Soft Computing (2020) 24:1999–2012
https://doi.org/10.1007/s00500-019-04030-2

METHODOLOGIES AND APPLICATION

Deep packet: a novel approach for encrypted traffic classification using deep learning

Mohammad Lotfollahi[1] · **Mahdi Jafari Siavoshani**[1] · **Ramin Shirali Hossein Zade**[1] · **Mohammdsadegh Saberian**[1]

Published online: 13 May 2019

Abstract
Network traffic classification has become more important with the rapid growth of Internet and online applications. Numerous studies have been done on this topic which have led to many different approaches. Most of these approaches use predefined features extracted by an expert in order to classify network traffic. In contrast, in this study, we propose a *deep learning*-based approach which integrates both feature extraction and classification phases into one system. Our proposed scheme, called "Deep Packet," can handle both *traffic characterization* in which the network traffic is categorized into major classes (e.g., FTP and P2P) and *application identification* in which identifying end-user applications (e.g., BitTorrent and Skype) is desired. Contrary to most of the current methods, Deep Packet can identify encrypted traffic and also distinguishes between VPN and non-VPN network traffic. The Deep Packet framework employs two deep neural network structures, namely stacked autoencoder (SAE) and convolution neural network (CNN) in order to classify network traffic. Our experiments show that the best result is achieved when Deep Packet uses CNN as its classification model where it achieves recall of 0.98 in application identification task and 0.94 in traffic categorization task. To the best of our knowledge, Deep Packet outperforms all of the proposed classification methods on UNB ISCX VPN-nonVPN dataset.

Keywords Network traffic classification · Application identification · Traffic characterization · Deep learning · Convolutional neural networks · Stacked autoencoder · Deep Packet

1 Introduction

Traffic classification is an important task in modern communication networks (Bagui et al. 2017). Due to the rapid growth of high-throughput traffic demands, to properly manage network resources, it is vital to recognize different types of applications utilizing network resources. Consequently, accurate traffic classification has become one of the prerequisites for advanced network management tasks such as providing appropriate Quality-of-Service (QoS), anomaly detection, pricing, etc. Traffic classification has attracted a lot of interests in both academia and industrial activities related to network management (e.g., see Dainotti et al. 2012; Finsterbusch et al. 2014; Velan et al. 2015) and the references therein).

As an example of the importance of network traffic classification, one can think of the asymmetric architecture of today's network access links, which has been designed based on the assumption that clients download more than what they upload. However, the pervasiveness of symmetric-demand applications [such as peer-to-peer (P2P) applications, voice over IP (VoIP) and video call] has changed the clients' demands to deviate from the assumption mentioned earlier. Thus, to provide a satisfactory experience for the clients, an application-level knowledge is required to allocate adequate resources to such applications.

The emergence of new applications as well as interactions between various components on the Internet has dramatically increased the complexity and diversity of this network which makes the traffic classification a difficult problem per se. In the following, we discuss in details some of the most critical challenges of network traffic classification.

First, the increasing demand for user's privacy and data encryption has tremendously raised the amount of encrypted traffic in today's Internet (Velan et al. 2015). Encryption procedure turns the original data into a pseudo-random-like

Communicated by V. Loia.

✉ Mahdi Jafari Siavoshani
mjafari@sharif.edu

[1] Sharif University of Technology, Tehran, Iran

format with the aim to make it hard to decrypt. As a result, it causes the encrypted data scarcely contain any discriminative patterns to identify network traffic. Therefore, accurate classification of encrypted traffic has become a real challenge in modern networks (Dainotti et al. 2012).

It is also worth mentioning that many of the proposed network traffic classification approaches, such as payload inspection as well as machine learning-based and statistical-based methods, require patterns or features to be extracted by experts. This process is prone to error, time-consuming and costly.

Finally, many of the Internet service providers (ISPs) block P2P file sharing applications because of their high bandwidth consumption and copyright issues (Lv et al. 2014). Hence, to circumvent this problem, these applications use protocol embedding and obfuscation techniques to bypass traffic control systems (Alshammari and Zincir-Heywood 2011). The identification of this kind of applications is one of the most challenging tasks in network traffic classification.

