Hybrid Deep Learning for Traffic Classification in Low-Resource Networks

Michael Gamsu and Josiah Chavula

Department of Computer Science
University of Cape Town, South Africa

Abstract. Accurate network traffic classification is essential for enhancing Quality of Service (QoS) and optimizing traffic engineering. However, traditional classification techniques face significant challenges in effectively handling encrypted traffic. Moreover, many machine learning approaches are either too computationally intensive or insufficiently generalized to be practical for community networks with limited resources. To address these constraints, this study investigates hybrid deep learning architectures that combine Convolutional Neural Networks (CNNs) with Recurrent Neural Networks (RNNs)—specifically Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) models. The evaluated models include 2D-CNN-LSTM, 2D-CNN-GRU, 1D-CNN-LSTM, and 1D-CNN-GRU, in addition to baseline models such as multilayer perceptrons (MLP), 2D-CNN, and 1D-CNN. Preliminary results indicate that hybrid models, particularly 1D-CNN-LSTM and 1D-CNN-GRU, achieve the highest classification precision. However, these models incur considerable computational overhead, limiting their applicability in resource-constrained environments. Conversely, 2D-CNN-based hybrids demonstrate a promising trade-off, offering competitive performance with lower computational demands, making them more suitable for deployment in community networks.

Keywords: Deep Learning · Community Networks · Traffic Classification · Neural Networks.

1 Introduction

Network traffic classification involves categorising network traffic based on the associated application (e.g., Gmail, Facebook) or protocol type (e.g., TCP, UDP). It plays a vital role in network engineering and management, supporting security enforcement, and enabling application-aware traffic prioritisation - and thus enhancing Quality of Service (QoS) [2]. Real-time classification can significantly improve QoS and user experience by prioritising critical traffic flows, while also offering insights into network behaviour that aid in resource allocation, anomaly detection, and security monitoring.

Community networks, which provide affordable connectivity solutions to underserved and remote populations, stand to benefit greatly from effective traffic

classification [16]. By enabling prioritisation of essential services, early anomaly detection, and efficient bandwidth utilisation, traffic classification can help these networks deliver more reliable service, improve user satisfaction, and support sustainable growth in resource-constrained environments [10].

Traditional traffic classification techniques include deep packet inspection (DPI), port-based methods, flow-based classification, and packet-level analysis [27]. DPI, which relies on inspecting packet headers and application signatures, poses both privacy concerns and computational challenges—particularly for low-resource networks [15]. Port-based methods depend on identifying TCP/UDP port numbers but are often defeated by port obfuscation techniques [28]. Flow-based classification aggregates packets into flows based on identifiers such as IP addresses and port numbers [4], yet this method suffers from high latency, limited scalability, and poor handling of encrypted or short-lived flows [20].

In contrast, packet-level classification offers a finer-grained and more adaptable solution by examining features such as protocol type, payload content, and header information [15]. This method has proven particularly effective in encrypted traffic scenarios and has been successfully applied to malware detection, as demonstrated by studies such as Wang et al. [25] and Lim et al. [13]. These advances underscore packet-level classification as a promising direction for secure and efficient traffic analysis.

With the advancement of deep learning, new opportunities have emerged for building accurate, generalisable, and low-cost classifiers, especially for encrypted traffic [27]. Nevertheless, there remains a gap in the literature concerning the comparative evaluation of classification accuracy and computational performance in real-time settings, particularly within the context of resource-constrained community networks. Previous work by Tooke, Weisz, and Dicks [24,27,10] has laid important groundwork by assessing traditional machine learning and deep learning methods for such environments, but further exploration of hybrid and lightweight models is needed to optimise performance under constrained conditions.

1.1 Research Questions

This study evaluates and compares the performance of hybrid deep learning architectures—specifically, 1D-CNN-LSTM, 1D-CNN-GRU, 2D-CNN-LSTM, and 2D-CNN-GRU—against simpler stand-alone models such as multilayer perceptrons (MLP), 2D-CNN, and 1D-CNN. The goal is to identify models that balance high classification accuracy with computational efficiency, making them suitable for real-time traffic classification in resource-constrained community networks. In particular, the evaluation focuses on accuracy, packet processing speed, and memory efficiency. The following research questions guide this investigation:

1. **Classification Accuracy:** To what extent do hybrid deep learning architectures—combining Convolutional Neural Networks (CNNs) with Recurrent Neural Networks (RNNs)—improve classification accuracy over simpler models such as MLP and standalone CNNs?

2. **Packet Processing Speed:** Are hybrid architectures suitable for real-time network traffic classification? How do they compare to simpler models in terms of packet processing speed, a critical metric for real-time applications?
3. **Memory Efficiency:** How do various deep learning architectures affect memory consumption as model complexity increases, and what trade-offs emerge between classification performance and computational resource requirements?

