Project Proposal: Traffic Classification using Hybrid Deep Learning for Community Networks

Boitumelo Mokoka
University of Cape Town
South Africa
MKKBOI005@myuct.ac.za

Ryan Murphy
University of Cape Town
South Africa
MRPRYA002@myuct.ac.za

Michael Gamsu
University of Cape Town
South Africa
GMSMIC004@myuct.ac.za

1 INTRODUCTION

Community networks are a global solution to extend internet access to rural and low-resource areas. These community networks are locally developed and managed to provide network infrastructure and access points to the internet without the need for internet service providers (ISPs) [19]. In remote and underdeveloped regions, community networks offer an opportunity to provide affordable connectivity. However, administration, scalability, and service quality challenges present significant obstacles in deploying and sustaining these low-resource networks [17]. Traffic classification can enhance network management capabilities, notably when applied to quality of service (QoS) management, resource allocation, intrusion and malware detection [1]. Thus, the network can prioritise particular applications depending on the network's requirements using traffic classification by administrators. Traffic classification enables packets coming over the network to be identified by their type, allowing network administrators to make informed decisions to improve QoS in community networks.

Traditional forms of traffic classification included port-based classification, which involved examining the ports of the incoming and outgoing packets, which are now obsolete due to port obfuscation [30]. Furthermore, packet encryption has made simple packet-based classification methods ineffective [16]. Traditional traffic classification techniques are not optimised for accurate and efficient network traffic classification in large-scale datasets, which presents challenges for expanding network traffic and user numbers [28]. Therefore, deep learning advancements have shown to be the most effective way to classify network traffic [28]. Furthermore, deep learning approaches have been effective in network traffic classification due to their improved ability to handle encrypted and complex data [8, 15]. Deep learning is a subset of machine learning, and the focus is specifically on training deep neural networks with multiple layers to learn complex patterns and representations directly from raw data, surpassing traditional techniques in accuracy and efficiency [10].

Specifically, we will be looking at hybrid deep-learning models. These models consist of more than one model, with the output of one model, being used as the input of another in a sequential manner. These are useful as the specialities of more than one model can be leveraged to gain better results[16]. We will be using hybrid models with a range of combinations of individual models to classify network traffic.

2 PROBLEM STATEMENT

This research applies deep learning models to classify network traffic accurately and efficiently on low-resource community networks. One of the pitfalls of the deep learning approach is that it is generally computationally intensive, presenting challenges in resource-constrained environments [26]. Our research problem aims to tackle the application of deep learning in a resource-constrained context of a low-resource community network. Due to community networks being locally developed and managed, they operate with limited computational resources such as CPU and memory, financial support and technical expertise [19]. This study addresses these challenges by investigating lightweight deep learning architectures and optimisation techniques tailored to the resource-constrained community network. By identifying and evaluating the most effective strategies for traffic classification in this context, our research aims to empower network administrators with the tools and insights needed to enhance QoS and resource allocation.

(1) How can deep learning architectures, specifically our hybrid architectures, be adapted and optimised to accurately classify network traffic within a community network's resource-constrained environment?
(2) Considering computational power and memory limitations, what are the most effective strategies and techniques for enhancing hybrid deep learning model performance in community networks?

3 DEEP LEARNING MODELS FOR TRAFFIC CLASSIFICATION

Deep learning has emerged as a suitable approach for achieving efficiency and accuracy in network traffic classification [16]. Within this context, four prominent deep-learning architectures are comprehensively explored, their mechanism understood, and subsequently integrated into various hybrid architectures for resource-constraint traffic classification.

3.1 Multilayer Perceptrons (MLP)

MLPs are a type of feedforward model. They consist of layers of neurons where input is multiplied by multiple weights, and biases are added. After that, at every neuron, the result gets inputted to some activation function, which transforms the output. The outputs are then used as input to the next layer of neurons. This process repeats until a value is outputted at the end. MLPs adapt their weights and biases using a backpropagation algorithm, which creates the illusion of "learning". MLPs can be used for both prediction and classification problems[23]. These models serve as the foundation for more complex models.

