UNIVERSITY OF CAPE TOWN

DEPARTMENT OF COMPUTER SCIENCE

CS Honours Project Final Paper 2024

Title: Traffic Classification using Hybrid Deep Learning for Community Networks

Author: Michael Alec Gamsu

Project Abbreviation: DL4WCN

Supervisor(s): Prof Josiah Chavula

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

Traffic Classification Using Hybrid Deep Learning for Community Networks

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

ABSTRACT

Network classifiers are crucial for improving network quality of service (QoS) and optimising traffic engineering. Traditional methods struggle to classify encrypted network traffic effectively. Furthermore, machine learning models need to be more computationally expensive and generalised as effective classifiers for community networks. Therefore, deep learning techniques provide practical and realistic solutions to overcome the computational constraints of community networks. To address these limitations, the exploration of hybrid deep learning models, combining Convolutional Neural Networks (CNNs) with Recurrent Neural Networks (RNNs), specifically Long-Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks are considered. These architectures leverage CNN's feature extraction capabilities with the temporal sequence learning strengths of RNNs. The models compared include 2D-CNN-LSTM, 2D-CNN-GRU, 1D-CNN-LSTM, and 1D-CNN-GRU, alongside simpler models such as Multi-Layer Perceptrons (MLP), 2D-CNN, and 1D-CNN. Initial results show that hybrid models, particularly 1D-CNN-LSTM and 1D-CNN-GRU, deliver the highest accuracy in traffic classification. However, this has significant trade-offs in computational demands, impacting their feasibility in resource-constrained settings. In contrast, the 2D-CNN hybrids offer a potential middle ground, balancing performance with reduced computational overhead.

KEYWORDS

Deep Learning, Community Networks, Traffic Classification, Neural Networks.

1 INTRODUCTION

Network traffic classification refers to categorising network traffic data based on their associated application (e.g., Gmail, Facebook) or protocol type (e.g., TCP, UDP). It plays a vital role in network engineering and management, helping to enhance quality of service (QoS) and ensure network security for users by prioritising specific applications [2]. Real-time classification can prioritise traffic flows to particular applications, improving user experience. Furthermore, it provides valuable insights into network behaviour, aiding administrators in making informed decisions regarding resource allocation, security, and anomaly detection. Community Networks offer affordable connectivity solutions to underserved communities [17]. These Community Networks stand to benefit significantly from network traffic classification as they prioritise critical applications, detect anomalies early, and utilise bandwidth more efficiently. By implementing effective traffic classification, community networks can enhance overall network performance, improve user experience, and support sustainable, scalable growth in underserved regions [10]. As deep learning techniques evolve, they present new opportunities for cost-effective, reliable, and secure classification, particularly when identifying encrypted traffic [28].

Historically, traditional classification methods such as port-based approaches, payload-based approaches, and Deep Packet Inspection (DPI) have faced limitations, particularly in modern IoT technologies. For example, port-based methods suffer from port obfuscation, where applications no longer adhere to standard port usage [29]. DPI, while effective, only works on unencrypted data and is computationally expensive [13]. Similarly, machine learning approaches like random forests and k-nearest neighbour algorithms require significant manual feature engineering and lack generalisability [21].

With the recent strides in deep learning, particularly in encrypted traffic classification, there is a promising opportunity to significantly enhance network performance, engineering and security. This study delves into the potential of hybrid deep learning architectures for traffic classification in community networks, leveraging the combined strengths of Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) to capture spatial and temporal patterns in network traffic.

1.1 Research Problem Statement

This paper will evaluate and compare the performance of hybrid architectures, such as 1D-CNN with LSTM, 1D-CNN with GRU, 2D-CNN with LSTM, and 2D-CNN with GRU, against standalone models like MLP, 2D-CNN and 1D-CNN models. This research explores which models are best suited for real-time traffic classification while operating within the hardware constraints of community networks. This involves evaluating accuracy and performance while minimising the number of trainable weights.

1.2 Research Questions

The following research questions are proposed:

(1) Classification Accuracy: How do hybrid deep learning architectures, which combine the strengths of CNNs and RNNs, enhance classification accuracy compared to simpler models like MLP and CNNs?
(2) Packet Processing Speed: Are these hybrid deep-learning architectures appropriate for real-time traffic classification? How do they compare to MLP and CNNs regarding packet processing speed, a crucial factor in real-time classification?
(3) Memory optimisation: How do different deep learning model architectures impact memory consumption as the number of parameters increases, and what trade-offs exist between

model complexity and computational efficiency in network traffic classification tasks?

The remainder of the paper is structured as follows. Section 2 provides the necessary background context for the remainder of the research. Section 3 and 4 presents an overview of the research concerning the utilisation of deep learning in traffic classification. Section 5 and Section 6 cover the experiment design, including data collection, preprocessing, deep learning architectures, and experimental methodologies. Finally, Section 7 will present the results, while Section 8 will delve into the key insights. The paper will conclude with final remarks in Section 9.

2 BACKGROUND

Before diving into our study on hybrid deep learning for traffic classification for community networks, it is essential to clarify a few concepts for the rest of the paper. Firstly, a clear understanding of what community networks mean, highlighting the role of providing affordable connectivity to underserved areas. Next, an explanation of what Quality of Service (QoS) entails. It finally highlighted the relevance of traffic classification in community networks and demonstrated how it can enhance QoS in underserved communities.

2.1 Community Networks

Community Networks provide hope for extending the Internet's reach to low-resource and rural areas. These networks are developed and maintained by their respective communities, which provide the necessary network infrastructure and access points connecting them to the larger Internet. Despite being self-managed and non-centralised, these networks operate with the principles of neutrality, freedom, and openness, making them a promising solution for affordable and accessible internet connectivity. Consequently, the administration, scalability, service quality, and cost of deploying and maintaining these networks constrain community networks. [17]. Community Networks are cheaper solutions to the digital divide as they provide affordable connectivity to impoverished and remote areas.

