Traffic Classification in Community Networks using Deep Learning

Matthew Dicks (DCKMAT004)
University of Cape Town

ABSTRACT

Community networks are networks that are run by the citizens for the citizens. Communities build, operate and own open IP-based networks which promotes individual and collective participation. These networks run with less resources than traditional Internet Service Providers so understanding how these resources are being used is necessary to ensure a good Quality of Service. Traffic classification has played an important role in helping achieve this. This literature review will look at how deep learning has been implemented to classify internet traffic. It will compare the different deep learning architectures based on their predictive performance as well as their computational efficiency and will also look at the different ways that data can be extracted from network traffic and compare their performances. It can be seen from this review that deep learning outperforms conventional machine learning methods and that raw data outperforms hand picked features. These findings show that deep learning will be a key tool used to better understand community networks.

CCS CONCEPTS

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

KEYWORDS

Community Networks, Deep Learning, Traffic Classification

1 INTRODUCTION

This literature review will give a background on deep learning methods used for traffic classification, the process of categorizing network traffic into appropriate classes, and their potential to be implemented in community networks. Traffic classification has become vitally important to advanced network management. Quality of Service (QoS) control, pricing, planning resource usage as well as malware and intrusion detection all rely on accurate traffic classification [16]. Due to the increased throughput of internet traffic, as well as new technology such as the Internet of Things (IoT) becoming more prevalent in our lives, it is becoming vital to understand how a network's resources are being allocated. Accurate traffic classification will help network management understand what applications are using the resources, and with this information a better allocation of resources can be realized. This will help in achieving the ultimate goal of this project, which is to improve the quality of service and traffic engineering in the context of a community network.

This project aims to to utilize deep learning to classify encrypted internet traffic for the purpose of improving the QoS and traffic engineering in community networks. Different deep learning architectures and data collection techniques will be analysed. Their accuracy and computational efficiency will be compared to determine the best model suited for traffic classification in a community network.

Previous traffic classification methods used port numbers, Deep Packet Inspection (DPI) or features hand crafted by an expert to classify internet traffic. There are a number of issues with each of these approaches. Using port numbers for traffic classification is the simplest and fastest way to classify internet traffic but due to port obfuscation, random port assignments, port forwarding, protocol embedding and network address translation (NAT) the accuracy of port-based methods has decreased [13]. The accuracy of DPI methods have been reduced due to the increase in the amount of encrypted traffic and user privacy agreements. DPI also has a large computational overhead [16]. Features that have been handcrafted by an expert can suffer a lack of generality because they focus on only a few key features. *Aceto et al* notes that these hand-crafted features rapidly become outdated due to the evolution and mix of internet traffic. It is also an expensive method because experts have to be hired and the hand picking procedure is subject to human error [2].

Deep learning is a proposed solution to these issues. Deep learning can learn a hierarchical set of features from raw or statistical flow data. It therefore removes the need for features to be hand crafted by an expert and also promises to be able to classify encrypted traffic. A system can leverage the power of this approach to build an end-to-end deep learning system which will limit the need for feature selection and engineering which would reduce the amount of prepossessing. Deep learning houses powerful architectures such as convolutional and recurrent neural networks that can learn the spacial and temporal patterns in the data. These methods thrive when large amounts of data are available which makes it a great approach to classify traffic.

2 COMMUNITY NETWORKS

A Community network is a network that is run by the citizens for the citizens, where communities build, operate and own open IP-based networks [3]. These networks promote individual and collective digital participation in rich, poor, rural and urban areas [14]. They also have minimal barriers to entry because their governance, knowledge and ownership is open [14]. These networks are usually run by non-profit organizations who can interact with stakeholders to provide services such as local networking, voice, data and internet access [14]. Community networks are large-scale distributed

and decentralized systems with many nodes, links, services and traffic. They are also characterized by their dynamic and heterogeneous behavior [14].

2.1 The potential for traffic classification in community networks

Community networks house a wide range of application services such as VoIP, content distribution, on-demand and live streaming media, instant messaging as well as back-ups and updates, but the challenge is that they have to provide these services on links and servers with limited capacity. They also have to manage the diverse and volatile resources in the network [3]. Therefore to be able to provide a good QoS to the users, network managers should be able to know how their resources are being used. This is where traffic classification comes in, it will provide a way for the citizens and non-profit organizations who are running the network to view what applications are using what resources. This will allow them to take more informed actions to try and improve the QoS delivered to their customers.

The end goal of this project will be to perform traffic classification with deep learning models. These models have a large number of parameters, which makes training and running them computationally expensive. Community networks try to provide their service at a low cost and one of the ways they do this is to use cheaper hardware and resources [3]. Since these models will need to be deployed in a low resource environment they will need to be compared not just based on their predictive performance but also on their computational efficiency.