3.2 Convolutional Neural Networks (CNN)

CNNs use the same principles MLPs use, such as the feedforward phase and backpropagation. The mathematical complexity of the CNN lies in its multiple hidden layers where convolutional and pooling layers perform feature extraction. The hidden layers converge at a fully connected layer just before the output layer [6]. During convolution, a sliding kernel window performs calculations on several bits at a time, and the following pooling layer acts on this kernel using either the maximum or average value. The fully connected layer uses an activation function (typically ReLU) to perform classification based on the output of the previous layers. CNNs are widely used to detect data spatial features. The difference between 1D and 2D CNNs is the kernel dimensions. For 2D CNNs, the kernel is two-dimensional, and for 1D CNNs, the kernel is one-dimensional[6]. Generally, 1-dimensional CNNs are better at learning temporal patterns, and 2-dimensional CNNs are better at learning spatial patterns [18]. We aim to utilise the CNN's spatial-feature extractions to identify repeating communication patterns in traffic data [8].

3.3 Recurrent Neural Networks (RNN)

RNNs simulate memory using memory cells, feedback loops and gates. These mechanisms make RNNs effective in identifying temporal patterns. Whereby previous inputs affect the computation of the current input [8]. Specialised RNN versions such as Long Short Term Memory Networks (LSTM) and Gated Recurrent Units (GRU) have been introduced to address the problematic vanishing gradient problem [11]. LSTMs use internal cell states to selectively remember and forget past information. The difference between LSTMs and GRUs lies in their internal complexity. Our goal will be to experiment with their incorporations into different hybrid architectures to find the balance between efficiency, accuracy, memory and computations when using different RNNs [5].

3.4 Stacked Autoencoders (SAE)

SAEs are an unsupervised learning technique that utilises feedforward and backpropagation techniques similar to MLPs during optimisation. They consist of encoders which employ dimensionality reduction and decoders that attempt to reconstruct the compressed input back to its original form. This stacked architecture allows the network to learn increasingly complex data hierarchically [8]. The reconstruction error is used to evaluate the model's performance by measuring the difference between the decoder's original and output [14].

4 RELATED WORK

From the literature, we found that flow-based and packet-based inspection are the two most commonly used approaches. Flow-based involves aggregating packets into flows based on their unique identifiers such as source IP, packet IP and source and destination port number[2]. Inter-arrival times can be extracted from these flows, which can be a beneficial input to a model. On the other hand, packet-based classification uses the actual contents of the packet itself. Several features, such as protocol, payload content, headers, and many more, can be extracted, giving the model plenty of features to make predictions.

Lotfollahi et al. used a packet-based inspection method and achieved an average F1 score of 92% using an SAE, while they achieved an average F1 score of 94% using a 1D-CNN [16]. However, their study focused on classifying traffic and protocols used, whereas our study focuses on evaluating the performance of deep learning on resource-constrained community networks. In their research, Bayat et al. utilized various techniques to classify mixed packet-based and flow-based data. Their approach involved a hybrid model that combined a CNN and a GRU. The data was first passed through the CNN, and the resulting output was then used as input for the GRU. The model achieved an accuracy level of 95% [3]. However, it is worth noting that Bayat et al. focused exclusively on HTTPS services, and did not explore application classification in resource-constrained environments.

The research most similar to ours is Tooke, Weisz and Dicks [7, 27, 29]. We will follow the same packet-based approach and pre-processing pipeline with a few modifications. Tooke, Weisz, and Dicks used a 1D-CNN, a 2D-CNN, and an LSTM, respectively. However, they compared their performance and accuracy against a simpler machine learning model called Support Vector Machine (SVM). The SVM performed better in packets classified per second but worse in accuracy. This is due to the higher parameter counts of the other models. Therefore, we focus on hybrid models that can achieve high precision and performance while limiting parameters.