There have been abundant studies on the network traffic classification subject, e.g., Kohout and Pevný (2018), Perera et al. (2017), Gil et al. (2016) and Moore and Papagiannaki (2005). However, most of them have focused on classifying a protocol family, also known as *traffic characterization* (e.g., streaming, chat, P2P, etc.), instead of identifying a single application, which is known as *application identification* (e.g., Spotify, Hangouts, BitTorrent, etc.) (Khalife et al. 2014). In contrast, this work proposes a method, i.e., Deep Packet, based on the ideas recently developed in the machine learning community, namely deep learning (Bengio 2009; LeCun et al. 2015), to both characterize and identify the network traffic. The benefits of our proposed method, which make it superior to other classification schemes, are stated as follows:

- In Deep Packet, there is no need for an expert to extract features related to network traffic. In light of this approach, the cumbersome step of finding and extracting distinguishing features has been omitted.
- Deep Packet can identify traffic at both granular levels (application identification and traffic characterization) with state-of-the-art results compared to the other works conducted on similar dataset (Gil et al. 2016; Yamansavascilar et al. 2017).
- Deep Packet can accurately classify one of the hardest class of applications, known to be P2P (Khalife et al. 2014). This kind of applications routinely uses advanced port obfuscation techniques, embedding their information in well-known protocols' packets and using random ports to circumvent ISPs' controlling processes.

The rest of paper is organized as follows. In Sect. 2, we review some of the most important and recent studies on network traffic classification. In Sect. 3, we present the essential background on deep learning which is necessary to our work. Section 4 presents our proposed method, i.e., Deep Packet. The results of the proposed scheme on network application identification and traffic characterization tasks are described in Sect. 5. In Sect. 6, we provide further discussion on experimental results. Section 7 discusses future work and possible direction for further inspection. Finally, we conclude the paper in Sect. 8.

2 Related works

In this section, we provide an overview of the most important network traffic classification methods. In particular, we can categorize these approaches into three main categories as follows: (I) port-based methods, (II) payload inspection techniques and (III) statistical and machine learning approaches. Here is a brief review of the most important and recent studies regarding each of the approaches mentioned above.

Port-based approach Traffic classification via port number is the oldest and the most well-known method for this task (Dainotti et al. 2012). Port-based classifiers use the information in the TCP/UDP headers of the packets to extract the port number which is assumed to be associated with a particular application. After the extraction of the port number, it is compared with the assigned IANA TCP/UDP port numbers for traffic classification. The extraction is an easy procedure, and port numbers will not be affected by encryption schemes. Because of the fast extraction process, this method is often used in firewalls and access control lists (ACL) (Qi et al. 2009). Port-based classification is known to be among the simplest and fastest method for network traffic identification. However, the pervasiveness of port obfuscation, network address translation (NAT), port forwarding, protocol embedding and random ports assignments have significantly reduced the accuracy of this approach. According to Moore and Papagiannaki (2005) and Madhukar and Williamson (2006), only 30% to 70% of the current Internet traffic can be classified using port-based classification methods. For these reasons, more complex traffic classification methods are needed to classify modern network traffic.

Payload inspection techniques These techniques are based on the analysis of information available in the application layer payload of packets (Khalife et al. 2014). Most of the payload inspection methods, also known as deep packet inspection (DPI), use predefined patterns like regular expressions as signatures for each protocol (e.g., see Yeganeh et al. 2012; Sen et al. 2004). The derived patterns are then used to distinguish protocols form each other. The need for updating patterns whenever a new protocol is released, and user privacy issues are among the most important drawbacks of this

approach. Sherry et al. proposed a new DPI system that can inspect encrypted payload without decryption, thus solved the user privacy issue, but it can only process HTTP Secure (HTTPS) traffic (Sherry et al. 2015).

Statistical and machine learning approach Some of these methods, mainly known as statistical methods, have a biased assumption that the underlying traffic for each application has some statistical features which are almost unique to each application. Each statistical method uses its own functions and statistics. Crotti et al. (2007) proposed protocol fingerprints based on the probability density function (PDF) of packets inter-arrival time and normalized thresholds. They achieved up to 91% accuracy for a group of protocols such as HTTP, Post Office Protocol 3 (POP3) and Simple Mail Transfer Protocol (SMTP). In a similar work, Wang and Parish (2010) have considered PDF of the packet size. Their scheme was able to identify a broader range of protocols including file transfer protocol (FTP), Internet Message Access Protocol (IMAP), SSH, and TELNET with accuracy up to 87%.