2.2 Quality of Service (QoS)

Quality of Service (QoS) is to evaluate the services provided by telecommunications. The four main metrics are latency, jitter, bandwidth and packet loss. These parameters can collectively determine a satisfactory and reliable performance of telecommunication services provided to the user over a network [20]. Furthermore, Quality of Experience (QoE) is a measure beyond the QoS and focuses on the user's perception of the overall quality of telecommunication services. Therefore, it ensures that the perceived services are satisfactory with user specifications. QoE relies on user satisfaction with content access and application utility [18].

2.3 Traffic Classification in Community Networks

Effective traffic classification enhances network QoS by prioritising essential services, thus ensuring efficient use of limited resources. It supports security measures by identifying and prioritising secure applications and traffic types, helping to mitigate potential threats [16]. Optimising resource use through traffic classification also reduces operational costs and delays costly upgrades, which is crucial for maintaining the economic viability of community networks [4] [24]. Additionally, understanding traffic patterns through classification aids in strategic planning and improving service delivery, ultimately enhancing user satisfaction and network reliability [18] [15]. However, deploying modern classification models, which often require significant computational and memory resources, can be impractical at network access points in these environments. This highlights the opportunity for deep learning that balances performance, accuracy, and resource efficiency.

3 NEURAL NETWORK ARCHITECTURES

3.1 Deep Learning Models

Deep learning models are a subset of machine learning algorithms that utilise neural networks with multiple layers to learn complex patterns in data. These models have revolutionised various fields by enabling machines to perform highly accurate tasks such as image recognition, natural language processing, and traffic classification. Unlike traditional models, deep learning models can automatically discover features from raw data, reducing the need for manual feature engineering. This section provides an overview of some fundamental deep learning models, including Multi-layer Perceptrons (MLPs), Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Hybrid Deep Learning Models, each of which brings unique strengths to different types of data processing tasks.

3.1.1 Multi-layer Perceptrons (MLP). MLPs are a feedforward model. They consist of neuron layers 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.The process repeats until it outputs a value at the end. MLPs adapt their weights and biases using a backpropagation algorithm, which creates the illusion of "learning". MLPs use for both prediction and classification problems[19]. These models serve as the foundation for more complex models.

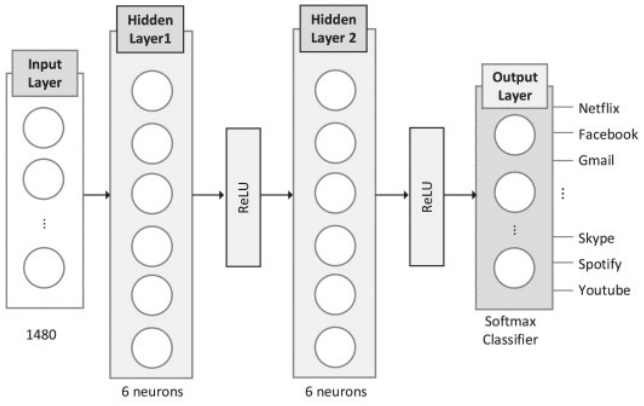

Figure 1: Multilayer Perceptron (MLP) model structure

3.1.2 Convolutional Neural Networks (CNN). Convolutional Neural Networks is an advanced deep learning model that receives input data and puts it through convolutional layers [16]. Convolutional layers are a set of kernels that identify vital characteristics. Furthermore, CNNs use a pooling layer that reduces the spatial size of the feature maps, thus decreasing the computational load and controlling overfitting [3]. CNNs lessen the number of learnable parameters by reusing the same kernels across the entire input. Parameter sharing and sparse interactions will decrease the computational complexity compared to other models [11]. CNNs have become extremely popular for computer vision and image recognition by using two-dimensional CNNs (2D-CNNs) to capture two-dimensional features [3]. Additionally, adjusting the model filter for spatial patterns in one dimension (1D-CNN) has succeeded in traffic classification for packet data. The convolutional layer recognises the spatial relationship between adjacent bytes, resulting in effective traffic classification [23]. According to Aceto et al.[1] and Wang et al. [27], the 1D-CNN is better at capturing sequential data than 2D-CNNs.

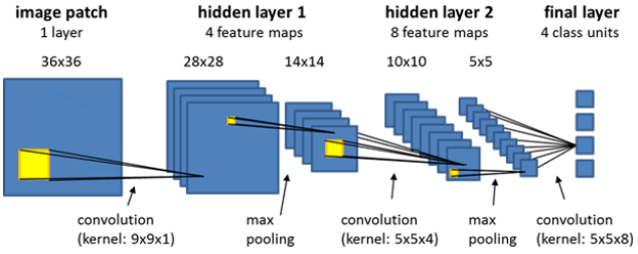

Figure 2: Convolutional Neural Networks (CNN) model structure

3.1.3 Recurrent Neural Network (RNN). A Recurrent Neural Network is designed for sequential data processing, whereby the current state depends on the current input and previous input. The model achieves this by using loops in the hidden layers, allowing neurons to retain temporal information [21]. There are two popular variations of RNNs: Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRU). These variations are solutions to the vanishing gradient problem evident in RNN. LSTM and GRU incorporate a gated mechanism to regulate the flow of the model. Furthermore, they can handle variable-length sequences. Both capture long-term dependencies in temporal patterns, crucial for flow-based traffic classification.

3.1.4 Hybrid Deep Learning Models. A hybrid model aims to integrate the different models into a single framework to leverage their complementary strengths to improve performance. Dang et al. [8] discuss that hybrid models consisting of a CNN and RNN are outperforming methods with a relatively high overall accuracy [8]. Therefore, combining both architectures can discern the intricate patterns in the encrypted data. The hybrid model can enhance its computability and accuracy considering the resource-constrained environment.