1.2 Contributions

This study makes the following key contributions:

- A comprehensive evaluation of hybrid deep learning models (1D-CNN-LSTM, 1D-CNN-GRU, 2D-CNN-LSTM, 2D-CNN-GRU) against simpler MLP and CNN architectures, specifically within the context of resource-constrained community networks.
- Empirical evidence linking model architecture complexity to classification accuracy, packet processing speed, and memory efficiency, highlighting trade-offs critical for real-time traffic classification.
- Demonstration of the feasibility and limitations of hybrid models in constrained environments, guiding future model design for community network deployments.
- Provision of a reproducible preprocessing and experimental framework tailored for encrypted and packet-level traffic classification in real-world community network datasets.

2 Background

2.1 Quality of Service (QoS) in Community Networks

Community networks aim to bridge the digital divide by providing affordable internet connectivity in low-resource and rural areas. These networks are developed, owned, and maintained by the communities they serve, relying on decentralised and participatory models. They are grounded in principles of openness, neutrality, and freedom, which make them a compelling solution for extending Internet access to underserved regions [16].

Despite their promise, community networks face significant challenges related to administration, scalability, service quality, and the cost of deployment and maintenance. Limited financial and technical resources make it essential to optimise performance without incurring prohibitive infrastructure expenses.

Quality of Service (QoS) is a critical consideration in network performance, encompassing metrics such as latency, jitter, bandwidth, and packet loss. These metrics directly impact the reliability and usability of network services [19]. Complementing QoS, Quality of Experience (QoE) focuses on the user's perceived satisfaction with a service. It takes into account both technical performance and subjective experience, emphasising end-user content accessibility and usability [17].

2.2 Traffic Classification in Community Networks

Traffic classification plays a crucial role in managing QoS by enabling networks to prioritise latency-sensitive or critical applications. This is especially important in community networks, where resources such as bandwidth and computational capacity are limited. By accurately classifying traffic, network administrators can enhance efficiency, reduce operational costs, and delay the need for infrastructure upgrades [15,4,23].

In addition to performance benefits, traffic classification contributes to network security by identifying anomalies, detecting malicious activity, and applying appropriate policies to different traffic types. It also facilitates long-term strategic planning by offering insights into usage patterns, which improves network reliability and user satisfaction [17,14].

However, the deployment of modern traffic classification techniques—especially those powered by deep learning—can be challenging in community networks due to the computational and memory resources they require. This necessitates the exploration of lightweight yet accurate models that are practical for such environments.

2.3 Deep Learning Neural Network Architectures

Deep learning, a subset of machine learning, employs multi-layered neural networks to automatically learn complex patterns from data. Unlike traditional machine learning models that often rely on manual feature engineering, deep learning models can extract hierarchical representations directly from raw inputs. They have been widely applied in domains such as image recognition, natural language processing, and, increasingly, network traffic classification.

Common deep learning architectures include Multilayer Perceptrons (MLPs), Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and hybrid models combining multiple architectures for improved performance.

Multilayer Perceptrons (MLPs): MLPs are feedforward neural networks composed of an input layer, one or more hidden layers, and an output layer. Each neuron in a layer is connected to every neuron in the subsequent layer, and the output of each neuron is computed by applying a weighted sum followed by an activation function. MLPs are trained using backpropagation and are effective for both regression and classification tasks [18]. While foundational, MLPs are limited in their ability to capture spatial or temporal patterns in data.

Convolutional Neural Networks (CNNs): CNNs are designed to process data with grid-like topology, such as images or packet byte sequences. They use convolutional layers to detect local patterns via filters (kernels), and pooling layers to reduce dimensionality and control overfitting [15,3]. Due to weight sharing and sparse connections, CNNs are computationally more efficient than fully connected networks [11].

In network traffic classification, 1D-CNNs are particularly effective, as they can model sequential data such as packet byte streams. The convolution operation captures spatial relationships between adjacent bytes, making them well-suited for encrypted or obfuscated traffic classification [22]. Studies by Aceto et al. [1] and Wang et al. [26] have shown that 1D-CNNs outperform 2D-CNNs in capturing temporal dependencies in sequential network data.

Recurrent Neural Networks (RNNs): RNNs are designed for sequential data and leverage internal memory to capture temporal dependencies. They process input sequences by maintaining a hidden state that evolves over time. However, traditional RNNs suffer from the vanishing gradient problem, which limits their ability to model long-term dependencies.

To overcome this, advanced RNN variants such as Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs) were developed. Both architectures introduce gating mechanisms to regulate the flow of information and retain long-term dependencies more effectively. These characteristics make LSTM and GRU models particularly suitable for flow-based traffic classification and temporal pattern analysis [20].

Hybrid Deep Learning Models: Hybrid models aim to combine the strengths of different neural network architectures. For example, CNN layers are effective at feature extraction, while RNN layers are adept at capturing temporal dependencies. Integrating these two components allows hybrid models to achieve higher accuracy in complex classification tasks. Empirical studies, such as by Dang et al. [8], have demonstrated that CNN-RNN hybrids outperform stand-alone models in traffic classification, particularly in terms of accuracy and generalisation capability.