A vast number of machine learning approaches have been published to classify traffic. Auld et al. proposed a Bayesian neural network that was trained to classify most well-known P2P protocols including Kazaa, BitTorrent, GnuTella, and achieved 99% accuracy (Auld et al. 2007). Moore et al. achieved 96% of accuracy on the same set of applications using a Naive Bayes classifier and a kernel density estimator (Moore and Zuev 2005). Artificial neural network (ANN) approaches were proposed for traffic identification (e.g., see Sun et al. 2010; Ting et al. 2010). Moreover, it was shown in Ting et al. (2010) that the ANN approach can outperform Naive Bayes methods. Two of the most important papers that have been published on "ISCX VPN-nonVPN" traffic dataset are based on machine learning methods. Gil et al. (2016) used time-related features such as the duration of the flow, flow bytes per second, forward and backward inter-arrival time, etc. to characterize the network traffic using k-nearest neighbor (k-NN) and C4.5 decision tree algorithms. They achieved approximately 92% recall, characterizing six major classes of traffic including Web browsing, email, chat, streaming, file transfer and VoIP using the C4.5 algorithm. They also achieved approximately 88% recall using the C4.5 algorithm on the same dataset which is tunneled through VPN. Yamansavascilar et al. manually selected 111 flow features described in Moore et al. (2013) and achieved 94% of accuracy for 14 class of applications using k-NN algorithm (Yamansavascilar et al. 2017). The main drawback of all these approaches is that the feature extraction and feature selection phases are essentially done with the assistance of an expert. Hence, it makes these approaches time-consuming, expensive and prone to human mistakes. Moreover, note that for the case of using k-NN classifiers, as suggested by Yamansavascilar et al. (2017), it is known that, when used for prediction, the execution time of this algorithm is a major concern.

To the best of our knowledge, prior to our work, only one study based on deep learning ideas has been reported by Wangc Wang (2015). They used stacked autoencoders (SAE) to classify some network traffic for a large family of protocols like HTTP, SMTP, etc. However, in their technical report, they did not mention the dataset they used. Moreover, the methodology of their scheme, the details of their implementation, and the proper report of their result is missing.

3 Background on deep neural networks

Neural networks (NNs) are computing systems made up of some simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs (Caudill 1987). In practice, these networks are typically constructed from a vast number of building blocks called *neuron* where they are connected via some links to each other. These links are called connections, and to each of them, a weight value is associated. During the training procedure, the NN is fed with a large number of data samples. The widely used learning algorithm to train such networks (called *backpropagation*) adjusts the weights to achieve the desired output from the NN. The deep learning framework can be considered as a particular kind of NNs with many (hidden) layers. Nowadays, with the rapid growth of computational power and the availability of graphical processing units (GPUs), training deep NNs have become more plausible. Therefore, the researchers from different scientific fields consider using deep learning framework in their respective area of research, e.g., see Hinton et al. (2012), Lotfollahi et al. (2018) and Socher et al. (2013). In the following, we will briefly review two of the most important deep neural networks that have been used in our proposed scheme for network traffic classification, namely autoencoders and convolutional neural networks.

3.1 Autoencoder

An autoencoder NN is an unsupervised learning framework that aims to reconstruct the input at the output while minimizing the reconstruction error (i.e., according to some criteria). Consider a training set $\{x^1, x^2, \ldots, x^n\}$ where for each training data we have $x^i \in \mathbb{R}^n$. The autoencoder's objective is defined to be $y^i = x^i$ for $i \in \{1, 2, \ldots, n\}$, i.e., the output of the network will be equal to its input. Considering this objective function, the autoencoder tries to learn a compressed representation of the dataset, i.e., it approximately learns the identity function $F_{\boldsymbol{W},\boldsymbol{b}}(x) \simeq x$, where $\boldsymbol{W}$ and $\boldsymbol{b}$ are the whole network weights and biases vectors. General

