Review of Deep Learning Approaches to Network Traffic Classification for Community Networks

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

ABSTRACT

Traffic classification is an important component of traffic engineering and quality of service in community networks. In recent years, it has become apparent that deep learning techniques are effective for such classification tasks, especially since classical approaches struggle to deal with encrypted traffic. In this paper, we review various approaches that have been taken by researchers in the traffic classification literature, including the constraints that have been considered, the deep learning techniques that have been used, and the typical end-to-end procedure followed. In particular, we aim to highlight the constraints that are particularly applicable to community networks, and analyze which deep learning approaches are likely to be successful in consideration of such constraints. In doing so, we identify the gap in the literature that shows the opportunity for useful research into a near-real-time fine-grained traffic classification system for community networks.

CCS CONCEPTS

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

KEYWORDS

Network traffic classification, deep learning, community networks

1 INTRODUCTION

Network traffic classification — the task of categorizing network traffic into different classes — has many important applications in traffic engineering. One of these applications is providing networks with smoother quality of service (QoS) by assigning different priorities to different applications' flows based on their classification. For example, applications involving video and voice traffic rely on fast packet transmission, whereas speed requirements are not as important for text services like email applications [12]. Other applications of traffic classification include detecting malware for security purposes, billing services for ISPs, as well as managing network resource use [19]. Due to its many important applications, various approaches to traffic classification have been studied extensively in the literature.

Classic approaches that have historically been used for traffic classification include port-based methods, payload-based methods such as deep packet inspection (DPI), and then classical machine learning techniques such as random forests or k-nearest-neighbours algorithms. However, these approaches have each been shown to have respective weaknesses in classifying modern network traffic. Classification methods that rely on port numbers are no longer reliable since many applications do not use standard ports. Additionally some applications use a technique known as port obfuscation to disguise their traffic by using well-known port numbers [25]. DPI techniques require substantial time and computational resources to derive, update and maintain the rules and patterns used to identify application signatures [6] — and this task has been made more difficult by the encryption of traffic [8]. Additionally, a disadvantage of classical machine learning approaches is that they rely on human-engineered features derived from data flows — which limits their generalizability [19]. In light of the problems that traditional approaches face with modern network traffic, deep learning approaches have recently been explored in order to improve on the performance of these methods. Many such studies have reported excellent results. Consequently, it is apparent that deep learning models offer strong potential for successful, accurate and generalizable approaches to traffic classification.

In particular, for our purposes there is demand for a lightweight deep learning approach to traffic classification that can be utilized by community networks in low resource environments. This would require such a model to balance the trade-off between classification speed and timeliness on the one hand, and computational efficiency in terms of low resource usage on the other. We will thus first highlight these constraints that are applicable to traffic classification for community networks, and then explore different deep learning approaches that have been taken in the literature — with emphasis on evaluating how successfully they consider these constraints. In doing so, we aim to identify the key aspects of such approaches that will enable the development of a low-resource, computationally efficient model that provides near-real-time accurate traffic classification for community networks.

The rest of the paper is organized as follows. In Section 2 we provide an overview of the constraints that apply to traffic classification, particularly those relevant to community networks. In Section 3 we outline the key concepts that pertain to the development of deep learning traffic classifiers. Section 4 then presents the prominent deep learning techniques that have been applied by relevant research papers. This is followed by a discussion and critical analysis in Section 5 of the success of various approaches with respect to the provided constraints. Lastly, we summarize our key conclusions in Section 6.

2 OVERVIEW OF CLASSIFICATION CONSTRAINTS

There are a number of constraints that must be considered for traffic classification tasks. These include whether the traffic is encrypted, whether the classification is in real-time or offline, and whether there are resource utilization constraints. Some of these constraints are particularly relevant for community networks. We

thus first consider general constraints applicable to community networks, and then explore the specific constraints pertaining to traffic classification that must be considered.

2.1 General Constraints for Community Networks

Community networks refer to network systems that are built, deployed and managed by local geographical communities (often with the help of non-profit organizations) to support their community by facilitating easier connectivity, communication and access to online services [15]. Technically speaking, these network infrastructures are typically distributed, decentralized low-resource systems that use low-cost hardware and wireless technologies to connect network nodes [4]. Wireless mesh networks are a popular wireless networking technology for this purpose due to their robust nature and ease of implementation compared to wired networks [11]. These are self-organizing networks in which all nodes act as routers which forward data, and which can dynamically adapt to nodes entering and exiting on an ad hoc basis — thus allowing them to be highly robust and scalable [17].