3 Experiment Design

This section outlines the design and execution of the preprocessing pipeline used to convert raw PCAP files, collected from a community network, into a structured format suitable for training traffic classification models.

The preprocessing stage produces training, testing, and validation datasets by transforming unstructured packet data. This involves extracting packet-level labels and converting packets into feature vectors used as inputs to machine learning models.

3.1 Packet Labelling

Our study performs traffic classification by application, with examples of application labels, including *Facebook* and *YouTube*. A widely adopted method in the literature for assigning these labels - especially when they are not recorded during data collection, as is the case in our dataset - is the use of deep packet inspection (DPI) tools. These tools leverage databases of application signatures

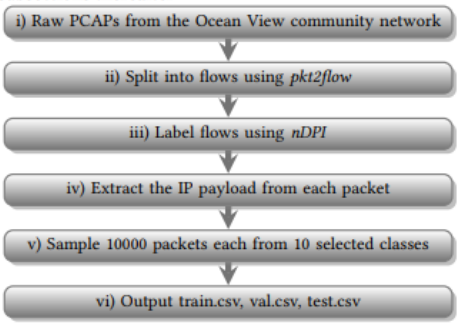

Fig. 1. Preprocessing structure [27]

to identify traffic classes from raw network traces[9]. The packet flows were labelled using the *nDPI* library [9], an open source deep packet inspection tool shown to be more accurate than alternative labelling methods [6]. This labelling approach was also followed in a previous traffic classification study by Tooke, Weisz and Dicks [24,27,10], which used nDPI to label their network dataset, including encrypted traffic, further validating the effectiveness of the tool. The pre-processing scripts and base models are available on github[1].

To streamline the labelling process, we first use the open-source package *pkt2flow* to segment the packets in the original PCAP files into individual flows, with each flow saved as a separate PCAP file. These flows were then processed with nDPI, which assigns an application label to each flow. Subsequently, every packet within a flow inherits the same label. The result of this preprocessing step is a CSV file that lists, for each flow derived from the raw PCAPs, the corresponding PCAP file name and the associated application label applied uniformly to all packets in that flow.

To address class imbalance in our dataset, we apply under-sampling, where classes containing more packets than needed are reduced by randomly selecting an equal number of packets from each class. Specifically, we extracted 9,000 packets from each of 10 selected application classes, ensuring a balanced data set for training and evaluation. The selected classes are presented in Table 1.

The decision to use 10,000 packets per class is informed by the typical dataset sizes used in related work. The specific application classes were selected based on two main criteria: (i) popularity within the community network, prioritising applications most relevant and useful to the network's user base, and (ii) ensuring a diverse representation across different application categories. This diversity is intended to evaluate the performance of the classifier across a wide spectrum of traffic types and application behaviours.

[1] https://github.com/chavula/DLOGs

Table 1. Application labels in the dataset

Application Label	Application Type
YouTube	Video
Facebook	Social Media
GoogleServices	Phone Background Services
Instagram	Images
WhatsApp	Instant Messaging
BitTorrent	Torrent Files
TeamViewer	Remote Desktop
Gmail	Email
WindowsUpdate	Desktop OS Updates
PlayStore	Mobile App Store

3.2 Extracting IP Payloads

The process to extract packet data started with HTTPS traffic being identified using the SSL filter in Wireshark [21]. To handle the bidirectional nature of network communication, the *pkt2flow* utility was used to split flows by direction [7]. A script from Bayat et al. [5] was used to generate a 4-tuple (source IP, destination IP, source port, and destination port) for each TCP connection. To standardise packet structure, UDP headers were zero-padded to match the length of TCP headers, following Lotfollahi et al. [15].

3.3 Feature Extraction and Transformation

Raw packet payload data was extracted and converted into byte format for model input, a method shown to improve classification performance, especially for encrypted traffic [15,25]. IP addresses were masked in headers to prevent over-fitting. The *scapy* library was used to extract payloads and convert them into byte streams, suitable for input into 1D models such as MLPs, 1D-CNNs, and LSTMs [25].

Packets in the dataset vary in length, which is incompatible with the fixed-size input requirements of neural networks. Based on the observation that most packets are smaller than 1480 bytes [15], the IP header and the first 1480 bytes of each packet were retained, resulting in a 1500-byte input vector. Shorter packets were padded with zeros.

3.4 Train, Test, and Validation Splits

The dataset consists of feature vectors - the 1480 bytes from each packet, and associated application labels as indicated in Table1. The dataset was split into training (64%), validation (16%), and testing (20%) subsets. To ensure balance, only application classes with at least 9,000 packets were considered. Ten classes meeting this threshold were selected, and each class was capped at 9,000 packets.