3 DATA COLLECTION TECHNIQUES

In machine learning the models are only as good as the data that they are trained on. That is why it is imperative to collect a large and representative data-set. This will allow the models to generalize and perform well on unseen data. In the context of traffic classification there are a few public data-sets but there is no commonly agreed upon data-set that can be used for most traffic classification problems [16]. This poses an issue because without an agreed upon data-set it becomes hard to compare models, feature selection techniques and performance across research papers. A common data-set that has been used by some of the papers reviewed is the "ISCX VPN-nonVPN" traffic data-set [5]. Models produced and systems built using this data-set can be reliably compared. The other research done used public data-sets specific to the problem they were working on, for example intrusion detection. Some researchers collected their own data, *Taylor et al* created their own data-set by creating accounts for applications and simulating interactions which allowed them to perform application fingerprinting [17]. Since there is no single data-set to test and compare models, care must be taken to ensure that reliable conclusions about the performance of models are drawn.

Another area where the research differs is in data labeling. Obtaining the labels is often not trivial and in some studies labels have to be obtained by by free DPI modules such as nDPI [16]. This restricts the performance of the models being investigated.

One of the benefits of using deep learning is that the models can learn a hierarchical set of features from raw data. This raw data can come from individual packets or traffic flows. Some studies have also used statistical data to classify packets or flows into categories. This section will give an overview of how the data was extracted from network traffic so that it could be used as input into a deep learning model.

The object of classification can either be flows or packets. This section is split into Flow and Packet classification since the inputs provided to the models and the effect of the inputs on the models may differ depending on whether flows or packets are being classified.

3.1 Data Collection for Flow Classification

More often than not the object of classification are flows [2], these flows can be bidirectional or unidirectional. Bidirectional means that the traffic flows in two directions, from source to destination and from destination back to source. Unidirectional means that the source to destination and destination to source paths are regarded as different objects. *Wang et al* found that bidirectional and unidirectional flows performed the same for data extracted from all protocol layers, but bidirectional outperformed unidirectional by an average accuracy of 2.15% on application layer data. The increase in performance that comes from adopting flows as bidirectional can be attributed to the extra interaction information which allows the model to learn more features [21].

3.1.1 Types of data for flow classification.

In the papers that were reviewed it can be seen that there are four approaches to collecting data from a traffic flow [2, 4, 12, 20, 21].

The first approach is to take raw data, in the form of bytes, from some of the packets in the flow. This approach can be further divided into two categories which are defined by where the data was taken from in the packet. L7 data is where the data is only taken from the application layer, and ALL data is where the data is taken from all protocol layers. In application classification it is argued that only the application layer data, L7, is relevant [21] because this is where the characteristics of the traffic reside. The reason ALL data is proposed is because it contains application layer data and it reserves more information for the phase of encryption negotiation [21]. An advantage of taking data from individual packets is that you now have sequential data. This means that you have a time dimension where temporal patterns in the data can be learned by powerful sequence models like LSTMs.

Another approach is to extract raw data from a flow. This mean that you only consider the first N bytes from the flow and you do not care about individual packets [20]. The issue with this approach is that you may be throwing away valuable time related information that sequence models will be able learn, which would lead to increased predictive performance. The advantage of just considering the first N bytes is that it is a simple and computationally cheap way to extract the data and since these models will be deployed in community networks with less powerful resources, this will save

valuable computational power.

The third approach that will be discussed is to get time series data like packet sizes, packets directions and inter-arrival times from individual packets [16]. The studies will take the first N packets in a flow and extract this information for classification. This data can also be used in conjunction with header information such as port numbers [16]. When using deep learning models, usually more data yields better results, the problem with using time series data instead of raw packet data is that it compresses the data for a given packet into a lower dimensional space which reduces the amount of information given to the models. This method can work well with standard machine learning methods that cannot handle high dimensional data but deep learning models, such as CNNs and LSTMs have this ability [16]. However, an advantage of reducing the dimensions of the input data is that it will lead to smaller models which will be cheaper to train and deploy on community networks.

Flow statistics is the fourth way that data can be extracted from a flow. Examples of flow statistics are means, standard deviations as well as minimums and maximums for packet sizes and inter-arrival times. The advantage to using flow statistics is that it keeps the input dimensions low and allows us to build classical machine learning classifiers and sometimes multi-layer perceptrons (MLP) [16]. This approach needs to use more packets from a flow so that estimates do not have too much variance, [16] notes that this may not be suitable for fast real time classification.