Community networks aim to help close the digital divide by providing cheaper connectivity in typically rural areas or developing regions that otherwise would struggle to obtain affordable and reliable internet access [15]. An example of such a community network in South Africa is the iNethi network, which is currently deployed in Ocean View — a township in Cape Town [10].

There are various general constraints that affect low-resource community networks, including — amongst others — bandwidth, performance and security issues. Limited bandwidth is one of the key constraints typically applicable to low-resource community networks [20]. These low-resource environments often have slower traffic transmission speeds which impact network users' application usage. Another key constraint is that of performance — in terms of computational resources available in such networks. Community networks generally consist of simple low-cost hardware infrastructure [4], and as a result the computational power in such networks is limited. An additional challenge faced in the design of community networks using wireless mesh technology is their vulnerability to network security issues, because traditional security schemes cannot be applied as a result of community networks' computational constraints [20].

2.2 Specific Traffic Classification Constraints

We now discuss some of the specific constraints that are considered when designing a traffic classifier — namely traffic encryption, online versus offline classification and computational intensity — and relate these to their applicability to community networks.

2.2.1 Encryption of Traffic. A key factor that affects the traffic classification task is whether or not the traffic is encrypted. Traditional classification methods do not perform as well for encrypted traffic, since these approaches often rely on information based on the content of packet payloads. The encryption of traffic is thus one of the main drivers behind turning towards deep learning approaches to the classification task.

Some studies have continued to explore classification of unencrypted traffic, however, namely to illustrate the effectiveness of deep learning techniques on such data relative to classical approaches. For example, Lim et al. produced a paper that demonstrates the effectiveness of convolutional neural networks and residual networks on unencrypted data, achieving excellent F1 scores of over 0.95 [12] (See Section 3.4 for an explanation of the F1 score). The study uses entire packet payload contents as data, and whilst this may not be as useful for encrypted traffic, it does serve to show the success that deep learning techniques can have using the mass of data available in traffic flows.

However, due to the prevalence of encryption in modern internet traffic, our focus in this paper will be on studies that classify *encrypted* traffic using deep learning. Many such studies have been conducted in recent years, and these will be explored in Section 4.

2.2.2 Online Classification. Traffic classification can also be divided into into online versus offline classification [19]. Online classification pertains to the scenarios where flows need to be classified in real-time within the first few packets. This can also be referred to as 'early' application identification [3]. For example, this is applicable in QoS provisioning since the classification results directly affect the priorities assigned to different flows. On the other hand, offline classification is used where the traffic does not need to be classified immediately and there is time for posterior analysis to be performed on the entire flow. This applies to situations such as helping ISPs determine the correct amount to bill customers, since determining application usage for customers does not need to take place in real-time.

Online classification can typically be considered either through a flow-based or packet-based approach. The flow-based approach aim to classify traffic flows using as few of the initial packets in a flow as possible, whilst packet-based approaches attempt to classify traffic on a packet-by-packet basis without needing to first associate packets with a particular flow.

This paper will try highlight approaches in the literature that are applicable to online classification in particular, since this will be suitable to the provision of QoS in community networks — which is where our research focuses. Deep learning approaches have been shown to be very effective for online classification, as shown by the flow-based study performed by Lopez-Martin et al. where high accuracy is attained despite only using the first 20 packets of each flow [13]. Their results even show that using just the first five or more packets were sufficient to achieve accurate classification results.

2.2.3 Low Computational Resources. Lastly, another important constraint that should be considered for the traffic classification task is that of computational efficiency. Namely, both training time and computational resource usage for the models are relevant, since deep learning architectures are highly computationally intensive. This is particularly prevalent in the network traffic classification environment, since models will need to be periodically retrained to account for the continually changing nature of Internet traffic and resulting aging of training data, especially for mobile application traffic [21]. Additionally, deep learning models can be very memory intensive when a large number of layers are used, and this will not be suitable for deployment in low-resource environments. Accordingly, in this paper we will try to emphasize deep learning models

that consider computational efficiency and resource utilization as an important factor in their design.

3 KEY CONCEPTS IN DEEP LEARNING FOR TRAFFIC CLASSIFICATION

With the above constraints serving as the context upon which the traffic classification task will focus, in this section we now discuss the key concepts pertaining to the classification task that are apparent in the literature.

