline for our project begins with work on the project proposal from the 4th of May and culminates with a completed web page showcasing our work on the 19th of October. The full detailed breakdown of the project timeline can be seen in the Gantt Chart in Appendix B. As shown in the Gantt chart, preprocessing and benchmarks will be completed as a team, whereafter each team member will work concurrently on an individual deep learning architecture approach to the problem. Work on the project papers will begin early in the development process, to allow for ample time for change, feedback, revisions and developments.

7.4 Milestones

The table below outlines the key project milestones as specified by the Computer Science Department.

Due Date	Deliverable
12 May	Literature Review
4 June	Project Proposal
29 June	Revised Project Proposal (after staff feedback)
3-11 August	Initial Software Feasibility Demonstration
11 September	Final Complete Draft of Paper
21 September	Final Submission of Project Paper
25 September	Final Submission of Project Code
5-9 October	Final Project Demonstration
12 October	Poster Due
19 October	Web Page
TBA	Open Evening

Of these, milestones of particular note regarding the culmination of the project are the submissions of our final papers on 21 September, the final submission of project code on 25 September, the project demonstration in the week starting 5 October, and the project poster and web page completed by 12 October and 19 October respectively. The dependencies between the tasks associated with the milestones, and additional intermediary milestones, can be seen on the timeline in the Gantt Chart in Appendix B.

7.5 Deliverables

The following is a list of the deliverables that will be produced during the course of the project (with the literature reviews already complete):

- Three literature reviews on deep learning approaches to network traffic classification
- Project proposal
- A demonstration of software feasibility, including preprocessing approach and implementation of deep learning models
- Three complete final papers evaluating each deep learning technique's success relative to our research objectives
- Completed software system with modules for preprocessing the data, training each deep learning architecture, and running the models on test data
- Project poster outlining key findings pertaining to our research objectives
- Project web page with key components and deliverables from the project

7.6 Work Allocation

As mentioned in reference to the project timeline, the project work will be distributed amongst the team members. The team will together build data processing tools and conduct the data preprocessing, in terms of dividing the raw PCAP files into flows and labelling them accordingly. Additionally, the benchmark models — to which our individual deep learning models will be compared — will be developed together as a team. Thereafter, each team member will begin individual work on applying their allocated deep learning architecture to the classification task. Namely, Matthew will be developing the LSTM RNN, Jonathan will be working on the 1D-CNN and Shane will take on the 2D-CNN. The associated model building, training, hyperparameter-tuning, testing and evaluation of each implementation will thus be done individually. Thereafter, the respective comparisons between the architectures in terms of performance and resource utilization will be carried out together.

REFERENCES

[1] Giuseppe Aceto, Domenico Ciuonzo, Antonio Montieri, and Antonio Pescapé. 2019. Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges. *IEEE Transactions on Network and Service Management* 16, 2 (2019), 445–458.

[2] A. O. Adedayo and B. Twala. 2017. QoS functionality in software defined network. In *2017 International Conference on Information and Communication Technology Convergence (ICTC)*. 693–699.

[3] Bart Braem, Chris Blondia, Christoph Barz, Henning Rogge, Felix Freitag, Leandro Navarro, Joseph Bonicioli, Stavros Papathanasiou, Pau Escrich, Roger Baig Viñas, et al. 2013. A case for research with and on community networks.

[4] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[5] iNethi Technologies. 2020. https://www.inethi.org.za/deployments/ Accessed: 2020-04-29.

[6] Manuel Lopez-Martin, Belen Carro, Antonio Sanchez-Esguevillas, and Jaime Lloret. 2017. Network traffic classifier with convolutional and recurrent neural networks for Internet of Things. *IEEE Access* 5 (2017), 18042–18050.

[7] Mohammad Lotfollahi, Mahdi Jafari Siavoshani, Ramin Shirali Hossein Zade, and Mohammdsadegh Saberian. 2020. Deep packet: A novel approach for encrypted traffic classification using deep learning. *Soft Computing* 24, 3 (2020), 1999–2012.

[8] P. Micholia, M. Karaliopoulos, I. Koutsopoulos, L. Navarro, R. Baig Vias, D. Boucas, M. Michalis, and P. Antoniadis. 2018. Community Networks and Sustainability: A Survey of Perceptions, Practices, and Proposed Solutions. *IEEE Communications Surveys Tutorials* 20, 4 (2018), 3581–3606.