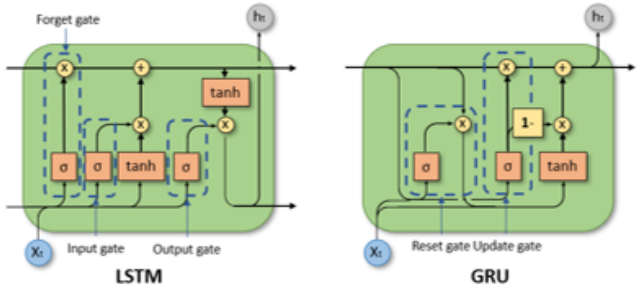

Figure 3: Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRU) structure

4 RELATED WORK

Network traffic classification is crucial for network management and quality of service (QoS) optimisation. Recently, Deep Learning and Machine Learning have been explored to classify encrypted network traffic accurately. The primary methods are deep packet inspection (DPI), port-based inspection, flow classification, and packet classification [28]. DPI examines packets using the headers and application signatures. However, privacy concerns and computational complexity make this challenging for community networks [16]. Port-based inspection uses the TCP/UDP headers, except it falls short with port obfuscation [29]. Flow-based classification involves aggregating packets into flows based on unique identifiers: source IP, packet IP and source and destination port number [4]. However, Rezaei et al. [21] do not recommend flow-based classification methods for real-time encrypted traffic classification due to their high latency, resource-intensive nature, difficulty in handling new or short flows and scalability issues [21]. Packet classification offers a more granular approach compared to flow-based methods, providing detailed analysis at the packet level by examining features such as protocol, payload content, and headers. [16]. This method's precision makes it particularly effective in identifying and classifying specific types of traffic, even within encrypted streams. Notably, studies by Wang et al. [26] and Lim et al. [14] have demonstrated the efficacy of packet classification in malware detection, further highlighting its potential for various network security applications. These successes underscore the relevance of packet classification as a robust technique in the ongoing research for network traffic classification implementation.

This study will employ packet-based classification to identify applications within community networks. Recent research has leveraged distinct, cleaned public datasets, focusing on identifying various layers or protocols within the Internet protocol suite, which makes direct comparisons between studies challenging. Lotfollahi et al. [16] employed a deep learning approach using 1D-CNNs, achieving an impressive average F1 score of 94% in their Deep Packet inspection method. However, as discussed earlier, Deep Packet inspection raises significant privacy concerns and is computationally intensive, particularly for low-resource communities. Moreover, Lotfollahi et al. [16] concentrated on application and protocol classification. This paper focuses on analysing application classification and computational performance for low-resource environments. Bayat et al. [5] used packet-based and flow-based

data techniques. Their research demonstrates a hybrid CNN and GRU, which achieved an accuracy level of 95%. Although they focused exclusively on HTTPS services [5]. The Figure below is an illustration of the Bayat hybrid model: Limited research compares

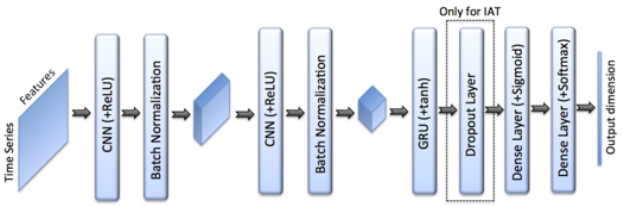

Figure 4: CNN-GRU model structure [5]

network traffic classification and computational performance for real-time classification in low-resource environments. A related study by Tooke, Weisz, and Dicks [10, 25, 28] explored the performance of machine learning and deep learning algorithms for low-resource community networks. They implemented a 1-D CNN, 2-D CNN and LSTM and compared the performance and accuracy to a Support Vector Machine (SVM). They concluded that the SVM performed better in packet per second but worse in accuracy due to more parameter counts of other models. The model with the highest accuracy was their 1D-CNN with a maximum of 88.64% [28]. Therefore, the focus was to implement hybrid models that can achieve high precision and performance while limiting parameters.

5 EXPERIMENT DESIGN AND EXECUTION

Constructing a preprocessing pipeline involves transforming the raw network packet data in the PCAP files from the Ocean View Community network in Cape Town into an output format suitable for traffic packet classifier development.

5.1 Obtaining Network Traffic Data

Ocean View is the community network where traffic data originates and is used to train and test our deep-learning models. 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 ethical approval to use the data for the research.

5.2 Preprocessing

The preprocessing stage generated the training, testing and validation data from a set of PCAP files. The PCAP files were the input collected from Ocean View Community Network by the University of Cape Town. This stage consists of packing label extraction from the raw data to be used as features for our model.

5.2.1 Packet Labbeling. This project aims to design a deep-learning model that performs traffic classification. Classification is a supervised learning task whereby every instance must be accompanied by a label that ensures the neural network learns from mistakes and adjusts parameters accordingly. The PCAP files have no prior

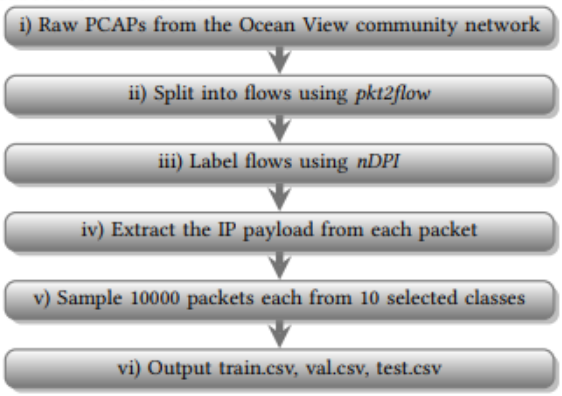

Figure 5: Preprocessing structure [28]

labelling associated with the raw data. Thus, raw traffic data must be labelled to compare the model's predicted and actual values. This is required for both the training and testing phases. Initially, the SSL filter feature in Wireshark is crucial for selectively extracting HTTPS traffic [22]. Then, 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* [7]. An open-source deep packet inspection (DPI) library called *nDPI* extracts the packet correlated with a particular flow and assigns the corresponding label [9]. Furthermore, Bujlow [6] reports that nDPI is more reliable and accurate than other packet label programs.

5.2.2 Extract IP payload from each packet. Bayat et al.[5] 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). Furthermore, removing unnecessary characters like numbers and dashed ensures a simplified labelling process and accurately determines the direction of packet traffic [5]. Lotfollahi et al. [16] 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].