3.1 Overview of Typical Procedure

As a general framework upon which to explore papers that have developed traffic classification deep learning models, we first give an overview of the general overall procedure followed [19]. First, the specific classification problem itself is decided upon — for example whether the study will be classifying by application (such as Skype vs YouTube), or by protocol (such as HTTP vs FTP). Appropriate labelled data is then obtained to be used for model training — this requires data preprocessing to clean the data and extract appropriate features that will be used in the model. The next step is deciding upon the models to be used - such as convolutional neural networks, recurrent neural networks or combinations thereof. Finally, the model is trained and fitted using the training data, after which the model results are evaluated on test data using appropriate classification metrics.

3.2 Data Preparation Pipeline

Various approaches have been taken in the literature to prepare appropriate training data that will be suitable for fitting classification models. Different papers conduct the process of collecting and preprocessing the data in different ways, whilst thereafter features from the data must be chosen for the model.

3.2.1 Data Collection and Preprocessing. The input data for classification models is typically raw packet data (often in the form of PCAP files) or information extracted directly from this data [19]. This data needs to be labelled to allowed the model to train from it. In order to obtain such labelled data, one approach taken by researchers is to make use of public labelled datasets. For example, this approach is used in the study in [12] where labelled PCAP traffic traces sourced from the UPC's Broadband Communications Research Group is used as the input data.

Some studies, however, need data that specifically relates to their particular classification goal. These studies will then capture their own traffic traces, and then will need to label the data with ground-truth values. One approach for labelling the data is to use deep packet inspection (DPI) packages which make use of a database of application signatures to identify different classes from traffic traces. Bujlow et al. performed an independent study of different DPI tools, evaluating their traffic classification accuracy [5]. The main result of their study shows that PACE — a commercial package — offers the highest accuracy; but the open-source tools nDPI and Libprotoident also attain very high accuracy. This approach is adopted by Lopez-Martin et al., who use the nDPI tool — which handles encrypted traffic — to label their dataset. [13].

In terms of preprocessing, one issue that often presents itself in the literature is the lack of a balanced dataset. For example, there may be a high amount of Skype flows, but few WeChat flows, which makes the training process more difficult. Some studies do not account for this imbalance, such as Lopez-Martin et al, and as a result do not perform as well when classifying some of the underrepresented classes [13]. One approach to solving this problem, used by Lim et al., is to preprocess the raw data to simply extract an equal number of flows or packets for each label class [12]. This method, also used by Lotfollahi et al., is called under-sampling, and usually works by randomly removing some of the major classes' samples until the dataset is more balanced [14].

Another approach to the imbalanced data problem is using a deep learning model called a *generative adversial network (GAN)*. The model is able to generate synthetic samples from the underrepresented classes in order to balance the number of samples between the minor and major classes in the training data. This is applied in a study by Vu et al. who apply a GAN to balance the classes in a public network traffic dataset [22]. Their results show a clear improvement in classifiers on both accuracy and F1 score (see Section 3.4 below for an explanation of these metrics) using the augmented dataset as compared to running the same classification algorithms on the imbalanced data.

3.2.2 Feature Selection. Once a labelled dataset has been collected, the next step is to prepare the actual training data that will serve as the input to the relevant model. One of the main advantages of deep learning techniques, is that it allows for automatic identification of relevant features during the learning process. Many papers thus simply take portions of the packet traces or packets in their entirety as the input, and allow the network to discover patterns in the data to facilitate the classification. For example, this approach is used by Lotfollahi et al., who allow their packet-based classification model to automatically select features by just using the IP header and first 1480 bytes of each IP payload as the input to their networks [14].

Another approach taken by some researchers is to carefully extract specific features from packet data to serve as input to their model. Such an approach is usually used for flow-based classification as opposed to packet-based. Lopez-Martin et al. found that source and destination ports, TCP window size, packet direction, packet timestamp and payload size (for the first 20 packets in a flow) were useful features to use as inputs to their model [13].

3.3 Model selection

After the training data has been prepared, the models that will be used as classifiers should be decided upon. Generally a few different network architectures are considered and compared in terms of various evaluation metrics explained in the section below. Once suitable models have been selected, appropriate hyperparameters (such as number of layers, number of nodes in each layer, and many more) are chosen by fine-tuning trained models. There are many approaches taken regarding fine-tuning of models, with a popular technique adopted in the literature being that of grid search [12, 22] — where all combinations of specified potential values for each hyperparameter are tested to see which yields the optimal model performance.