[9] S. Rezaei and X. Liu. 2019. Deep Learning for Encrypted Traffic Classification: An Overview. *IEEE Communications Magazine* 57, 5 (2019), 76–81.

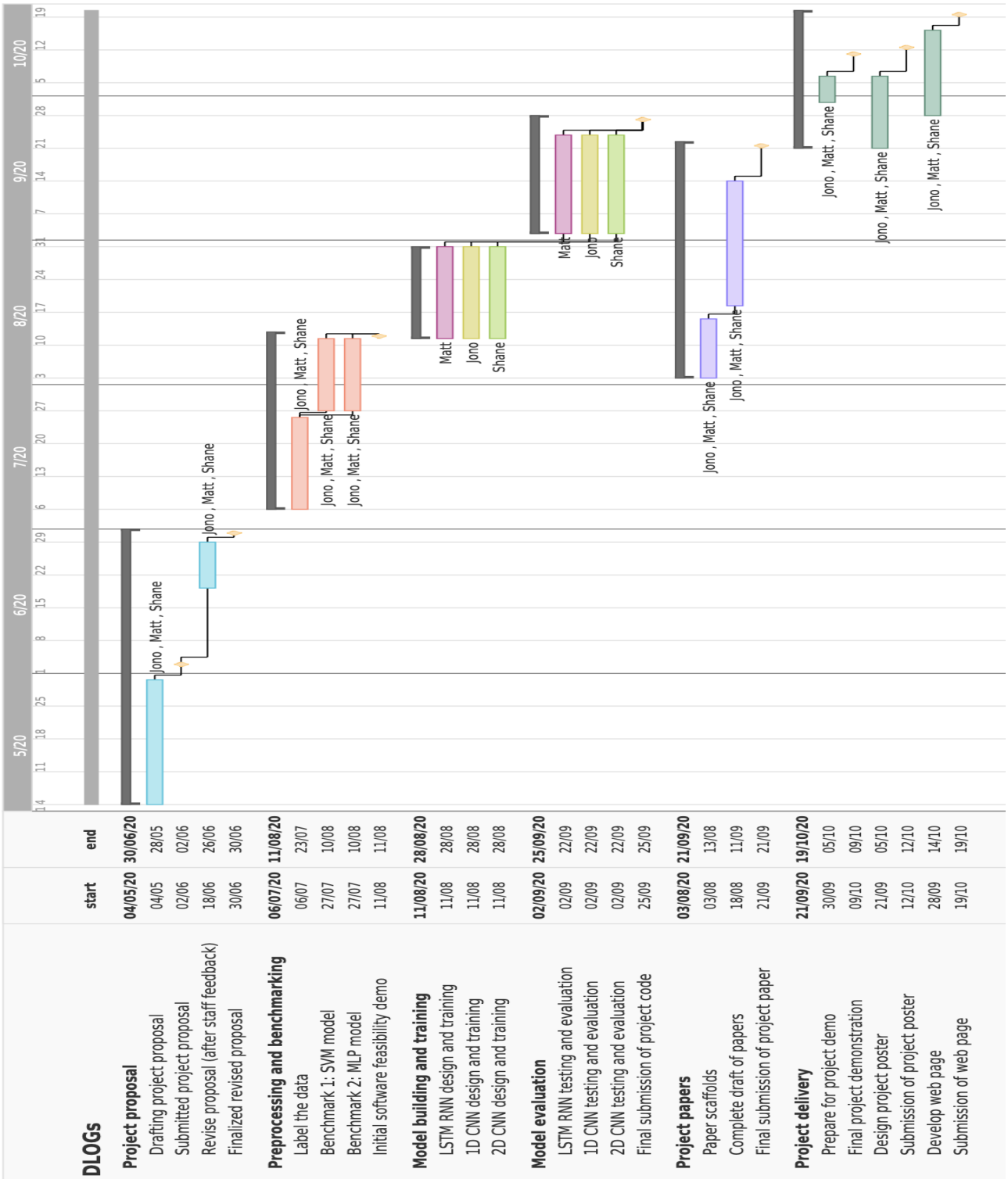
[10] Pan Wang, Feng Ye, Xuejiao Chen, and Yi Qian. 2018. Datanet: Deep learning based encrypted network traffic classification in sdn home gateway. *IEEE Access* 6 (2018), 55380–55391.