5.2.3 Feature extraction and transformation. Lotfollahi et al. [16] report the concern of packets with varying lengths. As neural networks require fixed input sizes, the options are eliminating packets, ensuring all packets are fixed, or incorporating 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 [26]. Additionally, encoding to bytes allows us to analyse encrypted and unencrypted packets with greater precision and accuracy [16] [26]. To avoid overfitting due

to reliance on IP addresses, we mask them in the IP header. This allows 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, and LSTM models are one-dimensional, so the byte stream can be given as input [26].

5.2.4 Train, Test and Validation Subsets. The dataset, comprising a single label and 1480 features, necessitates division into training, test, and validation sets. The training set facilitates model learning and weight adjustment based on loss function output, while the validation set assesses model performance on unseen data and identifies potential overfitting. Ultimately, the test set gauges the final model's performance. This process entails a split of 64% for training, 16% for validation, and 20% for testing.

5.2.5 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. A threshold of 9,000 packets was set to filter out outliers to ensure a balanced dataset. Ten applications were manually selected for the study from the classes that met this criterion and had sufficient data. Each of these chosen application classes contributed 9,000 packets, although some classes originally contained up to 100,000 packets and were truncated to 10000 packets.

5.2.6 Reshaping for the CNNs. Using 1D and 2D CNNs means the data cannot be directly input into these models. In the context of 1D-CNN, a channel dimension was introduced to represent the interval between packets, thereby encoding the data within a 3D array [12]. For the 2D-CNN, the data must be formatted into a grid structure. In this format, darker pixels correspond to higher feature magnitudes, while lighter pixels correspond to lower feature magnitudes [28].

5.2.7 Hyper-Parameter Tuning. The hyperparameter tuning process was conducted through a series of manual experiments aimed at optimising the models' accuracy and performance. While grid search, a systematic technique for identifying the best combination of predefined hyperparameters, is a commonly used approach, it can be both computationally expensive and time-consuming. To enhance efficiency, we manually tested several key hyperparameters, including the learning rate, regularisation techniques, and additional architectural components. The learning rate, a critical factor influencing the model's ability to reach the global minima of the objective function, was tested with values of 0.00005, 0.0001, 0.0005, and 0.001. Although all learning rates produced similar results, the lower rates required more epochs to converge. Therefore, a learning rate of 0.001 was selected for all models, as it provided faster convergence without sacrificing accuracy. Various regularisation techniques were applied, including dropout and L2 regularisation to prevent overfitting. Dropout rates of 0.2, 0.1, and 0.3 were tested, and a 0.2 and 0.3 dropout rate was decided. Subsequently, L2 regularisation was applied with values of 0.001, 0.01, and 0.1, leading to better generalisation performance. Additionally, architectural enhancements such as BatchNormalization, Flatten, and MaxPooling were explored with different configurations to optimise model performance further. These techniques collectively helped fine-tune the models to achieve a balance between accuracy, training speed, and generalisation. Additionally, different kernel sizes ranging from 3 to 5 were tested. A kernel size of 3 was chosen to aim at keeping the model lightweight and efficient. Larger kernel sizes, such as 5, increase the number of parameters and computational complexity, which can lead to slower training times and higher memory usage, especially in resource-constrained environments. A kernel size of 3 strikes a balance by capturing sufficient spatial information from the input data while maintaining a smaller parameter footprint. The parameters were changed by varying the filter size of each layer. Different configurations of filter sizes were tested, and the trainable weights varied. Filter sizes from 4 to 128 were tested. Any larger was not considered lightweight enough for the model.

5.3 Implementing Deep Learning Architectures

This section outlines the model architectures designed for traffic classification in low-resource community networks. The Benchmark models are 2D-CNN, MLP and 1D-CNN. The hybrid models combine CNNs and RNNs layers to leverage spatial and temporal data dimensions, enhancing their ability to classify network traffic accurately.

5.3.1 Multi-layer Perceptrons (MLP). The MLP architecture comprises four dense layers. The first three layers use the ReLU activation functions to handle non-linearity and intricate data patterns from 1480 features. The final layer employs softmax activation to output probabilities suitable for multi-class classification, adapting based on the number of target classes.

5.3.2 Convolutional Neural Networks (CNN).

- 2D-CNN: Begins with three convolutional layers using ReLU activations to extract spatial patterns from 40x37 single-channel input data. It includes padding to preserve dimensions, a flattening layer, and a final softmax-activated layer with L2 regularisation to reduce overfitting.
- 1D-CNN: Consists of three 1D convolutional layers with ReLU activations to capture temporal patterns from input data shaped by features and channels, flattening the outputs for final class probability calculations via softmax.

5.3.3 Hybrid Convolutional and Recurrent Architecture.

- 2D-CNN with LSTM: Integrates a 2D-CNN front end (ReLU activation, 'same' padding, MaxPooling, Dropout at 20%) feeding into an LSTM layer for sequential data analysis, with additional dropout at 30% and Global MaxPooling, concluding with a regularised softmax output layer.
- 2D-CNN with GRU: This is similar to the LSTM version but replaces the LSTM with a GRU layer. It maintains the structure of the MaxPooling and Dropout layers, followed by Global MaxPooling and a softmax output.
- 1D-CNN with LSTM: Features two 1D convolutional layers with ReLU activation, 'same' padding, and MaxPooling to handle temporal data. It includes a 20% dropout layer

before an LSTM layer, followed by a 30% dropout and GlobalMaxPooling1D, ending with a softmax output layer for classification.
- Mirrors the LSTM model with a GRU layer for temporal pattern analysis. This model also uses two 1D convolutional layers, MaxPooling, with a 20% dropout, followed by a 30% dropout, GlobalMaxPooling1D, and a softmax output layer for final classification.