3.5 Reshaping for CNN Models

CNN-based models require specific input shapes. For 1D-CNNs, a channel dimension was added to form a 3D input array, capturing packet intervals [12]. For 2D-CNNs, packet bytes were reshaped into grid structures, where pixel intensity corresponds to feature magnitude [27].

3.6 Hyperparameter Tuning

Manual tuning was employed to balance model accuracy, training speed, and memory efficiency. Although grid search is common, it was avoided due to its computational expense. Learning rates of 0.00005, 0.0001, 0.0005, and 0.001 were tested; 0.001 was selected as it achieved faster convergence without reducing accuracy.

Dropout rates of 0.1, 0.2, and 0.3 were evaluated to prevent over-fitting; optimal performance was found using 0.2–0.3. L2 regularisation with values of 0.001, 0.01, and 0.1 was also tested and improved generalisation.

Architectural enhancements like Batch Normalization, Flatten, and Max-Pooling layers were incorporated. Kernel sizes from 3 to 5 were assessed; a size of 3 was selected to reduce complexity. Filter sizes ranged from 4 to 128, and sizes exceeding 128 were deemed too resource-intensive for low-resource environments.

4 Experimental Methodology

This section presents the methodology for evaluating hybrid deep learning models against simpler models in the context of network traffic classification. The experiments aim to measure the trade-off between model complexity, computational efficiency, and classification performance.

The experiments used the datasets generated during the preprocessing phase (Section 3) and addressed the research objectives outlined in Section 1.1. Each experiment was conducted ten times to ensure statistical robustness. Evaluation metrics included classification accuracy, packet processing speed (PPS), and memory usage. Filter sizes were adjusted to study their effect on model performance and complexity.

Deep learning models were implemented in Python using the Keras Sequential API, with a TensorFlow 2.0 backend. Keras was selected as the deep learning framework due to its user-friendly interface and modular design, which facilitates rapid model prototyping without compromising flexibility.

To simulate deployment in a low-resource environment, all model testing and evaluation were conducted on a single-core Intel Xeon 2.50 GHz processor with 3.75 GB of RAM. Model training, however, was accelerated using GPU resources provided by Google Colaboratory.

4.1 Experiment 1: Classification Accuracy

The goal of this experiment was to evaluate the classification accuracy of hybrid models (e.g., CNN-RNN combinations) compared to simpler models like MLPs or stand-alone CNNs. Accuracy is defined as:

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

The hybrid models are expected to outperform simpler architectures by leveraging the spatial feature extraction of CNNs and temporal sequence modelling of RNNs. Although the F1 score is often used in imbalanced datasets, it was not employed here due to the balanced nature of the dataset.

4.2 Experiment 2: Packet Processing Speed (PPS)

This experiment measures the inference speed of models, quantified as packets processed per second (PPS). The goal is to determine the feasibility of using hybrid models for real-time classification, especially in resource-constrained environments.

The community network under study has an internet link capacity of 10 Mbps. In our dataset, the average packet size, including the IP header, is approximately 992.57 bytes. Based on this, we estimate that the network handles roughly 10,000 packets per second on average. This figure serves as a reference point for evaluating the prediction throughput of each classification model — measured in packets per second — as an indicator of the model's suitability for real-time traffic classification.

Each model was tested on the test dataset ten times, and the average PPS was recorded. A baseline threshold of 10,000 PPS was set to indicate real-time viability. Simpler models like MLPs are expected to achieve higher PPS due to lower computational requirements, whereas hybrid models may fall below this threshold.

4.3 Experiment 3: Memory Usage and Optimisation

This experiment investigates how the number of parameters affects memory consumption and model performance. Memory usage directly correlates with the total number of parameters, with each 32-bit float occupying 4 bytes. Reducing the parameter count improves inference speed and memory efficiency but may hinder accuracy.

By incrementally increasing the number of filters in the CNN layers, the experiment evaluates the impact on memory consumption and classification accuracy. Results are expected to highlight the trade-off between model complexity and performance, informing the development of lightweight models for deployment in constrained environments.

5 Results

This section presents the results of experiments conducted to evaluate the performance of various deep learning models, including MLP, 2D-CNN, 1D-CNN, 2D-CNN-GRU, 2D-CNN-LSTM, 1D-CNN-GRU, and 1D-CNN-LSTM. The models were assessed based on accuracy, packets per second (PPS), and memory consumption across different parameter configurations.

5.1 Accuracy Results

The initial experiment evaluated the classification accuracy of both standard and hybrid deep learning models. Hybrid models—particularly 1D-CNN-LSTM and 1D-CNN-GRU—consistently outperformed simpler architectures, demonstrating the effectiveness of combining CNNs with RNNs for traffic classification tasks.

Figures 2 and 3 show a clear trend: as model complexity increases, so does accuracy—though with diminishing returns beyond a certain point. This suggests that more complex models are better at capturing intricate patterns, but eventually reach an accuracy plateau.