5 MODEL ARCHITECTURE DESIGN

We aim to build on the work of Tooke, Weisz, and Dicks by refining traffic classification deep learning models to be more lightweight and computationally efficient. While maintaining similar accuracy, we strive to reduce the computational expense, making these models more practical for deployment in resource-constrained community networks.

We will implement three sets of hybrid architecture approaches to achieve this goal. The first will be a set of variations of SAE-RNN hybrid architectures, more specifically, the SAE-LTSM and SAE-GRU implementations. Here, the encoding stages of the Auto-Encoder are used for feature reduction, improving the efficiency of the classification step, which can be achieved using an LSTM or GRU network [14]. These RNNs are designed to learn temporal patterns. Additionally, the LSTM can be implemented in the encoding stage of the network to extract time-series patterns as features to assist classification [11].

The second set of hybrid architectures will be variations of SAE-CNN hybrids, specifically SAE with a 1D and 2D CNN.. The SAE will encode the data before reducing input features into the CNN, decreasing model complexity and computational operations [14]. Using different dimensions of CNNs enables us to learn temporal and spatial patterns [18]. Furthermore, Tooke, Weisz and Dicks' CNN was their best performing in terms of accuracy but not performance [7, 27, 29]. Thus, we will investigate whether reducing the input features ensures an accurate but better-performing model.

The final set will include variations of CNN-RNN hybrid models, inspired by Bayat et al.'s success in utilising this approach for HTTPS services [3]. We aim to adapt it for broader application classification while reducing computational complexity. By integrating

a 1-D CNN to extract local patterns, such as packet header information, the CNN-RNN hybrid model aims to enhance computational performance. This hybrid approach builds upon the findings of Tooke, Weisz, and Dicks, who demonstrated the effectiveness of CNN models in traffic classification [7, 27, 29]. This adaptation will use a 1-D CNN instead of a 2-D CNN, aiming to reduce computational complexity while maintaining accuracy. By leveraging the temporal pattern learning capabilities of RNNs, particularly LSTM and GRU networks, the hybrid model seeks to optimise performance while managing the memory-intensive nature of RNNs.

6 PROCEDURES AND METHODS

6.1 Obtaining Network Traffic Data

Community Network traffic data will be used to train and test our deep-learning models. Our research will use datasets collected from the Ocean View community network in Cape Town. The raw data is a collection of PCAP files collected at the gateway of the Ocean View community network. The captured data consists of traffic flowing between the Internet and the Network from February 2019 until May 2019. The PCAP files are stored in a data repository at the University of Cape Town, through which we have been granted access to the data.

6.2 Preprocessing

The preprocessing stage will generate our training data from a set of PCAP files as input. This stage consists of packing labelling extracted from the raw data to be used as features for our model.

6.2.1 Packet labelling. Since we are performing classification using supervised learning models, we need to label the traffic to compare the model's predicted value with the true value. This is needed for training and testing. The raw data given to us does not contain labels. Initially, the SSL filter feature in Wireshark is crucial for selectively extracting HTTPS traffic [25]. The data must be split since packet information is bi-directional (from a local machine to a specific server). This uses a utility program called pk2flow [4]. An open-source library called nDPI, used for deep packet inspection, will extract and label each flow. The packet correlated with a particular flow will be assigned the corresponding label. Bayat et al.[3] constructed a script differentiating incoming and outgoing packets. The scripts generated a 4-tuple for every TCP connection, comprising the source IP, destination IP, source port number, and destination port numbers. Pairing these TCP connections with the reversed source and destination IP/port configuration allowed them to quickly identify two-way configurations with specific servers. They then filter out all unknown Sever Name Indications (SNIs). Thus focusing solely on familiar entities. Furthermore, removing unnecessary characters like numbers and dashed ensures a simplified labelling process and accurately determines the direction of packet traffic [3]. Lotfollahi et al. ensured that all transport layer segments (like TCP or UDP) have the same header length by adding zeros to the end of the UDP headers to match the length of TCP headers [16].