6 EXPERIMENTAL METHODOLOGY

The experiments designed aim to evaluate the performance of hybrid deep learning architectures compared to simpler models in traffic classification. This section outlines the design, methodology and rationale. The study explores the feasibility of hybrid deep learning models in terms of complexity, computational efficiency, memory and overall performance. The data outlined in section 5 corresponds to the training, testing and validation dataset used for the models. The experiments use the following evaluation metrics: accuracy, packet processing speed, and memory usage. Each experiment is tailored to address the specific research question mentioned in Section1.2. Each experiment was iterated ten times, and the average results for accuracy and packet processing speed (PPS) were recorded to ensure reliability. Memory consumption was integrated ten times until it reached a stable level. Central to this experiment is manipulating the number of filters in the model, which directly alters its trainable weights (total parameters) and complexity.

6.1 Experiment 1: Classification Accuracy

The first experiment is designed to assess the classification accuracy of hybrid deep learning models compared to simpler models. The objective is to ascertain how the amalgamation of CNNs and RNNs enhances accuracy in traffic classification. Accuracy is a commonly used metric to assess a classification model's performance.

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

Thus, comparing all the models' accuracy will determine the hybrid deep learning architectures. Hybrid models are anticipated to exhibit superior accuracy to simpler models, assuming that the hybrid model leverages the strengths of CNNs and RNNs. An alternative metric could be the F1 score, which provides a more nuanced approach to model performance for imbalanced datasets. However, since the dataset has been prepossessed and balanced, the F1 score will not be used.

6.2 Experiment 2: Packet Processing Speed (PPS)

The secondary experiment evaluates the packet processing speed (PPS) to determine the suitability of hybrid models for real-time traffic classification in low-resource environments. The same models used in the accuracy experiments were assessed for their computational efficiency. Upon training the model, the PPS calculates the inference speed. PPS measures the number of packets the model could classify per second from the test set. This process was iterated ten times to ensure accurate measurements. Given the higher computational demands of hybrid models, it is expected that the MLP will exhibit higher PPS. There is a threshold of 10,000 packets per second to ensure real-time performance in low-resource environments. Any model below this threshold does not meet the fast inference requirements necessary for real-time classification.

6.3 Experiment 3: Memory Optimisation

The third experiment investigates the memory optimisation of hybrid models by examining the memory consumption usage. A model's complexity and computational requirements are reflected in its number of parameters. Each parameter corresponds to a weight transforming input data during training and inference. Reducing parameters decreases memory usage (each parameter taking 4 bytes for a 32-bit float) and increases processing speed, making models more practical for real-time network classification in low-resource environments. However, fewer parameters also reduce the model's ability to capture complex patterns, negatively impacting accuracy. This experiment increased the number of filters to study the effect on performance, memory consumption, and accuracy. Increasing the number of parameters is anticipated to increase memory consumption and accuracy at the expense of performance. Assuming that low-parameter models are preferable to high-parameter models in low-resource settings require less memory. However, it highlights a trade-off between parameter reduction, memory consumption, accuracy, and processing speed to optimise deep learning models for low-resource environments.

7 RESULTS

The experiments evaluated the following models; MLP, 2D-CNN, 1D-CNN, 2D-CNN-GRU, 2D-CNN-LSTM, 1D-CNN-GRU, and 1D-CNN-LSTM, across varying parameters. The relationships between accuracy, prediction performance and memory usage across the different models This section will present the findings obtained from these experiments.

7.1 Accuracy Results

The initial experiment was designed to evaluate the accuracy of various hybrid deep learning models in classifying traffic against simpler models. Hybrid models, specifically 1D-CNN-LSTM and 1D-CNN-GRU, consistently exhibited the highest accuracy across multiple configurations. This discovery underscores the effectiveness of combining CNNs with RNNs, resulting in superior performance in traffic classification tasks. The graphs below in Figures 6,7 illustrate a clear trend where the accuracy improves with increasing complexity, suggesting that these more complex models are better at capturing and interpreting intricate patterns within the data. However, all models reach an accuracy plateau, indicating diminishing returns on model complexity.

In detail, the 2D-CNN-LSTM and 2D-CNN-GRU hybrids achieved peak accuracy of 94.07% and 92.14%, respectively, outperforming the standard 2D-CNN and the MLP, which recorded maximum accuracies of 91.24% and 88.46%, respectively in Figure 6. This data emphasises the advantage of hybrid architectures in handling the spatial and temporal features of network traffic.

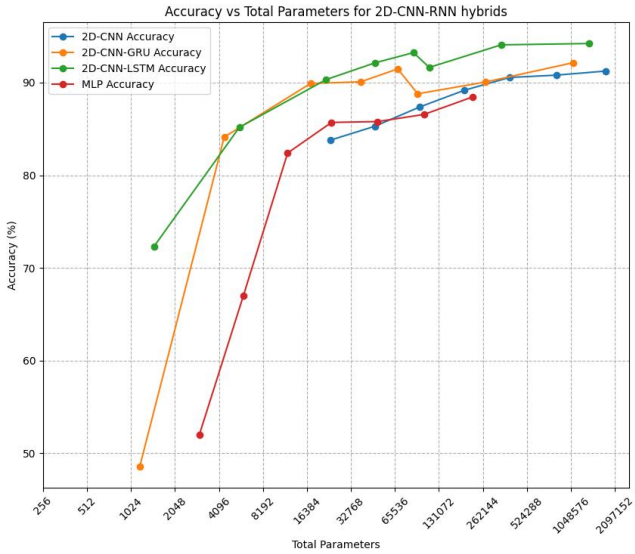

Figure 6: Comparison of Model Accuracy Across Different Architectures and Parameter Sizes for 2D-CNN-RNN Hybrids

Similarly, in Figure 7, the 1D-CNN-LSTM and 1D-CNN-GRU reached the highest accuracy scores of 94.3% and 94.21%, respectively, closely followed by the 1D-CNN model at 92.64%. These findings reaffirm that while hybrid models present the best performance, simpler models such as the 1D-CNN remain significantly applicable for low-resource communities, especially in scenarios where computational resources are a limiting factor.