Specifically, the 2D-CNN-LSTM and 2D-CNN-GRU achieved peak accuracies of 94.07% and 92.14%, respectively, outperforming the stand-alone 2D-CNN (91.24%) and MLP (88.46%), as shown in Figure 2. These results underscore the advantage of hybrid architectures in capturing both spatial and temporal characteristics of network traffic.

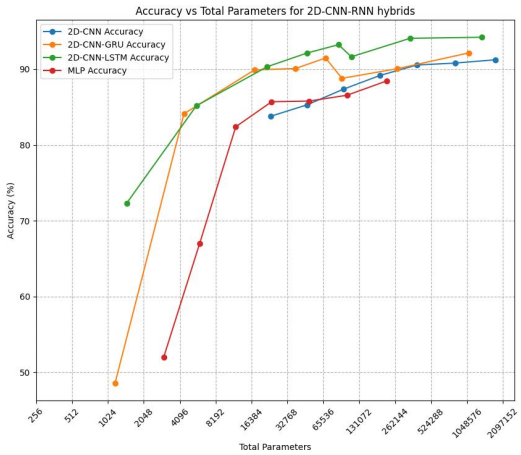

Fig. 2. Comparison of Model Accuracy Across Different Architectures and Parameter Sizes for 2D-CNN-RNN Hybrids

Similarly, Figure 3 shows that 1D-CNN-LSTM and 1D-CNN-GRU achieved the highest accuracies of 94.3% and 94.21%, respectively, slightly ahead of the

1D-CNN (92.64%). These results confirm the effectiveness of hybrid models, while also highlighting the practicality of simpler architectures like 1D-CNN for low-resource scenarios where computational efficiency is crucial.

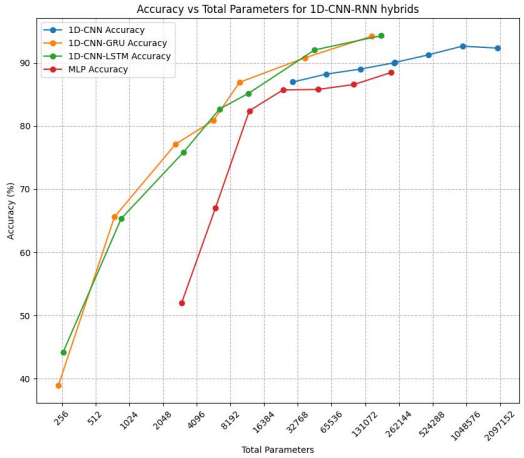

Fig. 3. Comparison of Model Accuracy Across Different Architectures and Parameter Sizes for 1D-CNN-RNN Hybrids

5.2 Packets Per Second (PPS) Results

The second experiment evaluated model throughput in terms of packets per second (PPS), a key metric for assessing real-time processing capability, particularly in resource-constrained environments.

Figure 4 reveals that the MLP model achieved the highest throughput, maintaining a consistent 16,000 PPS across all parameter configurations. This performance affirms MLP's computational efficiency, making it ideal for real-time applications with limited resources.

In contrast, while the 2D-CNN maintained between 10,000 and 15,000 PPS for most configurations, the 2D-CNN-GRU and 2D-CNN-LSTM hybrids showed a decline in PPS as their complexity increased, eventually dropping below the 10,000 PPS threshold.

Figure 5 shows a similar trend for 1D models. The 1D-CNN experienced a sharp decline in PPS with increasing parameters. The 1D-CNN-GRU and 1D-CNN-LSTM exhibited relatively stable performance (2,000–4,000 PPS), with GRU slightly outperforming LSTM at similar complexity. For instance, 1D-CNN-GRU achieved 2,211.08 PPS at 150,154 parameters, suggesting that GRU-based models are more computationally efficient than their LSTM counterparts.

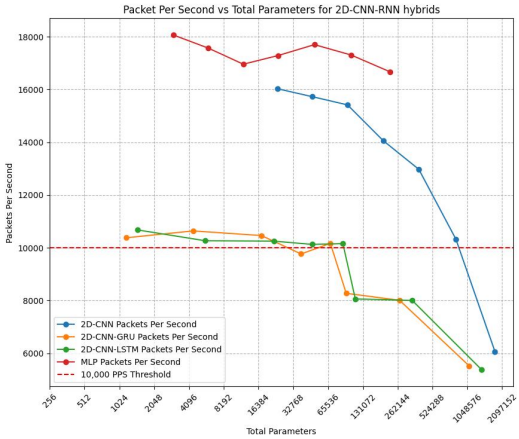

Fig. 4. Comparison of Model Performance in Terms of Packets Per Second for 2D-CNN-RNN Hybrids Across Various Parameter Sizes

These results highlight a trade-off between accuracy and throughput. While MLP provides superior speed, hybrid models offer better accuracy at the cost of lower PPS.