6.2.2 Feature extraction and transformation. Lotfollahi et al. discuss the issue of varying packet lengths. As neural networks require fixed-size inputs, the options are truncating packets to a fixed length or adding zeros to standardise lengths. Their observations indicated that most packets have a payload length of less than 1480 bytes. Therefore, optimising to keep the IP header and first 1480 bytes of each packet resulted in a 1500-byte vector as input for the neural network. For packets shorter than the payload length, they appended zeros to the end [16]. Furthermore, using bytes as input features for our deep-learning model ensures faster classification [28]. Additionally, encoding to bytes for us to analyse encrypted and unencrypted packets with greater precision and accuracy [16] [28]. To avoid overfitting due to reliance on IP addresses, we mask them in the IP header. This will allow the neural network to focus on relevant features for accurate classification [16].A Python open-source library called *scapy* processes packets and flows found in PCAP files. Some methods extract IP payloads from packets and convert them to bytes. Thus obtaining the features needed for each packet. After raw data extraction from the packets, it is transformed into an appropriate format so a given model can learn from it. MLP, 1D-CNN, SAE, and LSTM models are one-dimensional, so the byte stream can be given as input [28].

6.2.3 Balancing Datasets. Balanced datasets are crucial for machine learning classification models. An imbalance in the training data leads to reduced classification performance [16]. When classes are imbalanced, models may inadvertently prioritise the majority class, reducing accuracy for minority classes. Due to the amount of packet data collected, there will be enough examples to represent all classes. Furthermore, we can remove examples from the majority classes until we achieve classes of the same size.

6.3 Establishment of Benchmarks

To compare our models, we have decided to use a benchmark model. This model will act as the model that we will try to beat in terms of accuracy and justify using more complex models. We have decided to implement a CNN as Tooke, Weisz and Dicks achieved the highest accuracy with this model. However,it was computationally inefficient as it was 30 times slower than their MLP implementation [7, 27, 29]. Furthermore, we will compare the model's accuracy, speed, and computational cost. The speed is based on the number of packets classified per second, while the computational cost is the number of operations needed to classify a packet.

6.4 Evaluating the models

To evaluate the performance of the models comparatively, we will use a set of traditional analytical approaches. This process involves the creation of K training and test sets, the average Error over the K train-test cycles will be used to measure the adaptability of a model to unseen data and compare the model's predictive power [9]. We will be using 2 sets of tests to measure the classification accuracy of each chosen model: the confusion matrix and the F_1 score. The confusion matrix measure 4 aspects of predictive power: Sensitivity, Specificity, Precision and Negative Predictive Value [9]. These values are then used to calculate the F1 score, which combines precision and recall to reflect the model's overall performance (this value ranges from 0 to 1, with 1 indicating perfect precision and recall). The F_1 score is a popular metric for evaluating machine

learning models; this will allow us to measure our models' performance against the models implemented in related works such as [16].

The computational efficiency of each model will be measured using libraries that can track the runtime and memory usage of each function. We will test these models over different machines with different resource constraints. Our models' runtime and memory consumption are pertinent to our specific problem. These factors will significantly affect the implementation's architecture and hyperparameter tuning stages. The built-in cProfile library will be used to measure the runtime of each function, and the *memory_profiler* package will be used to measure memory consumption. The *memory_profiler* package can extract the increase in memory consumption due to code execution and the maximum memory used by the code at any point [22].

6.5 Hyperparameter Tuning

The hyperparameters are values passed before training that specify the details of the learning process. These values are traditionally found through trial and error, although there are a few sophisticated ways of making this process more efficient. We will use Python's Keras library to control the hyperparameters of our models, and we will take advantage of the open-source AiSara tuner's algorithm to expedite the process of searching for optimal hyperparameters. Pon et al. [24] found the AiSara tuner to have a better search time, cost and complexity than the standard Keras tuner library. AiSaratuner [13] is an open-source library that utilises Latin hypercube sampling, a statistical method for generating near-random samples of parameter values. Each randomly generated hyperparameter combination is tested, and the model with the lowest cross-validation error is returned.