[11] Wei Wang, Yiqiang Sheng, Jinlin Wang, Xuewen Zeng, Xiaozhou Ye, Yongzhong Huang, and Ming Zhu. 2017. HAST-IDS: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection. *IEEE Access* 6 (2017), 1792–1806.

APPENDIX A: RISK MATRIX

Risk	Probability	Impact	Consequence	Mitigation	Monitoring	Management
Higher difficulty collaborating between team members, and with supervisor, due to working remotely as a result of coronavirus restrictions	High	Medium	Slower project progress due to communication delays. Potential for miscommunication resulting in further delays to resolve issues.	Use online conferencing tools such as Jitsi, project management tools such as Trello and communication platforms like Slack in order to optimize our remote workflow.	Have regular meetings with supervisor and team members to ensure all parties are on the same page and understands where each other is at.	Emphasize stronger communication and allocate more time each week towards collaborating with team and ensuring mutual understanding of progress.
Insufficient practical background and experience in applied deep learning	Low	High	Project delays due to extra time that must be spent understanding how to apply the architectures at the time of implementation.	Spend time before the project term on online coursework and regular reading and practice in the area to ensure sufficient preparation for when implementation starts.	Practice applying deep learning through assignments and applications in typical problem domains to check understanding of techniques.	Seek additional sources for technical information and allocate more time to the project to cement deep learning understanding and be able to implement accordingly.
Difficulties with labelling the dataset efficiently, and associated project delays	Low	Medium	Unable to begin work on the core part of the project - the deep learning classification - until labelling of training data complete.	Spend a lot of time before the project term researching and deciding upon the best candidate tools and pipeline scripting approaches for preprocessing.	Test labelling and preprocessing approach on a few sample PCAP files to identify potential difficulties.	Consider restricting flows to, for example, HTTP flows in order to make the labelling task easier. Consider using a public labelled dataset.
Insufficient hardware for training deep learning models	Medium	Medium	Significant disruption to project due to difficulties in training models in reasonable amounts of time.	Find and decide upon suitable GPU architectures to use for training, such as through AWS or Google Colab.	Test GPU architectures beforehand to make sure they are suitable for training come implementation time.	Consider requesting access to UCT's HPC cluster to improve training time.
Team member dropping out	Low	Medium	The component of the research objectives this team member was responsible for may need to be dropped, and the parts done as a team will require more work from the remaining members.	Have regular check-ins amongst team members, and offer significant emotional support for one another.	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	Low	High	Inability to complete project report in time, resulting in failing course objectives.	Communicate regularly between team members to ensure each person's work is within scope and on schedule.	Utilize regular progress checks between team members and supervisor, in line with the timeline specified in our Gantt chart.	Consider redefining project scope, after consultation with supervisor, to allow for timely project completion

APPENDIX B: PROJECT TIMELINE



DLOGs	start	end
Project proposal	**04/05/20**	**30/06/20**
Drafting project proposal	04/05	28/05
Submitted project proposal	02/06	02/06
Revise proposal (after staff feedback)	18/06	26/06
Finalized revised proposal	30/06	30/06
Preprocessing and benchmarking	**06/07/20**	**11/08/20**
Label the data	06/07	23/07
Benchmark 1: SVM model	27/07	10/08
Benchmark 2: MLP model	27/07	10/08
Initial software feasibility demo	11/08	11/08
Model building and training	**11/08/20**	**28/08/20**
LSTM RNN design and training	11/08	28/08
1D CNN design and training	11/08	28/08
2D CNN design and training	11/08	28/08
Model evaluation	**02/09/20**	**25/09/20**
LSTM RNN testing and evaluation	02/09	22/09
1D CNN testing and evaluation	02/09	22/09
2D CNN testing and evaluation	02/09	22/09
Final submission of project code	25/09	25/09
Project papers	**03/08/20**	**21/09/20**
Paper scaffolds	03/08	13/08
Complete draft of papers	18/08	11/09
Final submission of project paper	21/09	21/09
Project delivery	**21/09/20**	**19/10/20**
Prepare for project demo	30/09	05/10
Final project demonstration	09/10	09/10
Design project poster	21/09	05/10
Submission of project poster	12/10	12/10
Develop web page	28/09	14/10
Submission of web page	19/10	19/10