3.4 Model evaluation

There are a few key metrics that are typically used for evaluating the success of classification models, namely *accuracy*, *precision*, *recall* and *F1 score* [18].

It is important to explain these metrics as some of these terms will be used to evaluate and compare different deep learning approaches in Section 4 below. Note that typically each measure is computed for each label present in the data, with a weighted average of the labels' metrics then taken as overall measures of the model's performance.

These metrics' definitions are based on the following four definitions: (1) *true positives* (TP) are the number of times a real label is correctly predicted, (2) *false positives* (FP) are the number of times a label is predicted when in fact it is not the true label, (3) *true negatives* are the number of times we correctly indicate that a particular label is not present, and (4) *false negatives* occur when we indicate a label is not detected when it is actually the true label.

The metrics are then defined as follows:

$$\begin{aligned} Accuracy &= \frac{TP + TN}{TP + TN + FP + FN} \\ &= \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \\ Recall &= \frac{TP}{TP + FN} \\ Precision &= \frac{TP}{TP + FP} \\ F1\ score &= 2 \times \frac{Precision \times Recall}{Precision + Recall} \end{aligned}$$

Accuracy is one of the most popular metrics used for evaluation of classifiers, indicating the proportion of correct classifications relative to the total number of predictions made. This metric is a good indication of model performance if the dataset is balanced, however it can give a false representation of the model if the dataset is highly unbalanced. For example, if 99% of a dataset has one label and 1% another, a model that always predicts the first label will achieve 99% accuracy even though it never classifies the second label correctly.

The F1 score is often used to solve this problem, calculated as the harmonic mean of the precision and recall metrics. The F1 score is a value between 0 and 1 and can be interpreted as a percentage. It is often regarded as the most important metric for classification tasks since it gives a better indication of a model's performance on datasets which are unbalanced.

4 PRESENTATION OF DEEP LEARNING APPROACHES

In this section we present and discuss representative papers that have implemented different deep learning techniques for traffic classification, with appropriate analysis and links to the constraints we highlighted in Section 2. The deep learning methods that we consider are *multi-layer perceptron models*, *convolutional neural networks*, *recurrent neural networks*, and *auto-encoders*. In particular, the papers we will discuss are based on encrypted traffic, since this is most applicable to real-world data.

4.1 Multi-Layer Perceptron

The *multi-layer perceptron (MLP)* model is the first basic neural network architecture, a non-linear model used for supervised learning [7]. The model consists on an input layer, one or more hidden layers, and then an output layer - with each layer consisting of a set of nodes, called neurons. Deep learning models, in particular, refer to neural networks that contain more than one hidden layer. Each neuron in a layer is fully connected by edges to every neuron in the next layer, with each edge having a weight parameter. The training data forms the input to the first layer, whilst thereafter each neuron computes a weighted sum of the outputs of the previous layer's neurons with each corresponding edge's weight, with a non-linear activation function applied to this weighted sum. The output layer's outputs are then used for predictions. The network then 'learns' the optimal weight parameter values that minimize the loss between the predictions and true values through a process called backpropagation.

In practice, pure MLP models are rarely used for traffic classification due to their computational complexity as well as comparatively low accuracy relative to other deep learning techniques [19]. Aceto et al. performs a study comparing numerous deep learning architectures, with the results clearly showing the relatively poor performance in traffic classification of MLPs with both one and two hidden layers, compared to other architectures [1]. However, this study shows the usefulness of MLPs as a baseline to which other models can be compared. Additionally, many deep learning architectures rely on using one or two fully connected MLP layers before the output layer, and hence MLPs also are valuable in this regard.

4.2 Convolutional Neural Networks

Convolutional neural networks (CNNs) are one of the most popular deep learning architectures that have been applied in the traffic classification field, despite traditionally being applied to recognizing patterns in image data [16]. CNNs are designed for processing data stored in a grid-like structure, uncovering local spatial patterns within the data [9]. Similar to MLPs, CNNs consist of an input layer, a series of hidden layers, and then an output layer. However, the hidden layers for CNNs typically consist of both convolutional layers — which use image kernels with a small number of learnable parameters to extract key spatial patterns from the input, and pooling layers — which are used to reduce the spatial dimensionality of the data. One-dimensional CNNs (1D-CNNs) use filters that capture spatial patterns in one-dimensional data, whereas two-dimensional CNNs' (2D-CNNs) filters, used in image recognition, can learn to recognize two-dimensional features. 1D-CNNs have been deemed ideal architectures for the traffic classification task using packet data, since they are able to recognize spatial dependencies between adjacent bytes (or packet features) in order to find key patterns that enable successful classification [14].