7 ETHICAL, PROFESSIONAL AND LEGAL ISSUES

We will be using a dataset collected from the Ocean View Community, which raises ethical concerns around confidentiality. However, the data has been ethically collected, stored, and utilised by other research projects at the University of Cape Town. We will not use the data to identify individuals and will abstract IP and MAC addresses to prevent any personal information from being linked to the dataset. The data is securely stored on the University of Cape Town's NAS server, accessible only to authorised personnel. We will categorise traffic into types (e.g., video streaming, social media) without revealing specific details. to protect the Ocean View community's reputation. [29]

8 ANTICIPATED OUTCOMES

The anticipated outcome of this research are attaining deep learning models that classify network traffic with enough accuracy and computational efficiency to be used in a community network. We will benchmark our hybrid models against a CNN implementation and anticipate producing a model with a classification accuracy close to the 90.1% accuracy achieved by Weisz et al.'s [29] 2D-CNN implementation. Furthermore we anticipate the production of a model which achieves the classification speed benchmark of 10000 packets per second while maintaining a suitable accuracy to prove its suitability for real-time classification.

9 PROJECT PLAN

This section will provide a plan to complete this project before the deadline. We will outline the risks, timeline, required resources, deliverables, milestones, and work allocation.

9.1 Required Resources

One of the primary resources required for any deep learning project is hardware that can train these models in ample time. Without that, the project would take too long. Graphical processing units (GPU) provide a solution to this problem[20]. GPUs have multiple cores to train these models in parallel. We will use the GPU provided by Google Colab to speed up training times. Furthermore, UCT's High-Performance Cluster (HPC) can also be used to train these models if we need something even faster than Google Colab.

Data is also required for any machine learning-based project. Without a sufficient amount of data, we will not be able to train the model, will not be able to learn the patterns in the data, and could suffer from being underfitting. Fortunately, we have plenty of data at our disposal. The PCAP files we will use to train and test the model are from the OceanView community network in Cape Town courtesy of Inethi Technologies [12]. To access the data we require a VPN to connect to UCT to view and use the PCAP files. Many gigabytes of data in those files will be more than sufficient to train our models.

Lastly, some specific software will be required in this project. The project will be coded in Python and will use the TensorFlow library. This library provides models that do not need to be coded from scratch. It provides a framework to train models, tune and deploy models [21]. Luckily, this software is free and can be used by anyone. Another piece of software that we will use is Github for version control.

9.2 Risks

We have identified many risks in this project that could disrupt our work or slow progress. We must recognise these risks now so we are aware of them and can do our best to prevent them or manage them if they present themselves. We have created a risk matrix in the appendix 1.

9.3 Timeline

This project started on the 22nd of February when we received our projects. From here, we are not expected to implement the project until our supervisor accepts our proposal on the 30th of April. From here we will all work on our project, with various submissions due throughout the year. To allocate our time and energy effectively, we will use a Gannt chart to show who is doing what task and how long they will be expected to work on it. This will give us a good indicator of if we are on track to reach the final submissions for various tasks. The chart can be found in the appendix.

9.4 Milestones

Milestones are points in a project's life-cycle that measure its progress. The table below contains the milestones and dates given to us before we started this project.

Date	Milestone
22-25 April	Project Proposal presentations
30 April	Project Proposal due
22-26 July	Project Progress Demonstration
23 August	Complete Draft of final paper due
30 August	Project Paper Final Submission
9 September	Project Code Final Submission
16-20 September	Final Project Demonstration
27 September	Poster Due
4 October	Website Due
22 October	School of IT Showcase

9.5 Deliverables

Below, we will outline the deliverables that are required for this project:

- Three literature reviews, with each member doing their review. This details other related work in this field and familiarises each member with the concepts in their project.
- A project proposal
- A demo to show our supervisor what we have been working on and for feedback.
- A final paper from each member detailing our techniques and results.
- Final version of our software, which we will submit individually. This will be the completed version of each of our project
- Project poster that outlines our findings and summarises our project to someone who was not involved in the development
- website, which details our project's needs and results and contains our final papers.