3.1.2 Comparison of performances for the types of flow classification data.

As mentioned above a common approach to flow classification is to look at the data provided by individual packets. *Wang et al* extracted raw data from packets in a flow to create an image for the flow, which was then classified by a one dimensional convolutional network. Bytes where taken from either the application layers (L7) or bytes were taken from all protocol layers (ALL) [21]. When comparing the different types of data extracted from the packets it can be seen that on average ALL data out performed L7 [21]. This could be because information from previous layers, such as the transport layer, house extra information about the application. *Aceto et al* compared ALL and L7 data to hand picked time series and header features obtained from the first twenty packets. The models performed better on the ALL and the L7 data [2]. A possible explanation for this would be that the hand picked features compressed the raw data which caused important information to be lost. It also showed the power of deep learning's ability to learn general patterns from raw information.

Studies such as [4] allowed us to compare whether packets sizes, inter-arrival times and packet directions obtained from the first ten packets outperformed flow statistics. Flow statistics in this study refers to means, standard deviations and minimums and maximums for packet sizes as well as inter-arrival times. For standard machine learning models such as Support Vector Machines (SVM) and Random Forests (RF) there was no clear evidence to show that one input type was better then the other. However, for the multi-layer perceptron (MLP) the packet sizes, inter-arrival times and packet directions from individual packets outperformed flow statistics by an average accuracy of 6.21% [4]. A possible reason for the added performance is because statistics per packet have a time series aspect to them and deep learning models, like MLPs, can learn this extra information.

Using the first 600 bytes in a flow *Wang et al* achieved an accuracy of 99.69% [20]. This shows that raw data obtained from a piece of the flow, and not just from specific packets in a flow or flow statistics, can also lead to very good performance. Note that this method was not compared to a method that leveraged a time series component.

If data is going to be hand picked from packets it is important to know what features give the best performance. *Lopez-Martin et al* used the first twenty packets with six features each to classify a flow. The features that they used were source and destination ports, TCP window sizes, inter-arrival times and packet directions. They compared the importance of the features and found that source and destination ports were the most important features and the models lost 15% accuracy when they were removed [12]. This is not surprising since it is known that port numbers were essential in traffic classification and should still be a key part of a packets features.

3.2 Data Collection for Packet Classification

A more fined grained approach would be to classify individual packets. The next few studies show that individual packet classification is possible and can yield very good results.

When doing individual packet classification times series features are not useful since you only have one packet. This means that individual packet classification is tough but deep learning offers a solution. As it has already been said deep learning has the ability to learn high dimensional data [16] and therefore it can learn from the raw data of a packet.

Using the ISCX VPN-nonVPN traffic data-set *Lotfollahi et al* classified individual packets into categories such as chat, email and streaming. When dealing with individual packets additional preprocessing is needed. They disguarded any packets with SYN, ACK or FIN tags that had no payload and they removed the Ethernet headers. Using the first 1480 bytes of the IP payload as well as the IP header as input, they obtained an F1 score of greater than 92% for both of their deep learning architectures. The researchers masked the IP addresses because they only used a limited number of hosts and servers [13]. This did not allow the model to use the information provided by the IP addresses which would have caused unreliable results.

In a similar study, [19] used the same data as [13] but disregarded the IP header and got performance of greater than 96%. They also used the ISCX VPN-nonVPN traffic data-set. The data that this study used was unbalanced which could affect the models' performance. The deep learning models would have learned the unequal distribution of classes and skewed its predictions in favor of the

majority class, which could have caused poor performance in the underrepresented classes. The study did not show per class performance which would have been vital information to determine how well the model classified underrepresented classes. To counter this the researches used the same model architectures on a balanced version of the same data-set and got results that were 2 to 3 percent less [19]. This could be due to less data in the training set, which have would affected the model's learning and it would have increased the variance in the test set. Therefore there is insufficient evidence to suggest that the drop in performance was because of the balanced data. The study did manage to build a remarkably accurate model with over 90% on both unbalanced and balanced data which shows that using raw data for individual packet classification together with deep learning can provide good predictive performance.

This section described and compared the different ways the data is extracted from traffic flows and/or packets. How this data is transformed to be used as input to a deep learning model will be talked about in the context of specific deep learning architectures.

4 SUPERVISED LEARNING

Supervised learning is the process where models predict values or classify objects into categories and they do this by learning from a large set of labeled examples. This section will describe the deep learning architectures and compare their performances on classifying flows and on classifying packets.