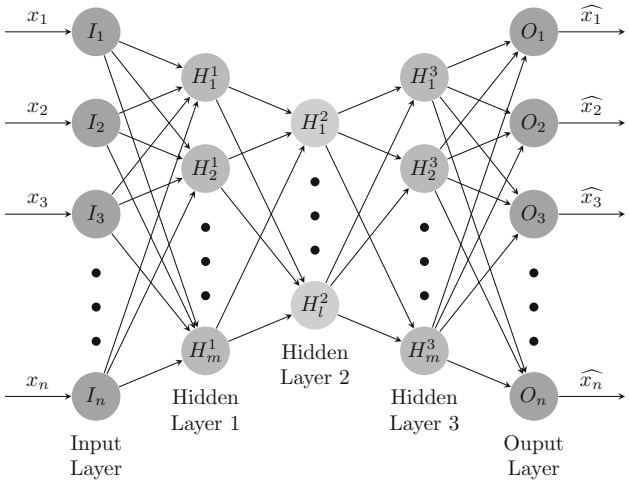


Fig. 1 The general structure of an autoencoder

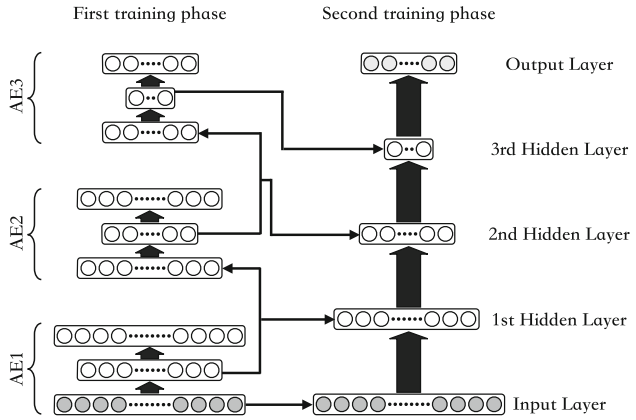


Fig. 2 Greedy layer-wise approach for training an stacked autoencoder

form of an autoencoder's loss function is shown in (1), as follows

$$\mathcal{L}(\boldsymbol{W}, \boldsymbol{b}) = \left\| x - F_{\boldsymbol{W},\boldsymbol{b}}(x) \right\|^2 . \tag{1}$$

Figure 1 shows a typical autoencoder with n inputs and outputs. The autoencoder is mainly used as an unsupervised technique for automatic feature extraction. More precisely, the output of the encoder part is considered as a high-level set of discriminative features for the classification task.

In practice, to obtain a better performance, a more complex architecture and training procedure, called stacked autoencoder (SAE), is proposed (Vincent et al. 2008). This scheme suggests to stack up several autoencoders in a manner that output of each one is the input of the successive layer which itself is an autoencoder. The training procedure of a stacked autoencoder is done in a greedy layer-wise fashion (Bengio et al. 2007). First, this method trains each layer of the network while freezing the weights of other layers. After training all the layers, to have more accurate results, fine-tuning is applied to the whole NN. At the fine-tuning phase, the backpropagation algorithm is used to adjust all layers' weights. Moreover, for the classification task, an extra softmax layer can be applied to the final layer. Figure 2 depicts the training procedure of a stacked autoencoder.

3.2 Convolutional neural network

The convolutional neural networks (CNN) are another types of deep learning models in which feature extraction from the input data is done using layers comprised of convolutional operations (i.e., convolutional filters). The construction of convolutional networks is inspired by the visual structure of living organisms (Hubel and Wiesel 1968). Basic building block underneath a CNN is a convolutional layer described as follows. Consider a convolutional layer with $N \times N$ square neuron layer as input and a filter ω of size $m \times m$. The output of this layer z^l is of size $(N - m + 1) \times (N - m + 1)$ and is computed as follows

$$z_{ij}^{\ell} = f\left(\sum_{a=0}^{m-1} \sum_{b=0}^{m-1} \omega_{ab} z_{(i+a)(j+b)}^{\ell-1} \right). \tag{2}$$

As it is demonstrated in (2), a nonlinear function f such as rectified linear unit (ReLU) is applied to the convolution output to learn more complex features from the data. In some applications, a pooling layer (e.g., max pooling) is also applied. The main motivation of employing a pooling layer is to aggregate multiple low-level features in a neighborhood to obtain local invariance. Moreover, by reducing the output size, it helps to reduce the computation cost of the network in train and test phase.