9.6 Work Allocation

Project Work will be allocated among the team members. The team will work together to construct the preprocessing instruments required for handling the raw data PCAP file sets, providing consistent labelling and data flows for the different models. Furthermore, the team will work to have a baseline model for comparison that individual deep learning models can be benchmarked against. Thereafter, individuals will construct a deep learning model for the classification task. Thus, the model building, training and hyperparameter-tuning, testing, and evaluation will be done separately. The reporting and comparison of the models in terms of accuracy, performance, and resource utilization will be conducted together.

9.6.1 Boitumelo: Boitumelo will be implementing a SAE-RNN hybrid models, with a focus on LSTM and GRU hybrid architectures. His role during development will be the software architect, he will organize the code-base and set up the experiment infrastructure.

9.6.2 Ryan: Ryan will also implement an SAE hybrid, with two different CNNs as the second set of implementations. He will be in charge of data analysis and preprocessing during.

9.6.3 Michael: Michael will implement a CNN-RNN hybrid. As project manager, he will set up meetings, ensure the team communicates effectively, and meet the deadlines.

REFERENCES

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

[2] Azab, A., Khasawneh, M., Alrabaee, S., Choo, K.-K. R., and Sarsour, M. Network traffic classification: Techniques, datasets, and challenges. *Digital Communications and Networks* (2022).

[3] Bayat, N., Jackson, W., and Liu, D. Deep learning for network traffic classification. *arXiv preprint arXiv:2106.12693* (2021).

[4] caesar0301. pkt2flow. https://github.com/caesar0301/pkt2flow, 2014.

[5] Cheng, B. Guide to rnns, grus, and lstms with diagrams and equations. https://medium.com/@bobbycxy/guide-to-rnns-grus-and-lstms-with-diagrams-and-equations-19065aa61cac, Unknown 2021. Website.

[6] Choi, K., Fazekas, G., and Sandler, M. Explaining deep convolutional neural networks on music classification. *arXiv preprint arXiv:1607.02444* (2016).

[7] Dicks, M., Tooke, J., and Weisz, S. A comparative evaluation of deep learning approaches to online network traffic classification for community networks.

[8] D'Angelo, G., and Palmieri, F. Network traffic classification using deep convolutional recurrent autoencoder neural networks for spatial–temporal features extraction. *Journal of Network and Computer Applications 173* (2021), 102890.

[9] Gareth James, Daniela Witten, T. H., and Tibshirani, R. *An Introduction to Statistical Learning with Applications in R.* Springer, 2023.

[10] Goodfellow, I., Bengio, Y., and Courville, A. *Deep learning.* MIT press, 2016.

[11] Hochreiter, S. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 6*, 02 (1998), 107–116.

[12] iNethi Technologies. inethi technologies. https://www.inethi.org.za/, 2024. Accessed: March 24, 2024.

[13] Intelligence, A. A. aisaratuners.

[14] Liu, G., Bao, H., and Han, B. A stacked autoencoder-based deep neural network for achieving gearbox fault diagnosis. *Mathematical Problems in Engineering 2018* (2018), 1–10.

[15] Lopez-Martin, M., Carro, B., Sanchez-Esguevillas, A., and Lloret, J. Network traffic classifier with convolutional and recurrent neural networks for internet of things. *IEEE access 5* (2017), 18042–18050.

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

[17] Martínez-Cervantes, L., and Guevara-Martínez, R. Community networks and the quest for quality.

[18] Mersico. Understanding 1d, 2d, and 3d convolution network. https://www.kaggle.com/code/mersico/understanding-1d-2d-and-3d-convolution-network, Unknown Unknown. Website.