4.1 Types of Deep Learning Networks

4.1.1 Multi-layer perceptron (MLP).

The multi-layer perceptron is the quintessential deep learning model and forms the basis for many commercial applications. For example convolutional neural networks, which will be described in the next section, is a type of MLP [6]. A MLP is a directed graph that consists of a set of nodes and edges, see figure 1. The nodes are either the input values, if they are at the input layer, or they are the activation functions. The edges are weights which are the parameters that will have to be learned. Every node besides the input nodes takes a weighted sum of all the nodes in the previous layer as input, which is then passed through a non-linear activation function. In a classification task, the outputs of the last layer are fitted with a softmax function and the neuron, which represents a category, with the highest value is taken. It can be seen that these models have a large amount of parameters and therefore training and deploying these models will be very computationally intensive. Due to this complexity and their low accuracy these models have not been widely used in traffic classification [16].

4.1.2 Convolutional Neural Network (CNN).

Convolutional Neural Networks (CNN) are a special kind of network that is useful for processing data with a grid like topology, like time series and image data. The name convolutional neural network indicates that the network employs a mathematical operation called a convolution, which is a linear operation that is used in place of matrix multiplication in the convolutional layers [6]. In a convolutional layer a small set of kernels with a small number of learnable parameters are used on the entire input to produce the next output layer [16], see figure 2. In classification tasks matrix

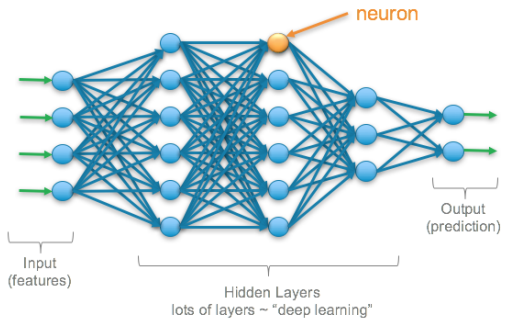

Figure 1: MLP

multiplication, dense layers, are used in the last layers of the model to produce the desired class predictions. The motivation for using a convolutional network comes from three key characteristics of the model namely, sparse interactions, parameter sharing and equivariant representations.

In an MLP a single node is connected to every other node in the previous layer, which creates a large number of dependencies and makes the model hard to train. CNNs use sparse representations which reduces the number of neurons that that a single neuron depends on. This is done by using kernels that are smaller than the input [6]. This drastically reduces the number of parameters in the model.

Another way the number of parameters in the network is reduced is by using parameter sharing. Parameter sharing is achieved by having the same parameters used for more than one function in the model. This means that instead of learning new parameters at each location just one set of parameters is learned over every location [6].

Equivariant representations means that if the input changes then the output changes in the same way. So when processing time series data, this means that the convolution produces a sort of timeline where the features appear in the input [6]. So if an event is moved later in the input then the representation of that input will also be moved later.

CNNs are useful when trying to discover if something exists rather than where something exists. The network achieves this if pooling is introduced. Pooling is where the input is reduced by taking summary statistics, like a maximum, of input values that are close together. This allows the models to learn shift invariant features [6]. CNNs have proved to be powerful models and with parameters sharing as well as sparse interactions it is also efficient. This makes the CNN a good candidate network to be used for traffic classification in community networks.

4.1.3 Recurrent Neural Networks (RNN).

Recurrent Neural Networks are used for processing sequential data where the current state may depend not only on the current input but also previous inputs, for example language modeling and time series predictions [6]. To be able to model these sequences the network contains loops, as shown in figure 3. Like a CNN's ability to scale in size and process grid like topology, RNN's have the ability to scale to longer sequences than otherwise would be possible by

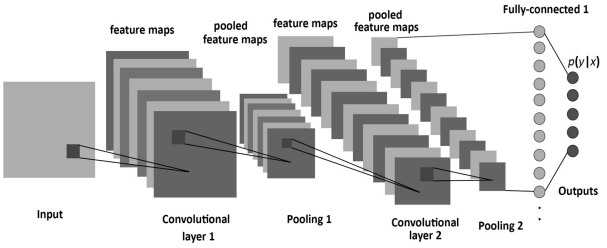

Figure 2: CNN

employing a different model architecture [6]. These networks also take advantage of parameter sharing because they look for information that could occur in multiple locations, and as found with CNNs, parameter sharing drastically reduces the memory requirements. The issue with RNNs is that they cannot learn long term dependencies due to the vanishing and exploding gradients problem that comes from applying the same function over and over again [6]. To solve this problem Long Short-Term Memory (LSTM) networks were created. These are gated RNNs where the gates are used by the network to remember or forget states depending on what time step you are at. This is useful because it allows the gradients to propagate further into the future without exploding or vanishing. In traffic classification approaches LSTMs are used instead of vanilla RNNs because they can learn long term dependencies. They are particularly useful when classifying flows into categories because they can learn the temporal patterns between individual packets. Some studies used a combination of a CNN and an LSTM [12, 20]. There motivation for doing this was that the CNN can extract spatial features and the LSTM can extract the temporal features and this would lead to greater performance. LSTMs are very good at modeling sequential data, they also make use of parameter sharing but they are inherently sequential which makes them very hard to train and run in parallel. Therefore they should take longer to train and run, when compared with other networks, which could use up vital resources in a community network.