CNNs have been successfully applied to different fields including natural language processing (dos Santos and Gatti 2014), computational biology (Alipanahi et al. 2015), and machine vision (Simonyan and Zisserman 2014). One of the most interesting applications of CNNs is in face recognition (Lee et al. 2009), where consecutive convolutional layers are used to extract features from each image. It is observed that the extracted features in shallower layers are simple concepts like edges and curves. On the contrary, features in deeper layers of networks are more abstract than the ones in shallower layers (Yosinski et al. 2015). However, it is worth mentioning that visualizing the extracted features in the middle layers of a network does not always lead to meaningful concepts like what has been observed in the face recognition task. For example in one-dimensional CNN (1D-CNN) which we use to classify network traffic, the feature vectors extracted in shallow layers are just some real numbers which make no sense at all for a human observer.

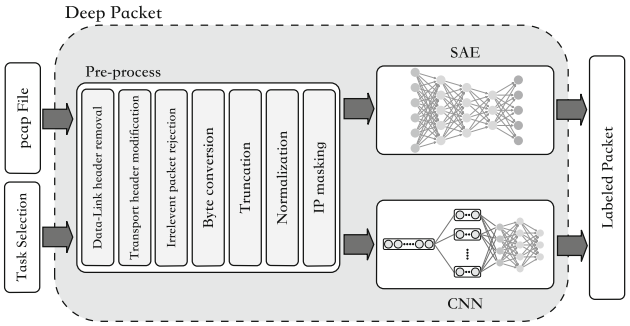


Fig. 3 General illustration of Deep Packet toolkit

We believe 1D-CNNs are an ideal choice for the network traffic classification task. This is true since 1D-CNNs can capture spatial dependencies between adjacent bytes in network packets that leads to find discriminative patterns for every class of protocols/applications, and consequently, an accurate classification of the traffic. Our classification results confirm this claim and prove that CNNs performs very well in feature extraction of network traffic data.

4 Methodology

In this work, we develop a framework, called Deep Packet, that comprises two deep learning methods, namely convolutional NN and stacked autoencoder NN, for both "application identification" and "traffic characterization" tasks. Before training the NNs, we have to prepare the network traffic data so that it can be fed into NNs properly. To this end, we perform a pre-processing phase on the dataset. Figure 3 demonstrates the general structure of Deep Packet. At the test phase, a pre-trained neural network corresponding to the type of classification, application identification or traffic characterization, is used to predict the class of traffic the packet belongs to. The dataset, implementation and design details of the pre-processing phase and the architecture of proposed NNs will be explained in the following.

4.1 Dataset

For this work, we use "ISCX VPN-nonVPN" traffic dataset, that consists of captured traffic of different applications in pcap format files (Gil et al. 2016). In this dataset, the captured packets are separated into different pcap files labeled according to the application produced the packets (e.g., Skype, and Hangouts, etc.) and the particular activity the application was engaged during the capture session (e.g., voice call, chat, file transfer, or video call). For more details on the captured traffic and the traffic generation process, refer to Gil et al. (2016).

The dataset also contains packets captured over Virtual Private Network (VPN) sessions. A VPN is a private overlay network among distributed sites which operates by tunneling traffic over public communication networks (e.g., the Internet). Tunneling IP packets, guaranteeing secure remote access to servers and services, is the most prominent aspect of VPNs (Chowdhury and Boutaba 2010). Similar to regular (non-VPN) traffic, VPN traffic is captured for different applications, such as Skype, while performing different activities, like voice call, video call, and chat.

Furthermore, this dataset contains captured traffic of Tor software. This traffic is presumably generated while using Tor browser, and it has labels such as Twitter, Google, Facebook, etc. Tor is a free, open source software developed for anonymous communications. Tor forwards users' traffic through its own free, worldwide, overlay network which consists of volunteer-operated servers. Tor was proposed to protect users against Internet surveillance known as "traffic analysis." To create a private network pathway, Tor builds a circuit of encrypted connections through relays on the network in a way that no individual relay ever knows the complete path that a data packet has taken (Dingledine et al. 2004). Finally, Tor uses complex port obfuscation algorithm to improve privacy and anonymity.