[19] Micholia, P., Karaliopoulos, M., Koutsopoulos, I., Navarro, L., Vias, R. B., Boucas, D., Michalis, M., and Antoniadis, P. Community networks and sustainability: a survey of perceptions, practices, and proposed solutions. *IEEE Communications Surveys & Tutorials 20*, 4 (2018), 3581–3606.

[20] Owens, J. D., Houston, M., Luebke, D., Green, S., Stone, J. E., and Phillips, J. C. Gpu computing. *Proceedings of the IEEE 96*, 5 (2008), 879–899.

[21] Pang, B., Nijkamp, E., and Wu, Y. N. Deep learning with tensorflow: A review. *Journal of Educational and Behavioral Statistics 45*, 2 (2020), 227–248.

[22] Pedregosa, F. memory_profiler library.

[23] Pinkus, A. Approximation theory of the mlp model in neural networks. *Acta numerica 8* (1999), 143–195.

[24] Pon, M. Z. A., and KK, K. P. Hyperparameter tuning of deep learning models in keras. *Sparklinglight Transactions on Artificial Intelligence and Quantum Computing (STAIQC) 1*, 1 (2021), 36–40.

[25] Shbair, W. M., Cholez, T., Francois, J., and Chrisment, I. A multi-level framework to identify https services. In *NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium* (2016), IEEE, pp. 240–248.

[26] Thompson, N. C., Greenewald, K., Lee, K., and Manso, G. F. The computational limits of deep learning. *arXiv preprint arXiv:2007.05558* (2020).

[27] Tooke, J. Deep learning classifiers for qos and traffic engineering in community networks.

[28] Wang, P., Ye, F., Chen, X., and Qian, Y. Datanet: Deep learning based encrypted network traffic classification in sdn home gateway. *IEEE Access 6* (2018), 55380–55391.

[29] Weisz, S., and Chavula, J. Community network traffic classification using two-dimensional convolutional neural networks. In *International Conference on e-Infrastructure and e-Services for Developing Countries* (2021), Springer, pp. 128–148.

[30] Zhang, J., Chen, X., Xiang, Y., Zhou, W., and Wu, J. Robust network traffic classification. *IEEE/ACM transactions on networking 23*, 4 (2014), 1257–1270.

10 APPENDIX A: RISK MATRIX

Table 1: Risk Matrix

Risk	Probability	Impact	Consequence	Mitigation	Monitoring	Management
Loss of a Team Member	Low	Medium	The sections of the project will either be left out of the final product or completed by the remaining members	knowledge sharing so that other team members are aware of each other's responsibilities and progress	support for one another and monitoring signs of potential team member loss	consistently updated plan for redistribution of work amng remaining team members to ensure research objectives are fulfilled
Insufficient hardware resources	low	medium	inability to efficiently train models and effectively perform hyperparameter optimisation	use of Google Colab or UCT's High Performance COmputing grid	Regular benchmarking of code to ensure optimisation and find unexpected delays in code	Compile a list of options to outsource hardware resources. Alternatives to Google colab such as Amazon Sagemarke and Azure ML
Loss of Source Code	Low	High	Disruption in project schedule. Having to re-write large portions of the software	Use of GitHub to store multiple backups and versions of source code on multiple devices	Ensuring all members regularly commit and push to the github repository	Pulling updated clones of the project source code from team member's commits on github
Insufficient programming skill/experience	Low	Medium	inability to implement software part of the project due to lack of programming skill causing delays in schedule	allocating enough time during over the next few weeks to online tutorials and practice	Peer reviewing team member's code using github to ensure their programming is of quality	Compiled list of online resources and regular team meeting discussing deep learning understanding
inability to meet project deadline	low	high	large deduction of mark and failure to meet course requirements	Regular communication with supervisor and between team members to ensure work is being done within the schedule's constraints	regular progress checks with reference to the Gantt chart and supervisor expectations	Consultation with project supervisor to redefine scope in order to complete a quality project on time