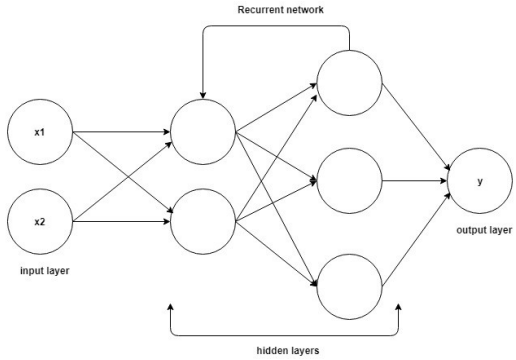

Figure 3: RNN

4.1.4 *Stacked Autoencoder (SAE).*

An autoencoder is a neural network that is usually used in unsupervised learning where the model tries to generate output as close to the input as possible. These networks have been used for dimensionality reduction by making the hidden layers smaller than the input and output layers [16]. A Stacked Autoencoder (SAE) is a stacked network of autoencoders where the input of the next autoencoder is the output of the previous autoencoders hidden layer, this allow the SAE to be trained in a greedy fashion, see figure 4. After the unsupervised learning part is done the encoded layer gets fitted with a softmax function and the network gets fined tuned by a supervised learning approach. SAE have the advantage that they can leverage both the unsupervised and the supervised aspects of the data and they can be efficiently trained which is great for deployment in community networks.

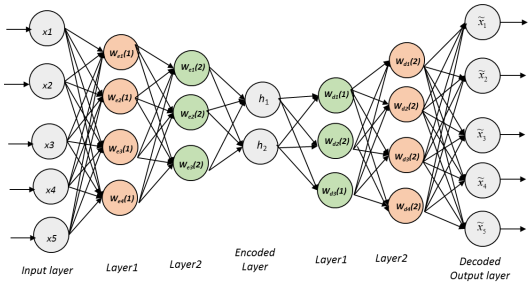

Figure 4: SAE

4.2 Deep learning's performance in flow classification

As spoken about before, when the objects of classification are flows, the data collected changes and therefore the inputs to the models change. Due to the changes in the structure of inputs new model architectures become useful, such as the Long Short-Term Memory (LSTM) network. This deep learning architecture can find long and short term temporal relationships in the data. The 2D-CNN also gets used to find spatial patterns.

The data used to classify flows is generally taken from individual packets. The list of packet data from a flow forms a sequence. Therefore models that are built for sequential data should outperform other model types.

A 1D-CNN and a 2D-CNN were used to classify flows with inputs in the form of ALL and L7 data. In both data representations the first 784 or 1000 bytes of the bidirectional flows were converted either into a 1D vector, for the 1D-CNN, or into a 2D images, for the 2D-CNN. The study did not specify how they transformed the inputs into those vectors. The 1D-CNN had an accuracy of 91.25% and the 2D-CNN had and accuracy of 90% [21]. This is not a surprising result since 1D-CNNs are better suited to processing sequential data [10]. The study also compared the 1D-CNN to the state of the art C4.5 decision tree. The CNN's average recall was 8.1% higher than the average recall of the decision tree [21]. This showed that end-to-end deep learning proved to be better than state-of-the-art machine learning. *Aceto et al* also found that 1D-CNNs outperform 2D-CNNs. As mentioned before LSTMs are used to find long term

dependencies between inputs and it is currently one of the best models to use when processing sequential data. The study showed this to be true by observing the LSTM model outperform a 2D-CNN. A couple of surprising results came out of this study, the SAE had an average accuracy of 36.47% showing that it may not be suited to process sequential data, and the LSTM had the most efficient training time [2]. This is a surprising result because the training and running of an LSTM cannot be done in parallel but most of the other architectures can be run in parallel. As expected the MLP had better results than the SAE but worse results than the three other models.

In the last two studies presented, the 2D-CNN model was not performing as hoped. *Chen et al* took a different approach and encoded the static as well as the dynamic aspects of the flow into an image. They used the first nine packet sizes, inter-arrival times and packet directions as input data. This data is encoded as $I_1, I_2, ..., I_N$, where I_j is either the jth packet size, direction or inter-arrival time. To understand the static and dynamic properties of the flow they estimated the marginal distributions, $P(I_j)$, and the conditional distributions, $P(I_{j+1}|I_j)$, of the flow statistics. To estimate the distributions they used the Reproducing Kernel Hilbert Space embedding. This is an efficient and non-parametric approach to represent these distributions [4]. Since there are three statistics per packet there are six distribution estimates related to each packet. They transformed a list of packets into a six channel image. They then used a 2D-CNN to classify the images into application types and achieved an accuracy of 99.84% which far outperformed the Support Vector Machine (SVM), MLP, Naive Bayes (NB) and decision tree models. They also tested the 2D-CNN on real world data and obtained an accuracy of 88.42% which was a over 11% higher than the next best model [4]. This approach saw the 2D-CNN work really well since there was a well thought out and a well executed way of transforming flow data into an image.