Aceto et al. performed a comprehensive comparison of many deep learning architectures, focusing on classifying mobile app traffic [1]. Their approach is, however, applicable to the traffic classification task in general. One aspect of their study compares the use of 1D filters, and 2D filters, in CNNs for traffic classification and observed similar performance results. They conclude that this

supports intuition that traffic data can be considered as inherently sequential, and hence 2D filters are not necessary in better capturing spatial information in traffic data. This corroborates the findings made by Lotfollahi et al. of the effectiveness of 1D-CNNs for traffic classification [14]. It should be noted that these results are applicable to packet-based classifiers that use the first n bytes of a packet's payload as input, and have not necessarily been shown to be relevant when selected features are chosen as input.

Wang et al. [24] proposes an end-to-end procedure for using a 1D-CNN to characterize encrypted traffic into twelve classes including Email, Chat and Streaming - each split into both VPN and non-VPN-based traffic. The study makes use of the public ISCX VPN-nonVPN dataset as training data, which allows for comparison with other studies on the same dataset. The data is preprocessed by taking the first 784 bytes of each flow/session, and converting this into a two-dimensional image as input to the CNN. The study's CNN consists of two convolutional layers, each followed by a max pooling layer, after which two fully-connected layers link to the output layer. The model outperformed a state-of-the art C4.5 decision tree classifier operating on the same dataset, highlighting the suitability of this model for encrypted traffic classification. Furthermore, the study compares a 2D-CNN to their 1D-CNN model and observes higher accuracy, precision and recall for the 1D-CNN for the majority of the labels – further supporting the discussion above. We also should note, however, that this approach of using all the initial bytes of a flow is likely to be more computationally intensive than if features were more selectively chosen.

Chen et al. also use a CNN-based approach, developing a classification framework which they termed Seq2Img [6]. However, in particular, their study is aimed at online classification. To this end, they opt to use only the first 10 packets in a flow in order to classify packets in a flow as early as possible. The method adopted consists of extracting raw flow information such as packet sizes and inter-arrival times, as opposed to hand-crafted statistical features that depend on a priori knowledge of the entire flow. A technique called a Reproducing Kernel Hilbert Space embedding is then used to embed these features into an image that is fed into their CNN classifier. Similar to that used in the study by Wang et al. above [24], this CNN is made up of two convolutional layers and two pooling layers, but with three fully connected layers before the output layer. Seq2Img attained 99.84% accuracy when classifying by protocol and 88.42% accuracy when classifying by application (on a private dataset consisting of 5 applications and 5 protocols). The study compares this performance to both classical machine learning methods such as Naive Bayes Classifiers and Decision Trees, as well as to an MLP model, with Seq2Img significantly outperforming each of these. This highlights the effectiveness of their CNN-based model, especially when considering our relevant constraints in that it prioritizes online classification and minimizing computational complexity by using selective data features.

4.3 Recurrent Neural Networks

Recurrent neural networks (RNNs) are a particular class of neural networks designed for working with sequential data, where outputs may depend on previous inputs in a sequence [9]. RNNs make use of loops in the hidden layers to allow neurons to retain temporal information [19]. LSTMs are very popular variants of RNNs that help to solve the 'vanishing gradient' problem (which occurs during the backpropagation algorithm when gradients are computed over multiple stages) to allow the network to better learn long-term dependencies in a sequence [9]. Such networks are appropriate for the traffic classification task, since traffic flows consist of a sequence of packets where packet data could depend on previous packets, and also bytes within a packet contain sequential information.