11 APPENDIX B: GANTT CHART

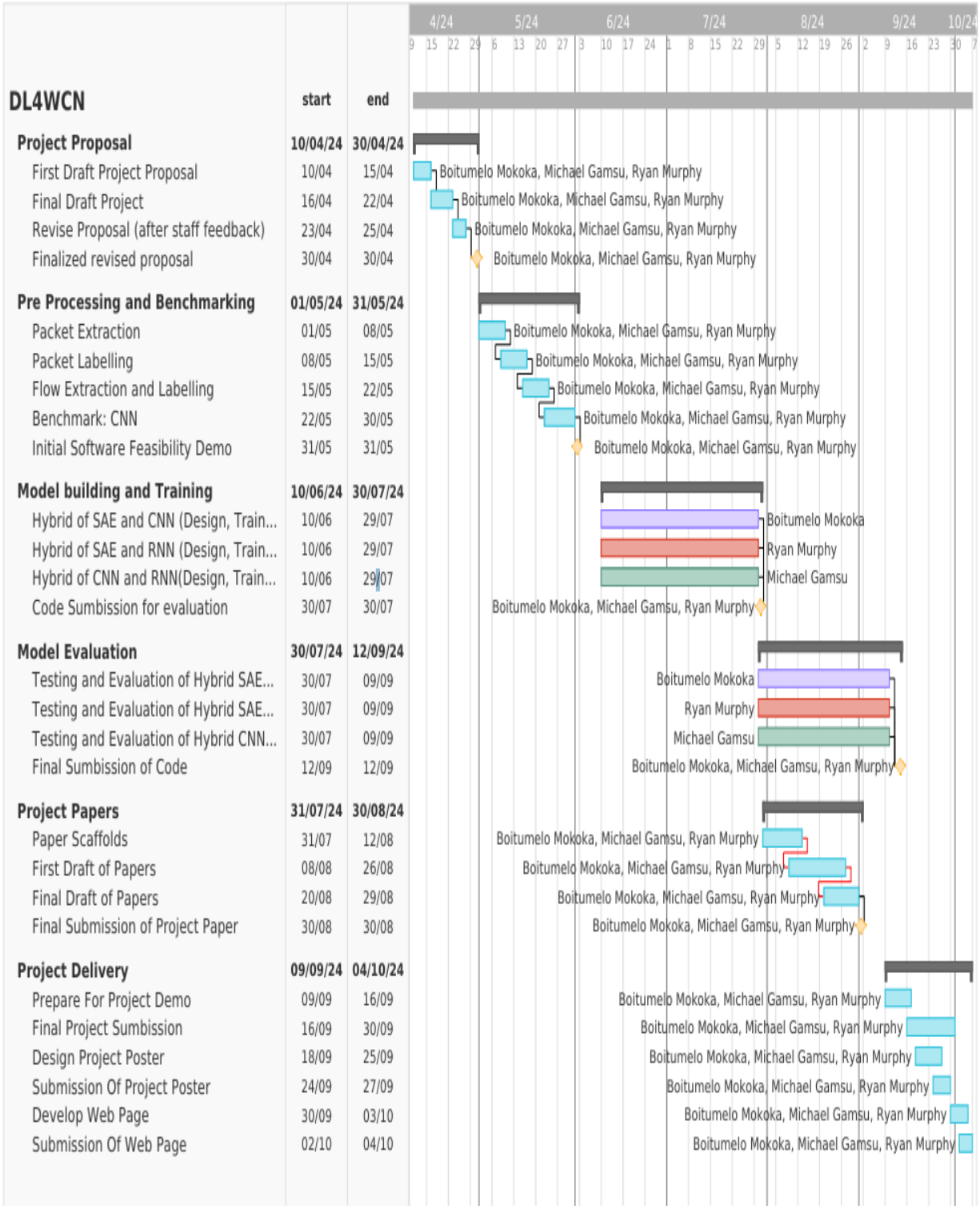

Figure 1: Gantt Chart