So far studies have used 2D-CNN models to find spatial patterns and they have used LSTM models to find the temporal patterns. There have been approaches that try to combine the two models to try learn both spatial and temporal patterns [12, 20]. *Lopez-Martin et al* used the first 20 packets and 6 features from each packet to make a 20x6 matrix, with the rows as the time dimension and the columns as the feature dimension. They passed that matrix into a 2D-CNN-LSTM network. The output of the CNN part is a 3D matrix, with the extra dimension coming from the number of filters. Since the LSTM accepts a 2D matrix, this matrix was squashed along the feature axis to preserve the time and the filter dimensions which was past into the LSTM. The 2D-CNN-LSTM model obtained an F1 score of 96% which was an improvement over the standard 2D-CNN and LSTM models, which achieve F1 scores of 94.5% and 95.5% respectively. The plain LSTM model seemed to capture the same information as the 2D-CNN-LSTM [12]. This may be due to the CNN compromising the time dimension of the input matrix by passing over it with 2D kernels. Therefore the LSTM part of the model did not have a meaningful sequence to learn.

Another approach to this which did preserve the time aspect of the data was taken by [20]. They took 100 bytes from each of the first 6 packets. They converted the data from each packet into an image until they ended up with 6 images. They used 6 CNN models, one for each packet to extract the spacial features out of the images. Each CNN model had a dense last layer which created a feature vector for each packet, thus preserving the time dimension. These feature vectors were then run into the LSTM part of the model to classify the flow. The CNN-LSTM model had an accuracy of 99.89% and outperformed the plain CNN network [20]. By splitting up the data into 6 images and processing them separately they allowed the output of the CNNs to be stacked in a manner which would preserve the order of the packets and therefore the time. This is a much more suitable input for the LSTM part of the model when you compare it with the previous study.

4.3 Deep learning's performance in packet classification

One of the most successful deep learning architectures is the Convolutional Neural Network (CNN). This model has lead to incredible performances in image recognition and object detection. Normally CNNs are used on 2D images but they can be adapted to be used on 1D vectors. *Lotfollahi et al* used a 1D-CNN to classify individual packets into classes, on the ISCX VPN-nonVPN data-set, and obtained an F1 score of 98% in application classification, and an F1 score of 93% in traffic categorization. The study also used a Stacked Auto-encoder (SAE) to classify internet traffic [13]. Auto-encoders are used as an unsupervised learning algorithm to try an generate output that is as close to the input as possible. This allows the model to learn a more comprehensive feature set which can be used to train supervised learning models [8]. The SAE was trained and then fitted with a soft-max layer to classify the traffic. The model performed slightly worse than the 1D-CNN and achieved F1 scores of 95% and 92% for application classification and traffic characterization respectively [13]. A similar done study by [19] on application classification saw the CNN and SAE achieve F1 scores of 98.4% and 98.8% respectively. They also compared that with a Multi-Layer Perceptron (MLP) which had an F1 score of 96.5%. The MLP was a smaller model with only two hidden layers and six neurons each, which probably caused the reduced performance. In both of these studies the input to the models was a 1D vector that corresponded to the first 1480 bytes of the IP payload data, but in [13] they also added the IP header.

4.4 Comparison of the computational efficiency of the deep learning architectures

In community networks they try to lower the cost of entry by using cheaper resources [3]. Deep learning models are typically large in terms of the number of parameters that they have. To optimize these parameters you need to have a large amount of data and good computing resources or the models will not have sufficient accuracy and may take too long to train to be feasible as a solution. This section will compare key characteristics of deep learning models to try find out which ones might be better suited to a low resource environment.

As discussed before MLPs are the quintessential deep learning architecture [6] but they are very complex. They have a large number of

parameters which will use up a lot of memory and the dense nature of the network creates many dependencies per neuron which will make the training time longer. On the other hand, CNN models use parameter sharing which will decrease the number of parameters in the model [6] and free up memory. They also use sparse interactions [6] which reduces the number of dependencies and allows the model to be trained easier and to be run in parallel. CNNs have also show better classification accuracy in the above studies. LSTMs are another model type that makes use of parameter sharing, but this model cannot be run in parallel because it is inherently sequential. One of the criticisms of LSTM models is that they take a long time to train. However, this model seems to perform better than the CNNs on prediction accuracy, therefore a trade-off between time and accuracy will have to be made. SAEs can be trained in a greedy layer-wise fashion [16] which would create an efficient training time but SAEs do have to be trained twice. The training time will depend on how well the unsupervised step initialized the weights for the supervised learning algorithm.