Lopez-Martin et al. performed a study that highlighted the success that RNNs can have as part of traffic classifiers [13]. They aimed to perform classification on a dataset consisting of labelled flows from 108 application services - with the particular goal of improving Internet of Things (IoT) traffic management. For each flow, only up to the first 20 packets are chosen to form the input to the network, with six features extracted for each packet — namely source port, destination port, payload size, TCP window size, timestamp (for measuring inter-arrival time) and packet direction. Hence the input to their models was a time series of 20 feature vectors per flow, with each vector consisting of six features. Their RNN model (using LSTM layers in particular) achieved an aggregated F1 score score of above 95%, illustrating the effectiveness of RNNs for traffic classification. Interestingly, in this study Lopez et al. also created models that combined RNNs and CNNs, and noted that this combination of the two architectures achieved a better performance than either model individually. Such results coincides with the conclusions made by Aceto et al. [1] that 'advanced hybrid DL architectures' offer strong potential for improving the performance of traffic classifiers. Their best combined model achieved an F1 score of 95.74%, with accuracy of above 98% for the 15 most frequent labels in the dataset. With respect to the constraints we are focusing on in this paper, we note that this study was both useful for online classification (since only the first few packets in a flow are necessary for classification) and also considered computational efficiency by choosing not to include more packets than 20, despite the potential for this to improve results.

Wang et al. also proposed a classifier consisting of a combination of RNNs and CNNs, designed for detecting malware [23]. They propose an intrusion detection system that learns hierarchical spatial-temporal features, termed HAST-IDS. In essence, the system consists of transforming raw traffic data into the format of 2D images which are fed into a CNN, the result of which is passed into an LSTM network. Their hypothesis was that the CNN will learn the spatial features in the network packets, and then the LSTM will capture temporal information — thus allowing the overall network to learn the important features from the data for classification. The HAST-IDS model achieved high accuracy compared to multiple other published methods that use the same dataset, which demonstrates the effectiveness of their proposed system.

4.4 Stacked Autoencoders

Stacked autoencoders are our final deep learning technique from the literature that we shall consider. Autoencoder (AE) neural networks describe a method of unsupervised learning used for dimensionality reduction and automatic feature extraction, by attempting to reconstructing the input at the output and minimize the reconstruction error, thus removing noise [14]. Models called stacked

autoencoders (SAE) are typically used more in practice to achieve better performance — such models consist of several autoencoders stacked after each other, with the output of the previous one serving as the input to the next.

One study that used SAEs effectively for the classification task is that done by Lotfollahi et al [14]. The scheme, termed Deep Packet, is very well-executed and makes use of SAEs and CNNs to classify both by application and by traffic type (such as chat and streaming). The SAE model is made up of five fully connected layers, whilst the core of the CNN is two convolutional layers followed by a pooling layer. Unlike many of the other research papers we have considered which classify flows, Deep Packet performs packet-based classification. This approach is the ideal approach for the real-time classification constraint, since deployed models will not need to first identify packets as being part of a flow before classifying them. Lotfollahi et al. justifies the packet-based approach, despite the payloads being encrypted, because their deep learning models are able to learn the distinguishable patterns within the encrypted data that characterize applications, despite the content itself being inaccessible due to encryption. Their results testify to this end, with their CNN and SAE models attaining highly impressive F1 scores of 0.95 and 0.92 respectively. These are shown to clearly outperform previous classifiers on the same dataset that made use of k-nearest neighbours and C4.5 decision tree algorithms.

It is interesting to note that the study by Aceto et al. [1] also performed useful experiments on the training time of various architectures including SAEs, CNNs and LSTMs. An interesting observation made is that their SAE model was considerably faster to train than the CNN model. They also noted that — when trained on the same dataset — the LSTM models they used were faster to train than the CNNs. Whilst it is difficult to do a direct comparison of different deep neural network architectures in terms of computational efficiency regarding training time (due to variability in number of layers and nodes per layer), their study does still provide a useful framework for looking at training times — and indicate that SAEs might be a useful consideration for improving computational efficiency.

5 DISCUSSION

There has clearly been substantial research into the traffic classification task, guided by each particular study's specific aims. This literature can thus help provide a guide as to which approaches will work for the particular constraints applicable to our desired deep learning-based traffic classifier for community networks. Therefore, we will now discuss and compare the results of our above research, with particular emphasis on the approaches that we think will work best for our key constraints of encrypted traffic, real-time classification and computational efficiency.

The majority of papers we have anaylzed dealt with encrypted traffic. This is testament to the capabilities of deep learning towards overcoming the difficulties that classical approaches face when dealing with encryption. The literature has made it evident that this encryption constraint can be dealt with in multiple ways - either by using general packet features in a flow such as packet length and inter-arrival times which are independent of the encryption [6, 13], or through training neural networks to learn patterns in the encryption without needing to decipher the content [14].