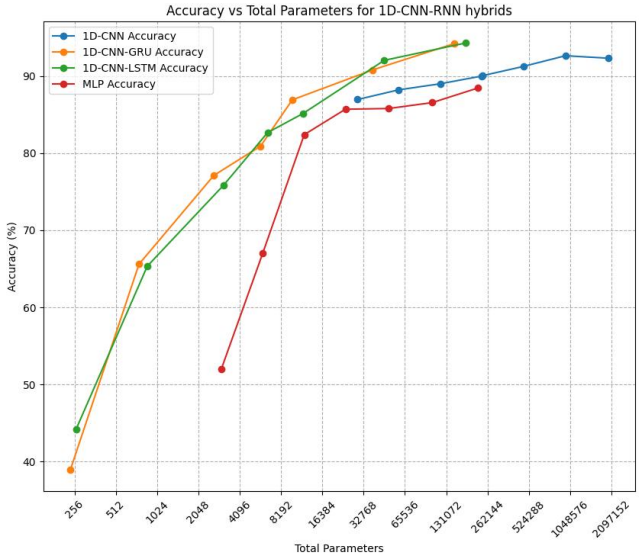

Figure 7: Comparison of Model Accuracy Across Different Architectures and Parameter Sizes for 1D-CNN-RNN Hybrids

7.2 Packet Per Second Results

The second experiment assessed the packet per second (PPS) to determine whether hybrid models could handle real-time classification in resource-constrained environments. The PPS is critical for further inference on community network access points.

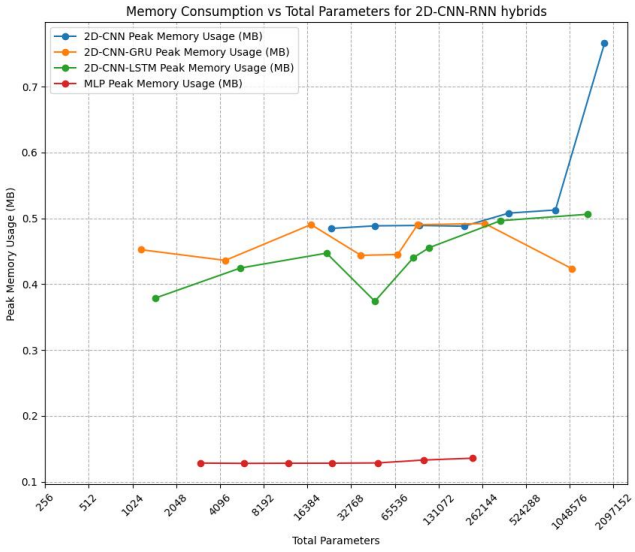

Figure 8: Comparison of Model Performance in Terms of Packets Per Second for 2D-CNN-RNN Hybrids Across Various Parameter Sizes

Figure 8 demonstrates that the MLP outperformed all other models in terms of performance, maintaining a consistent 16000 PPS across all parameter configurations. These results highlight the MLP's computational efficiency, making it suitable for real-time classification in low-resource environments. The 2D-CNN performed moderately well, processing between 10000 and 15000 PPS for all parameters except the highest parameter count tested. Therefore, it is a robust model in terms of performance. The 2D-CNN-LSTM and 2D-CNN-GRU initially performed well. The hybrid was above the threshold of 10000 PPS with smaller parameters, but their processing speed decreased substantially as the complexity increased. Figure 9: The critical observation is that the 1D-CNN model shows a sharp decline in PPS as the total parameters increase, eventually dropping below the threshold. The 1D-CNN-GRU and 1D-CNN-LSTM display relatively consistent PPS, though they are considerably slower, ranging between 2000 and 4000 PPS. 1D-CNN-GRU showed a slightly better PPS than 1D-CNN-LSTM, achieving 2,211.08 PPS with 150,154 parameters, which suggests that GRU-based architectures are simpler than LSTM. Therefore, a trade-off between computational performance and accuracy is required in the convolutional models. While the MLP was consistent in performance

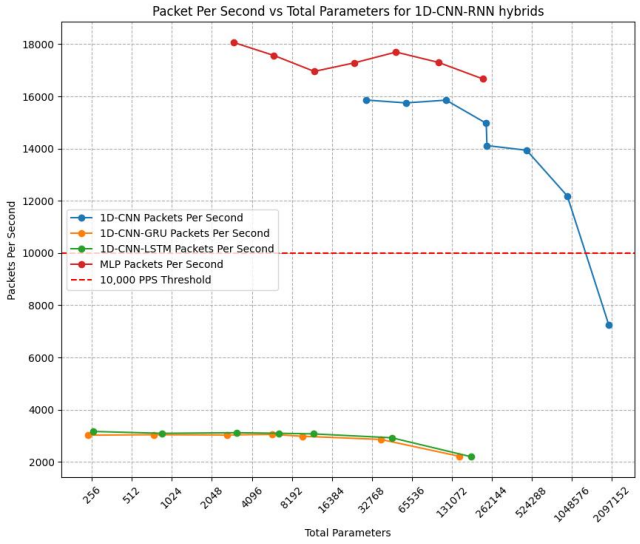

Figure 9: Performance Comparison Across Different 1D-CNN-RNN Hybrid Architectures as a Function of Parameter Size

7.3 Memory Optimisation Results

The third experiment provides an analysis of various deep learning architectures' impact on memory consumption as the number of parameters increased. We aimed to investigate the trade-offs between model complexity and computational efficiency.