The nature of deep learning is that you have to make a trade-off between the accuracy of the network and computational efficiency. This section just gives an overview of what characteristics of the networks should be taken into account when running them in a low resource environment.

5 UNSUPERVISED, SEMI-SUPERVISED, GENERATIVE LEARNING

In the studies that have been looked at so far most of the models were trained and evaluated on unbalanced data-sets. The data-sets have a majority class, which will contain most of the training examples. The machine learning models' predictions will then be skewed towards the majority class, and the evaluation of these models will not reflect true results. Machine learning models work best when classes are balanced [1]. To solve this problem *Vu et al* used an Auxiliary Classifier Generative Adversarial Network (AC-GAN) to over sample minor classes and under sample major classes and create a balanced training data-set. This model has the ability, unlike regular GANs, to generate samples from specific classes [18]. They used a RF, SVM and a decision tree to perform the classification. After augmenting the training data with synthetic data, the models increased their accuracy, F1 and AUC scores. The AC-GAN method proved to perform better than the other generative models and had a faster training time [18]. This showed that using deep generative models to balance data-sets before training can increase the performance of the models and it can do so in the most efficient way.

In traffic analysis there is a large amount of unlabeled data and labeled data is expensive and hard to come by. In building deep learning networks we want to be able to take advantage of all the data available. *Liu & Rezaei* developed a semi-supervised learning approach that used the unlabeled data to increase their prediction accuracy. They trained a 1D-CNN, on the unlabeled data, to predict the statistical features of a flow by using packet length, direction and inter-arrival times as inputs. Using the small amount of labeled data they had, they then used supervised learning to train a MLP attached to the end of the CNN to predict class labels for the flows. The accuracy of the MLP without the pre-training on the labeled data was 40.37% and the accuracy with pre-training was 84.82% [15]. This is a remarkable increase in accuracy and proves that taking a semi-supervised learning approach, when only a small amount of labels are available, will dramatically increase a networks predictive capabilities.

If only unlabeled data is available then traffic classification is still possible with unsupervised deep learning. A neural auto-encoder and statistical flow features were used by *Höchst et al* to achieve an F1 score of 76%. They took the encoder from the auto-encoder and applied a soft-max layer to it, which split the flows into clusters. The clusters were labeled based on their characteristics. Examples of the cluster names are upload, download and browsing [7]. The F1 score shows that even with unlabeled data traffic classification can give network managers a high level understanding of the network.

6 DISCUSSION

One of the issues with the study of traffic classification is that there is no widely used data-set where models can be evaluated [16]. Studies tend to use their own data-sets or publicly available data-sets that are designed for specific tasks. Due to this it becomes tough to compare results from different studies and most of the conclusions drawn, will be from intra-study comparisons.

The rise of data-sets such as MNIST [11] and the CI-FAR data-sets [9] have allowed areas such as computer vision and object detection to thrive. A possible way to increase the reliability of results and the performance of models would be to identify key areas where traffic classification can be applied, such as intrusion detection and application classification, and to create large data-sets specific for these tasks, which can then be used as benchmarks. This will allow for more robust conclusions to be drawn from studies.

When using flows as the object of classification you can either use bidirectional flows or unidirectional flows. It has been found that using bidirectional flows increases the accuracy of a models predictions, which can be attributed to the extra interaction information [21]. There are also multiple ways to extract raw information from the flows, such as using ALL or L7 data. A key finding of this review was that using data from all protocol layers (ALL) performed better than just using the data found in the application layers (L7) [2]. They also found that using raw data was better than using hand picked time series features. These findings have shown that if you increase the raw information available to the deep learning models they will yield greater prediction accuracy.

One of the issues with raw data is that it can have a high dimensionality. To be able to classify this data accurately the deep learning models will need to be larger but these models have to be deployed in community networks which limits their size. Therefore it may not be feasible to use raw data for traffic classification in community networks. To reduce the dimensions of the data, packet or flow statistics can be used. It must be noted that using packet statistics

instead of summary flow statistics allowed models to learn the dynamic structure of the flows, which lead to an increase in predictive capabilities [4].