Regarding the data itself, another key point that we saw highlighted in the literature is that of the importance of a balanced dataset for successful classification across all classes. Studies that do not deal with the issue of imbalanced data achieve poor results when classifying the under-represented classes [13, 24]. Two viable approaches to overcome this issue include under-sampling from the major classes (if sufficient raw data is available) [12] or artificially generating data using GANs to account for the imbalance [22]. The latter approach is likely more appropriate to scenarios when the amount of data is too small to consider under-sampling.

There are many insights to be gathered from our analysis of different deep learning approaches in Section 4. Firstly, it appears clear that as a stand-alone model pure MLP models are not as successful as other deep learning techniques. CNN models appeared the most popular technique in the literature, due to their ability to capture spatial patterns within the traffic data. In particular, there is evidence to suggest that 1D CNNs are more suitable than 2D CNNs to the classification task [14, 24]. However, when classifying traffic based on flows (rather than more fine-grained packet-by-packet classification), LSTM RNN networks also have been shown to be very effective, due to their ability to retain temporal information regarding the relationships between successive packets [13]. Using hybrid model structures, such as applying a CNN and then an LSTM, also appears to be a promising route to explore in order to improve model performance[1, 13, 23]. In the CNN and LSTM combination scenario, this has the advantage of being able to recognize both spatial and temporal patterns in the traffic data.

It is clear that using statistical features of entire traffic flows such as flow duration and mean packet size [2] will clearly not be suitable for online classification as the entire flow must have ended before the packets in a flow can be classified. So, when considering the constraint of real-time classification, it appears that two main strategies have arisen in the literature. Either features based on only the initial packets in a flow are used to classify the packets from a flow [6, 13], or classification is done on a packet-by-packet basis independent of the flow [14]. It appears that there may be a potential trade-off between computational efficiency and online classification in this regard. The packet-based classification certainly meets the real-time constraint better, since packets do not first need to be identified as part of a flow before they can be classified. However, classifying based on initial packets in the flow has the advantage of likely being less computationally intensive, since selected features from these packets are used as opposed to the overall packet payload data needed for packet-based classification. Additionally, the approach using initial packets in a flow allows models to capture temporal flow-based information which could aid the performance of such models.

Another important constraint to be discussed is that of the training time for different models. It is difficult to compare different deep learning architectures directly in terms of training time, since there is high variability in network parameters. However, the obvious trend is that the greater the number of layers, and greater the number of neurons in each layer, the longer the model will take to train. As such, models with fewer parameters should be emphasized in order to adhere to the computational efficiency constraint,

especially since models will have to be regularly retrained in the low-resource community networks to account for changing traffic patterns. Additionally, we note that it is suggested that SAE traffic classifiers are faster to train than other deep learning architectures [1], and so this could be explored further to particularly emphasize the low computational resource usage constraint.

Finally, we note that to the best of our knowledge, it appears that no studies exist yet in the literature on a traffic classifier aimed primarily at suitability for community networks — with studies having been done that focus on other areas such as IoT, mobile traffic, security systems etc. As such, there is opportunity for further contribution to the knowledge in the traffic classification field by using the current literature to guide work on a low-resource, real-time traffic classifier applicable to the constraints of community networks.

6 CONCLUSIONS

Traffic classification clearly plays an important role in the field of network management for reasons such as traffic engineering and QoS. Accordingly, there has been extensive research into the topic in the literature. In community networks in particular, the classification task is made more difficult by the added constraints associated with these low-resource environments — and so these constraints guided our research.

In this review, we provided an overview of community networks, and the relevant constraints of encrypted traffic, real-time classification and low computational resource usage that are applicable to the traffic classification task in community networks. We explored the typical end-to-end procedure followed in the literature regarding building a deep learning traffic classifier, and then examined the different deep learning techniques that have been employed by researchers to this end. This was followed by a critical discussion and analysis of the approaches that seemed likely to be successful in satisfying the relevant community network constraints, in order to see where our own research can fit into the field.

From the above analysis, it appears clear that in order to satisfy the online classification constraint, we should explore either purely packet-based classification or classification that only relies on the initial packets in a flow. In terms of performance, it seems that CNNs and LSTMs are the most popular deep learning techniques for traffic classification, and also that a hybrid structure that combines these models could be explored as a promising avenue for improved performance. Additionally, smaller deep learning models should aim to be emphasized in order to minimize computational resource usage and training time, whilst other deep learning architectures such as SAEs could also be looked into in terms of relative computational efficiency.