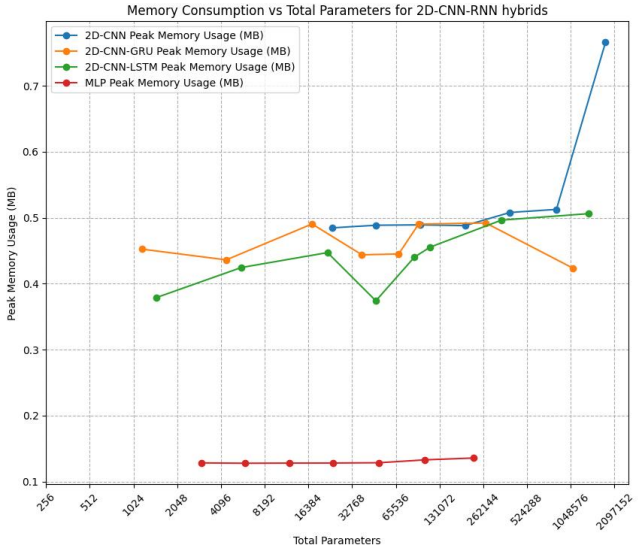

Figure 10: Accuracy Comparison for 2D-CNN-RNN

Figure 10 illustrates the memory consumption patterns for 2D-CNN models and their hybrid variations. It revealed that as the number of parameters increased, memory usage also increased, with the highest consumption observed at larger parameter configurations. The inclusion of recurrent layers, such as GRU and LSTM, in the 2D-CNN-GRU and 2D-CNN-LSTM hybrids led to a significant surge in memory requirements at higher parameter sizes. This demonstrates the memory cost associated with integrating recurrent layers with convolutional architectures, particularly at high parameter counts.

Additionally, Figure 11 observed the memory consumption trends for 1D-CNN models and its 1D-CNN-RNN hybrids. While the pure 1D-CNN model maintained relatively stable memory usage at lower parameter counts, it exhibited an increase, peaking at 0.8MB. In contrast, the hybrid models, 1D-CNN-GRU and 1D-CNN-LSTM, showcased periodic fluctuations in memory usage due to the recurrent layers' need to maintain state information over sequences. These variations highlighted the inherent trade-offs between capturing temporal dynamics and increased memory demands. Although there were spikes in memory usage, they were not consistently high across all parameter settings, indicating potential areas for optimisation. Additionally, the MLP model consistently showed low memory usage across the board, peaking at under 0.2 MB, confirming its suitability for memory-constrained applications due to its streamlined architecture. The results indicate that hybrid models

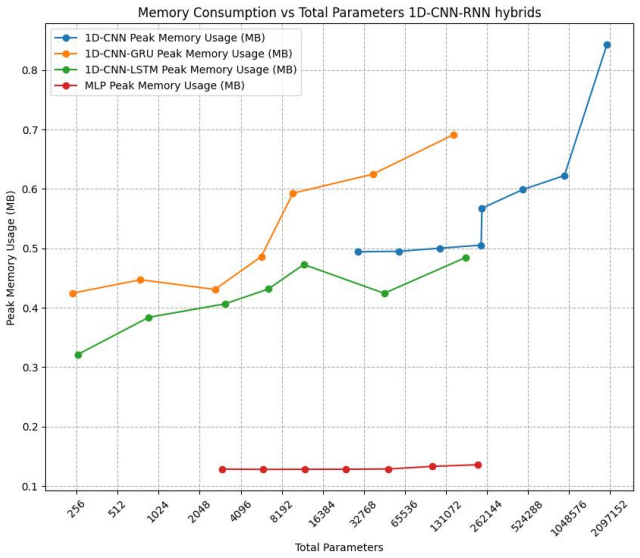

Figure 11: Accuracy Comparison for 1D-CNN-RNN

like 2D-CNN-GRU, 2D-CNN-LSTM, 1D-CNN-GRU, and 1D-CNN-LSTM offer significant memory efficiency. These models deliver competitive accuracy with relatively low parameter counts. Compared to the MLP model, hybrid models like 2D-CNN-GRU provide a clear advantage regarding memory efficiency and accuracy. For instance, the MLP model with 22382 parameters achieves an accuracy of 88.46% and 16670 PPS, while the 2D-CNN-GRU model with 17338 parameters attains a higher accuracy of 89.9%, albeit with a lower PPS of 10,463. This suggests that these models can effectively process data with reduced memory usage, making them well-suited for memory-constrained environments.

Although MLP models generally offer higher PPS due to reduced parameters, maintaining approximately 90% requires more parameters to achieve comparable accuracy. For instance, an MLP model

with 49834 parameters has an accuracy of 85.79% more memory in the process.

In summary, while MLP models provide faster packet processing speeds, the hybrid 2D-CNN-GRU model is more memory efficient, requiring fewer parameters to achieve higher accuracy. This makes the hybrid models better suited for scenarios where memory is limited, and accuracy is a priority. In contrast, MLP models may be preferred when speed is more critical than memory or accuracy.

8 DISCUSSION

The experimental results show that hybrid deep learning models, such as 2D-CNN-GRU, 2D-CNN-LSTM, 1D-CNN-GRU, and 1D-CNN-LSTM, significantly outperform simpler architectures like MLP and CNN in terms of classification accuracy. These models effectively capture both spatial and temporal patterns within network traffic data by leveraging the strengths of both CNNs and RNNs. Specifically, the 1D-CNN-LSTM and 1D-CNN-GRU models achieved the highest accuracy rates of 94.3% and 94.21%, respectively, making them particularly effective for handling encrypted traffic in low-resource environments like community networks. This aligns with Bayat et al. [5], who used a hybrid architecture to achieve 95% accuracy.

However, in addition to packet processing speed (PPS), peak memory usage is a critical factor for models deployed in low-resource environments. Models like 2D-CNN-GRU and 2D-CNN-LSTM achieved high accuracy while maintaining low memory footprints. For example, the 2D-CNN-LSTM model with 22170 parameters, which reached a PPS of 10253.14 and an accuracy of 90.32%, only required 0.447 MB of peak memory. This balance of high accuracy, PPS, and low memory usage makes it ideal for deployment in memory-constrained environments. Similarly, the 2D-CNN-GRU model with 68458 parameters achieved an accuracy of 91.47% and a PPS of 10169.63, with a memory usage of only 0.444 MB, showcasing its ability to maintain efficient performance under tight resource limitations.

On the other hand, simpler models like MLP achieve higher PPS and require significantly less memory (e.g., only 0.128 MB for MLP with 24,140 parameters). However, they only achieve an accuracy of 85.7%. This suggests they may not be as suitable for applications where accuracy is vital. The hybrid models thus offer a better trade-off between accuracy, speed, and memory efficiency, making them more appropriate for resource-constrained environments.