Packet sizes, directions and inter-arrival times tend to be the features used in time series data. They have proved to be valuable and have allowed models to perform well, but there has been minimal analysis on these features in the context of deep learning. *Lopez-Martin et al* did some feature analysis and found that port numbers were a really important feature in their study. They also found that the inter-arrival time was providing some information that was not well aligned with the port numbers. When the inter-arrival information was used in conjunction with the port numbers there was a lower accuracy then when the inter-arrival time was removed [12]. More analysis like this could lead us to a better understanding of the features and how they interact with each other and it will allow us to design better feature variables that will increase the quality of the classifications.

Flow classification has the benefit of being able to learn time related features but if you want real time fast classification then packet classification is the way to go. In packet classification you do not have to wait to collect multiple packets before you can perform a classification task. Two studies [13] and [19] showed that packet classification is not only possible but can yield impressive performance because of deep learning. This may be a better option in a community network because the data needed per object of classification is lower which means that the deep learning models can be smaller which will decrease the amount of computational power needed to run them.

CNN, LSTM and SAE models generally have performed better than the MLP model [4, 19] and these models are more computationally efficient. SAEs were shown to do well on packet classification [13, 19] but they struggled when classifying flows [2], this may be because the model struggled to detect patterns in the sequence of packets. The two networks that showed the most promise were the CNN and LSTM networks. In flow classification the 1D-CNN and the LSTM networks were able to capture the time series features better than the 2D-CNN and therefore yielded greater prediction accuracy [2, 19]. However, the 2D-CNN network should not be discarded because when the dynamic aspects of the flow were encoded into the image the network was able to achieve great accuracy [4]. The 1D-CNN and the LSTM are more suited to sequential data and thus perform well on flow classification. The LSTM is likely to have better accuracy but it is more computationally expensive to run which will need to be taken into account when running it on a community network. A direct comparison between these two model types would provide valuable information.

The 1D-CNN model had a high predictive accuracy when classifying individual packets [13, 19]. This gives some evidence for a sequential structure to the data. Therefore LSTM models which are more suited to processing this data should be tested and compared with the 1D-CNN in this context as well.

7 CONCLUSION

Deep learning has shown to be better at classifying internet traffic when compared with regular machine learning models. They can limit the need for pre-processing by using raw data as input and they are able to accurately classify the high dimensional and complex data that network traffic produces. Deep learning can be a valuable tool that community network managers use to better understand how their limited resources can be better utilized. This will lead to increased QoS in community networks and will allow more people to have access to affordable internet.

REFERENCES

[1] Razan Abdulhammed, Miad Faezipour, Abdelshakour Abuzneid, and Arafat Abu-Mallouh. 2018. Deep and machine learning approaches for anomaly-based intrusion detection of imbalanced network traffic. *IEEE sensors letters* 3, 1 (2018), 1–4.

[2] Giuseppe Aceto, Domenico Ciuonzo, Antonio Montieri, and Antonio Pescapé. 2018. Mobile encrypted traffic classification using deep learning. In *2018 Network Traffic Measurement and Analysis Conference (TMA)*. IEEE, 1–8.

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

[4] 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.

[5] Gerard Draper-Gil, Arash Habibi Lashkari, Mohammad Saiful Islam Mamun, and Ali A Ghorbani. 2016. Characterization of encrypted and vpn traffic using time-related. In *Proceedings of the 2nd international conference on information systems security and privacy (ICISSP)*. 407–414.

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

[7] Jonas Höchst, Lars Baumgärtner, Matthias Hollick, and Bernd Freisleben. 2017. Unsupervised traffic flow classification using a neural autoencoder. In *2017 IEEE 42nd Conference on Local Computer Networks (LCN)*. IEEE, 523–526.

[8] Jonnalagadda. 2018. *Sparse, Stacked and Variational Autoencoder*. https://medium.com/@venkatakrishna.jonnalagadda/sparse-stacked-and-variational-autoencoder-efe5bfe73b64

[9] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

[10] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature* 521, 7553 (2015), 436–444.

[11] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[12] 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.

[13] Mohammad Lotfollahi, Ramin Shirali Hossein Zade, Mahdi Jafari Siavoshani, and Mohammdsadegh Saberian. 2017. Deep Packet: A Novel Approach For Encrypted Traffic Classification Using Deep Learning. arXiv:cs.LG/1709.02656

[14] Leandro Navarro, Pau Escrich, Roger Baig, and Axel Neumann. 2012. CommunityLab: Overview and invitation to the research community. In *2012 IEEE 12th International Conference on Peer-to-Peer Computing (P2P)*. IEEE, 71–72.

[15] Shahbaz Rezaei and Xin Liu. 2018. How to achieve high classification accuracy with just a few labels: A semi-supervised approach using sampled packets. *arXiv preprint arXiv:1812.09761* (2018).

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

[17] 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.

[18] 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.

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

[20] 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.

[21] 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.