5.3 Memory Optimization Results

The final experiment analyzed memory usage across different architectures to understand how model complexity impacts resource consumption.

Figure 6 shows that memory usage increased proportionally with the number of parameters. Hybrid models with recurrent layers—2D-CNN-GRU and 2D-CNN-LSTM—consumed significantly more memory at higher parameter counts, reflecting the cost of managing sequential data through RNN components.

Similarly, Figure 7 shows that while 1D-CNN maintained relatively stable memory usage (peaking at 0.8 MB), its hybrid variants exhibited fluctuations. These were due to the recurrent layers' state retention requirements. However, the spikes were not consistent, indicating opportunities for optimization. Notably, the MLP consumed the least memory throughout—remaining under 0.2 MB—highlighting its efficiency for memory-constrained applications.

Overall, the results suggest that hybrid models such as 2D-CNN-GRU, 2D-CNN-LSTM, 1D-CNN-GRU, and 1D-CNN-LSTM offer favourable memory efficiency while delivering high accuracy at relatively lower parameter counts. For example, the MLP model with 22,382 parameters reached 88.46% accuracy and 16,670 PPS, while the 2D-CNN-GRU, with only 17,338 parameters, achieved 89.9% accuracy — although at a lower PPS of 10,463. This demonstrates that hybrid models can offer strong performance while using less memory, making them suitable for constrained environments where accuracy is a priority.

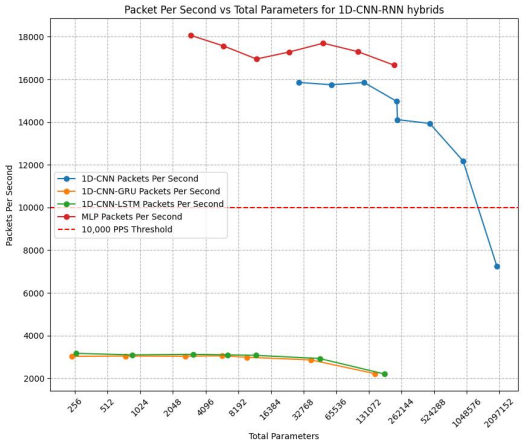

Fig. 5. Performance Comparison Across Different 1D-CNN-RNN Hybrid Architectures as a Function of Parameter Size

In contrast, MLP models are better suited for scenarios where speed and simplicity outweigh accuracy. While they offer high PPS and low memory usage, achieving comparable accuracy requires significantly more parameters, as seen with the 49,834-parameter MLP model reaching only 85.79% accuracy.

6 Discussion

The experimental results demonstrate that hybrid deep learning models—specifically 2D-CNN-GRU, 2D-CNN-LSTM, 1D-CNN-GRU, and 1D-CNN-LSTM—significantly outperform simpler architectures such as MLP and stand-alone CNNs in terms of classification accuracy. By combining the spatial feature extraction capabilities of CNNs with the temporal sequence modeling strength of RNNs, these hybrid models effectively capture the complex patterns inherent in encrypted network traffic.

Among the evaluated models, 1D-CNN-LSTM and 1D-CNN-GRU achieved the highest classification accuracies of 94.3% and 94.21%, respectively. This reinforces their suitability for deployment in low-resource environments like community networks, where the detection of encrypted or obfuscated traffic is a critical challenge. These findings are consistent with Bayat et al. [5], who employed a similar hybrid architecture and reported a classification accuracy of approximately 95%.

Beyond accuracy, real-world deployment requires a careful balance between classification performance and resource efficiency—particularly packet processing speed (PPS) and peak memory usage. In this regard, models such as 2D-CNN-GRU and 2D-CNN-LSTM offer a compelling trade-off. For instance, the 2D-CNN-LSTM model with 22,170 parameters achieved a PPS of 10,253.14 and

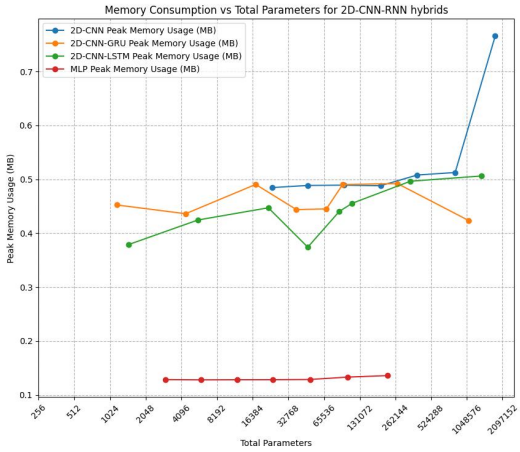

Fig. 6. Memory Consumption for 2D-CNN Models and Hybrid Variations

an accuracy of 90.32%, while consuming only 0.447 MB of memory. Similarly, the 2D-CNN-GRU with 68,458 parameters reached an accuracy of 91.47%, a PPS of 10,169.63, and a memory usage of just 0.444 MB. These results highlight the potential of hybrid models to deliver high performance with modest computational overhead, making them well-suited for deployment in constrained environments.