These findings align with Tooke's [25] previous work, which showed that simpler models like a 1D-CNN and 2D-CNN are sufficient for network traffic classification in low-resource communities. However, the 2D-CNN-LSTM and 2D-CNN-GRU demonstrate that the accuracy gain can be achieved without severely compromising performance or memory.

9 CONCLUSION

This study investigated the effectiveness of hybrid deep-learning models for traffic classification in low-resource community networks. By combining Convolutional Neural Networks (CNNs) with Recurrent Neural Networks (RNNs), specifically Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks, the hybrid architectures were able to leverage the strengths of both models in spatial feature extraction and temporal sequence learning. The experimental results demonstrated that hybrid models, particularly 1D-CNN-LSTM and 1D-CNN-GRU, outperformed simpler models regarding classification accuracy, such as Multi-Layer Perceptrons (MLP) and CNNs. The highest accuracies were 94.3% with 1D-CNN-LSTM and 94.21% with 1D-CNN-GRU. However, these accuracy gains came with trade-offs in computational efficiency, particularly in packet processing speed (PPS), which poses challenges for real-time applications in resource-constrained environments.

The key highlights are the performance of the 2D-CNN-LSTM and 2D-CNN-GRU. The 2D-CNN-LSTM model with 22170 parameters only required 0.447 MB of memory while delivering an accuracy of 90.32% and a PPS of 10253.14. Similarly, the 2D-CNN-GRU model, with a memory usage of just 0.444 MB, strikes an excellent balance between accuracy, speed, and memory efficiency. Achieving an accuracy of 91.47% and a PPS of 10169.63 for a total parameter count of 68458 provides a strong balance between accuracy and real-time processing, which is crucial for deployment in community networks.

In summary, this study has shown that hybrid deep learning models, such as 2D-CNN-GRU and 2D-CNN-LSTM, present a promising solution for traffic classification in low-resource community networks. These models offered low memory usage, making them ideal for deployment in environments with limited computational resources. This balance between accuracy, processing speed, and memory efficiency is crucial for resource-constrained environments. However, as the model parameters increase, computational efficiency decreases, indicating a need for further optimisation to scale these models in larger or more complex network setups. Overall, the 2D-CNN-GRU and 2D-CNN-LSTM models effectively balance accuracy, real-time performance, and memory usage, making them strong candidates for improving traffic classification in community networks.

10 LIMITATIONS AND FUTURE WORK

This study highlights several promising directions for future research. An essential step is to evaluate the models' ability to generalise across diverse datasets and network environments by testing them on labelled data from different networks. This will validate the models' resilience and effectiveness in real-world applications. Additionally, expanding the classification scope by introducing more traffic classes and evaluating the models' performance as the complexity of tasks increases will provide valuable insights into their scalability and sustainability. Such an approach will further enhance the precision and dependability of these models in more complex classification scenarios.

Another critical area for exploration is the integration of lightweight models like MobileNet and SqueezeNet, which are tailored for resource-constrained environments. These models offer a more efficient alternative, balancing accuracy with reduced computational and memory demands, making them well-suited for community networks with limited resources. Deploying these models in real-time network environments and simulating practical conditions would provide insights into their operational performance. Additionally, expanding research across various network types,

such as urban, IoT, and cross-regional networks, will test the models' generalisation capabilities, ensuring they can adapt to diverse infrastructures and traffic patterns, ultimately advancing network management and digital inclusion.

11 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. Without revealing specific details, we will categorise traffic into types (e.g., video streaming and social media). To protect the Ocean View community's reputation [28].

REFERENCES

[1] Aceto, G., Ciuonzo, D., Montieri, A., and Pescapé, A. 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] 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.

[3] Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M., and Farhan, L. Review of deep learning: concepts, cnn architectures, challenges, applications, future directions. *Journal of big Data 8* (2021), 1–74.

[4] 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).

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

[6] Bujlow, T., Carela-Español, V., and Barlet-Ros, P. Independent comparison of popular dpi tools for traffic classification. *Computer Networks 76* (2015), 75–89.

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

[8] Dang, C. N., Moreno-García, M. N., and De la Prieta, F. Hybrid deep learning models for sentiment analysis. *Complexity 2021*, 1 (2021), 9986920.

[9] Deri, L., Martinelli, M., Bujlow, T., and Cardigliano, A. ndpi: Open-source high-speed deep packet inspection. In *2014 International Wireless Communications and Mobile Computing Conference (IWCMC)* (2014), IEEE, pp. 617–622.

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

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

[12] Han, D., Yun, S., Heo, B., and Yoo, Y. Rethinking channel dimensions for efficient model design. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition* (2021), pp. 732–741.

[13] Jiang, C., Xu, S., Geng, G., Weng, J., and Zhang, X. Seq2path: a sequence-to-path-based flow feature fusion approach for encrypted traffic classification. *Cluster Computing 26*, 3 (2023), 1785–1800.

[14] Lim, H.-K., Kim, J.-B., Heo, J.-S., Kim, K., Hong, Y.-G., and Han, Y.-H. Packet-based network traffic classification using deep learning. In *2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)* (2019), IEEE, pp. 046–051.

[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] 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.

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

[20] Recommendation, I. E. 800, definitions of terms related to quality of service. *International Telecommunication Union's Telecommunication Standardization Sector (ITU-T) Std* (2008).

[21] Rezaei, S., and Liu, X. Deep learning for encrypted traffic classification: An overview. *IEEE Communications Magazine 57*, 5 (2019), 76–81.

[22] 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.

[23] Sherry, J., Lan, C., Popa, R. A., and Ratnasamy, S. Blindbox: Deep packet inspection over encrypted traffic. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication* (2015), pp. 213–226.

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

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

[26] 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.

[27] Wang, W., Zhu, M., Wang, J., Zeng, X., and Yang, Z. End-to-end encrypted traffic classification with one-dimensional convolution neural networks. In *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)* (2017), IEEE, pp. 43–48.

[28] 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.

[29] 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.