4.2 Pre-processing

The "ISCX VPN-nonVPN" dataset is captured at the data-link layer. Thus, it includes the Ethernet header. The data-link header contains information regarding the physical link, such as Media Access Control (MAC) address, which is essential for forwarding the frames in the network, but it is uninformative for either the application identification or traffic characterization tasks. Hence, in the pre-processing phase, the Ethernet header is removed first. Transport layer segments, specifically Transmission Control Protocol (TCP) or User Datagram Protocol (UDP), vary in header length. The former typically bears a header of 20 bytes length, while the latter has an 8 bytes header. To make the transport layer segments uniform, we inject zeros to the end of UDP segment's headers to make them equal length with TCP headers. The packets are then transformed from bits to bytes which helps to reduce the input size of the NNs.

Since the dataset is captured in a real-world emulation, it contains some irrelevant packets which are not of our interest and should be discarded. In particular, the dataset includes some TCP segments with either SYN, ACK, or FIN flags set to one and containing no payload. These segments are needed for three-way handshaking procedure while establishing a connection or finishing one, but they carry no information regarding the application generated them, thus can be safely discarded. Furthermore, there are some Domain Name Service (DNS) segments in the dataset. These segments are used for hostname resolution, namely translating URLs to IP addresses. These segments are not relevant to either appli-

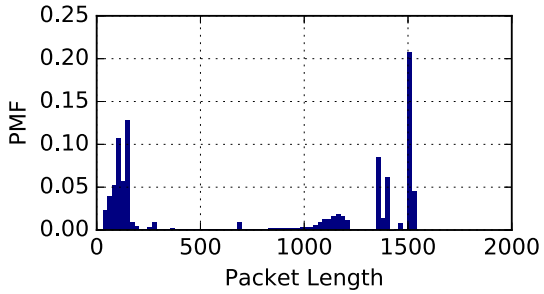


Fig. 4 Empirical probability mass function of the packet length in ISCX VPN-nonVPN traffic dataset

cation identification or traffic characterization, hence can be omitted from the dataset.

Figure 4 illustrates the histogram (empirical distribution) of packet length for the dataset. As the histogram shows, packet length varies a lot through the dataset, while employing NNs necessitates using a fixed-size input. Hence, truncation at a fixed length or zero-padding is required inevitably. To find the fixed length for truncation, we inspected the packets length's statistics. Our investigation revealed that approximately 96% of packets have a payload length of less than 1480 bytes. This observation is not far from our expectation, as most of the computer networks are constrained by Maximum Transmission Unit (MTU) size of 1500 bytes. Hence, we keep the IP header and the first 1480 bytes of each IP packet which results in a 1500 bytes vector as the input for our proposed NNs. Packets with IP payload less than 1480 bytes are zero-padded at the end. To obtain a better performance, all the packet bytes are divided by 255, the maximum value for a byte, so that all the input values are in the range [0, 1].

Furthermore, since there is the possibility that the NN attempts to learn classifying the packets using their IP addresses, as the dataset is captured using a limited number of hosts and servers, we decided to prevent this over-fitting by masking the IP addresses in the IP header. In this matter, we assure that the NN is not using irrelevant features to perform classification. All of the pre-processing steps mentioned above take place when the user loads a pcap file into Deep Packet toolkit.

4.2.1 Labeling dataset

As mentioned before in Sect. 4.1, the dataset's pcap files are labeled according to the applications and activities they were engaged in. However, for application identification and traffic characterization tasks, we need to redefine the labels, concerning each task. For application identification, all pcap files labeled as a particular application which were collected during a non-VPN session are aggregated into a single file. This leads to 17 distinct labels shown in Table 1a. Also for traffic characterization, we aggregated the captured traffic of different applications involved in the same activity, taking into account the VPN or non-VPN condition, into a single pcap file. This leads to a 12-class dataset, as shown in Table 1b. By observing Table 1, one would instantly notice that the dataset is significantly imbalanced and the number of samples varies remarkably among different classes. It is known that such an imbalance in the training data leads to a reduced classification performance. Sampling is a simple yet powerful technique to overcome this problem (Longadge and Dongre 2013). Hence, to train the proposed NNs, using the undersampling method, we randomly remove the major classes'

Table 1 Number of samples (packets) in each class for (a) application identification, and (b) traffic characterization

Application	Size (K)
(a)	
AIM chat	5
Email	28
Facebook	2502
FTPS	7872
Gmail	12
Hangouts	3766
ICQ	7
Netflix	299
SCP	448
SFTP	418
Skype	2872
Spotify	40
Torrent	70
Tor	202
Voipbuster	842
Vimeo	146
YouTube	251
Class name	**Size (K)**
(b)	
Chat	82
Email	28
File transfer	210
Streaming	1139
Torrent	70
VoIP	5120
VPN: Chat	50
VPN: File transfer	251
VPN: Email	13
VPN: Streaming	479
VPN: Torrent	269
VPN: VoIP	753