In contrast, while simpler models like MLP excel in throughput and memory efficiency, they lag behind in accuracy. For example, an MLP model with 24,140 parameters required only 0.128 MB of memory and achieved a high PPS, but its classification accuracy plateaued at 85.7%. This makes MLP less suitable for scenarios where accurate classification of encrypted or evasive traffic is a priority.

These observations echo the work of Tooke [24], who demonstrated that models like 1D-CNN and 2D-CNN can perform sufficiently in low-resource settings. However, the current findings suggest that hybrid models like 2D-CNN-LSTM and 2D-CNN-GRU can push accuracy boundaries further without drastically increasing computational or memory requirements. Therefore, while simpler models may be favoured in extremely constrained settings, hybrid models offer a scalable and practical solution when accuracy is mission-critical.

7 Conclusion

This study explored the viability of hybrid deep learning architectures for encrypted traffic classification in low-resource community network environments. By integrating Convolutional Neural Networks (CNNs) with Recurrent Neural Networks (RNNs), specifically Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks, the proposed hybrid models capitalize on the strengths of both spatial and temporal pattern recognition.

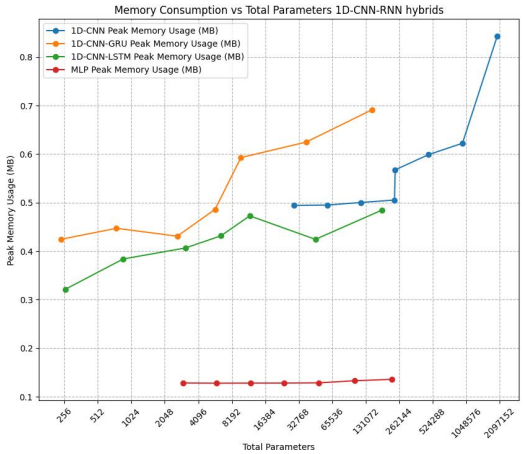

Fig. 7. Memory Consumption for 1D-CNN Models and Hybrids

The experimental results confirm that hybrid models—particularly 1D-CNN-LSTM and 1D-CNN-GRU—outperform traditional architectures such as MLP and stand-alone CNNs in terms of classification accuracy. The top-performing models achieved accuracies of 94.3% and 94.21%, respectively. However, these gains in accuracy were accompanied by increased computational demands, especially in terms of PPS, highlighting a trade-off between accuracy and efficiency in real-time applications.

Significantly, the 2D-CNN-LSTM model with 22,170 parameters achieved an accuracy of 90.32%, a PPS of 10,253.14, and required only 0.447 MB of memory. Similarly, the 2D-CNN-GRU model with 68,458 parameters achieved an accuracy of 91.47%, a PPS of 10,169.63, and used just 0.444 MB of memory. These results suggest that hybrid models not only enhance classification accuracy but also maintain reasonable performance and memory footprints—making them highly suitable for real-world deployment in resource-constrained community networks.

In summary, the research findings underscore the potential of hybrid deep learning models such as 2D-CNN-GRU and 2D-CNN-LSTM to serve as effective solutions for encrypted traffic classification in low-resource environments. Their ability to balance accuracy, throughput, and memory efficiency makes them compelling candidates for practical implementation. Nevertheless, as model complexity grows, computational efficiency may decline. This indicates a need for future research into optimization techniques—such as model pruning, quantization, or edge-accelerated inference—to ensure scalable and efficient deployment in larger or more complex network settings.

7.1 Future Work

Future research should explore strategies to further optimize these hybrid models for deployment on edge devices with even more constrained computational resources. Techniques such as model pruning, quantization, and knowledge distillation could reduce the memory footprint and inference latency without sacrificing significant accuracy. Additionally, investigating adaptive or dynamic model architectures that can adjust their complexity based on available resources and traffic conditions may enhance practicality in diverse environments.

Another promising direction is the integration of unsupervised or semi-supervised learning approaches to reduce the reliance on large labelled datasets, which are often difficult to obtain in community network settings. Expanding the evaluation to encompass a wider range of traffic types, including emerging protocols and attack scenarios, would provide deeper insights into model robustness.

Finally, real-world deployment and longitudinal studies within actual community networks would validate the effectiveness of these models in operational conditions, accounting for evolving network behaviours and hardware constraints.