Going forward, the first step will be building a pipeline to extract community network traffic data in a suitable format to be used for classification. The open-source nDPI tool appears to be a viable strategy for labelling the data [5], but other options shall also be explored. Additionally, we should consider adopting the technique of under-sampling in order to produce a balanced dataset that the review has shown is important for successful overall model performance. Thereafter, we can begin exploring different deep learning traffic classifiers guided by the above constraint analysis in order to develop options for real-time, computationally-friendly classification models that will be suitable for low-resource community networks.

REFERENCES

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

[2] Pedro Amaral, Joao Dinis, Paulo Pinto, Luis Bernardo, Joao Tavares, and Henrique S Mamede. 2016. Machine learning in software defined networks: Data collection and traffic classification. In *2016 IEEE 24th International Conference on Network Protocols (ICNP)*. IEEE, 1–5.

[3] Laurent Bernaille, Renata Teixeira, and Kave Salamatian. 2006. Early application identification. In *Proceedings of the 2006 ACM CoNEXT conference*. 1–12.

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

[5] Tomasz Bujlow, Valentín Carela-Español, and Pere Barlet-Ros. 2015. Independent comparison of popular DPI tools for traffic classification. *Computer Networks* 76 (2015), 75–89.

[6] Zhitang Chen, Ke He, Jian Li, and Yanhui Geng. 2017. Seq2Img: A sequence-to-image based approach towards IP traffic classification using convolutional neural networks. In *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, 1271–1276.

[7] Simon S Cross, Robert F Harrison, and R Lee Kennedy. 1995. Introduction to neural networks. *The Lancet* 346, 8982 (1995), 1075–1079.

[8] Alberto Dainotti, Antonio Pescape, and Kimberly C Claffy. 2012. Issues and future directions in traffic classification. *IEEE network* 26, 1 (2012), 35–40.

[9] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press.

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

[11] Johnathan Ishmael, Sara Bury, Dimitrios Pezaros, and Nicholas Race. 2008. Deploying rural community wireless mesh networks. *IEEE Internet Computing* 12, 4 (2008), 22–29.

[12] Hyun-Kyo Lim, Ju-Bong Kim, Joo-Seong Heo, Kwihoon Kim, Yong-Geun Hong, and Youn-Hee Han. 2019. Packet-based network traffic classification using deep learning. In *2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*. IEEE, 046–051.

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

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

[15] Panagiota Micholia, Merkouris Karaliopoulos, Iordanis Koutsopoulos, Leandro Navarro, Roger Baig Vias, Dimitris Boucas, Maria Michalis, and Panayotis Antoniadis. 2018. Community networks and sustainability: a survey of perceptions, practices, and proposed solutions. *IEEE Communications Surveys & Tutorials* 20, 4 (2018), 3581–3606.

[16] Keiron O'Shea and Ryan Nash. 2015. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458* (2015).

[17] J Rejina Parvin. 2019. An Overview of Wireless Mesh Networks. In *Wireless Mesh Networks-Security, Architectures and Protocols*. IntechOpen.

[18] David Martin Powers. 2011. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. (2011).

[19] Shahbaz Rezaei and Xin Liu. 2019. Deep learning for encrypted traffic classification: An overview. *IEEE communications magazine* 57, 5 (2019), 76–81.

[20] Muhammad Shoaib Siddiqui et al. 2007. Security issues in wireless mesh networks. In *2007 International Conference on Multimedia and Ubiquitous Engineering (MUE'07)*. IEEE, 717–722.

[21] Vincent F Taylor, Riccardo Spolaor, Mauro Conti, and Ivan Martinovic. 2017. Robust smartphone app identification via encrypted network traffic analysis. *IEEE Transactions on Information Forensics and Security* 13, 1 (2017), 63–78.

[22] Ly Vu, Cong Thanh Bui, and Quang Uy Nguyen. 2017. A deep learning based method for handling imbalanced problem in network traffic classification. In *Proceedings of the Eighth International Symposium on Information and Communication Technology*. 333–339.

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

[24] Wei Wang, Ming Zhu, Jinlin Wang, Xuewen Zeng, and Zhongzhen Yang. 2017. End-to-end encrypted traffic classification with one-dimensional convolution neural networks. In *2017 IEEE International Conference on Intelligence and Security*

Informatics (ISI). IEEE, 43–48.

[25] Jun Zhang, Xiao Chen, Yang Xiang, Wanlei Zhou, and Jie Wu. 2014. Robust network traffic classification. *IEEE/ACM transactions on networking* 23, 4 (2014), 1257–1270.