References

1. Aceto, G., Ciuonzo, D., Montieri, A., Pescapé, A.: Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges. IEEE Transactions on Network and Service Management **16**(2), 445–458 (2019)
2. Adedayo, A.O., Twala, B.: Qos functionality in software defined network. In: 2017 International Conference on Information and Communication Technology Convergence (ICTC). pp. 693–699. IEEE (2017)
3. Alzubaidi, L., Zhang, J., Humaidi, A.J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M.A., Al-Amidie, M., Farhan, L.: Review of deep learning: concepts, cnn architectures, challenges, applications, future directions. Journal of big Data **8**, 1–74 (2021)
4. Azab, A., Khasawneh, M., Alrabaee, S., Choo, K.K.R., Sarsour, M.: Network traffic classification: Techniques, datasets, and challenges. Digital Communications and Networks (2022)
5. Bayat, N., Jackson, W., Liu, D.: Deep learning for network traffic classification. arXiv preprint arXiv:2106.12693 (2021)
6. Bujlow, T., Carela-Español, V., Barlet-Ros, P.: Independent comparison of popular dpi tools for traffic classification. Computer Networks **76**, 75–89 (2015)
7. caesar0301: pkt2flow. `https://github.com/caesar0301/pkt2flow` (2014)
8. Dang, C.N., Moreno-García, M.N., De la Prieta, F.: Hybrid deep learning models for sentiment analysis. Complexity **2021**(1), 9986920 (2021)
9. Deri, L., Martinelli, M., Bujlow, T., Cardigliano, A.: ndpi: Open-source high-speed deep packet inspection. In: 2014 International Wireless Communications and Mobile Computing Conference (IWCMC). pp. 617–622. IEEE (2014)
10. Dicks, M., Chavula, J.: Deep learning traffic classification in resource-constrained community networks. In: 2021 IEEE AFRICON. pp. 1–7 (2021). `https://doi.org/10.1109/AFRICON51333.2021.9570875`
11. Goodfellow, I., Bengio, Y., Courville, A.: Deep learning. MIT press (2016)

12. Han, D., Yun, S., Heo, B., Yoo, Y.: Rethinking channel dimensions for efficient model design. In: Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition. pp. 732–741 (2021)
13. Lim, H.K., Kim, J.B., Heo, J.S., Kim, K., Hong, Y.G., Han, Y.H.: Packet-based network traffic classification using deep learning. In: 2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC). pp. 046–051. IEEE (2019)
14. Lopez-Martin, M., Carro, B., Sanchez-Esguevillas, A., Lloret, J.: Network traffic classifier with convolutional and recurrent neural networks for internet of things. IEEE access **5**, 18042–18050 (2017)
15. Lotfollahi, M., Jafari Siavoshani, M., Shirali Hossein Zade, R., Saberian, M.: Deep packet: A novel approach for encrypted traffic classification using deep learning. Soft Computing **24**(3), 1999–2012 (2020)
16. Martínez-Cervantes, L., Guevara-Martínez, R.: Community networks and the quest for quality
17. Micholia, P., Karaliopoulos, M., Koutsopoulos, I., Navarro, L., Vias, R.B., Boucas, D., Michalis, M., Antoniadis, P.: Community networks and sustainability: a survey of perceptions, practices, and proposed solutions. IEEE Communications Surveys & Tutorials **20**(4), 3581–3606 (2018)
18. Pinkus, A.: Approximation theory of the mlp model in neural networks. Acta numerica **8**, 143–195 (1999)
19. Recommendation, I.: E. 800, definitions of terms related to quality of service. International Telecommunication Union's Telecommunication Standardization Sector (ITU-T) Std (2008)
20. Rezaei, S., Liu, X.: Deep learning for encrypted traffic classification: An overview. IEEE Communications Magazine **57**(5), 76–81 (2019)
21. Shbair, W.M., Cholez, T., Francois, J., Chrisment, I.: A multi-level framework to identify https services. In: NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium. pp. 240–248. IEEE (2016)
22. Sherry, J., Lan, C., Popa, R.A., Ratnasamy, S.: Blindbox: Deep packet inspection over encrypted traffic. In: Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication. pp. 213–226 (2015)
23. Thompson, N.C., Greenewald, K., Lee, K., Manso, G.F.: The computational limits of deep learning. arXiv preprint arXiv:2007.05558 (2020)
24. Tooke, J., Chavula, J.: Resource-constrained real-time network traffic classification using one-dimensional convolutional neural networks. In: International Conference on e-Infrastructure and e-Services for Developing Countries. pp. 107–127. Springer (2021)
25. Wang, P., Ye, F., Chen, X., Qian, Y.: Datanet: Deep learning based encrypted network traffic classification in sdn home gateway. IEEE Access **6**, 55380–55391 (2018)
26. Wang, W., Zhu, M., Wang, J., Zeng, X., 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). pp. 43–48. IEEE (2017)
27. Weisz, S., Chavula, J.: Community network traffic classification using two-dimensional convolutional neural networks. In: International Conference on e-Infrastructure and e-Services for Developing Countries. pp. 128–148. Springer (2021)
28. Zhang, J., Chen, X., Xiang, Y., Zhou, W., Wu, J.: Robust network traffic classification. IEEE/ACM transactions on networking **23**(4), 1257–1270 (2014)