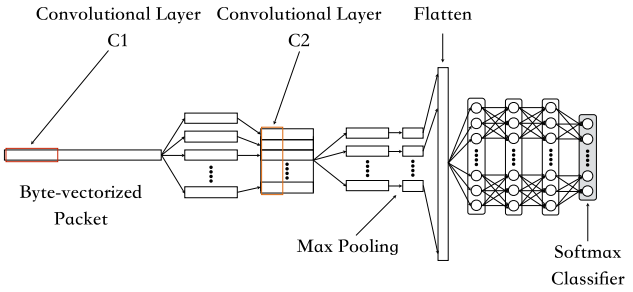


Fig. 5 A minimal illustration of the proposed one-dimensional CNN architecture

samples (classes having more samples) until the classes are relatively balanced.

4.3 Architectures

In the following, we explain our two proposed architectures used in the Deep Packet toolkit.

The proposed SAE architecture consists of five fully connected layers, stacked on top of each other which made up of 400, 300, 200, 100 and 50 neurons, respectively. To prevent the over-fitting problem, after each layer the dropout technique with 0.05 dropout rate is employed. In this technique, during the training phase, some of the neurons are set to zero randomly. Hence, at each iteration, there is a random set of active neurons. For the application identification and traffic characterization tasks, at the final layer of the proposed SAE, a softmax classifier with 17 and 12 neurons is added, respectively.

A minimal illustration of the second proposed scheme, based on one-dimensional (1D) CNN, is depicted in Fig. 5. We used a grid search on a subspace of the hyper-parameters space to select the ones which results in the best performance. This procedure is discussed in detail in Sect. 5. Our final proposed model consists of two consecutive convolutional layers, followed by a pooling layer. Then, the two-dimensional tensor is squashed into a one-dimensional vector and fed into a three-layered network of fully connected neurons which also employ dropout technique to avoid over-fitting. Finally, a softmax classifier is applied for the classification task, similar to the SAE architecture. The best values found for the hyper-parameters are shown in Table 2. The detailed architecture of all the proposed models for application identification and traffic characterization tasks can be found in "Appendix A".

5 Experimental results

To implement our proposed NNs, we have used Keras library (Chollet et al 2017), with Tensorflow (Abadi et al. 2015) as its backend. Each of the proposed models was trained and evaluated against the independent test set that was extracted from the dataset. We randomly split the dataset into three separate sets. The first one which includes 64% of samples is used for training and adjusting weights and biases. The second part containing 16% of samples is used for validation during the training phase, and finally the third set made up of 20% of data points is used for testing the model. Additionally, to avoid the over-fitting problem, we have used *early stopping* technique (Prechelt 1998). This technique stops the training procedure, once the value of loss function on the validation set remains almost unchanged for several epochs, and thus prevents the network to over-fit on the training data. To speed up the learning phase, we also used *Batch Normalization* technique in our models (Ioffe and Szegedy 2015).

Table 2 Selected hyper-parameters for the CNNs

Task	C1 filter			C2 filter		
	Size	Number	Stride	Size	Number	Stride
App. idn.	4	200	3	5	200	1
Traffic char.	5	200	3	4	200	3

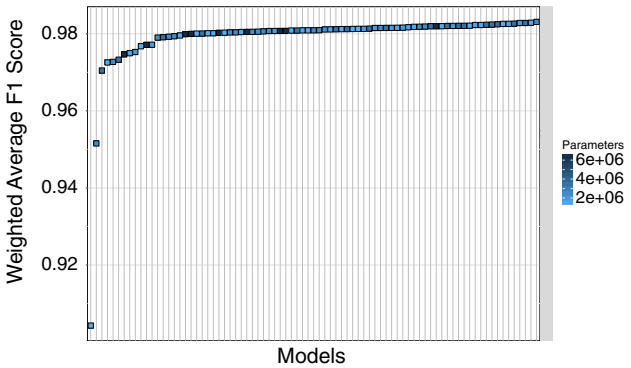
For training SAE, first each layer was trained in a greedy layer-wise fashion using *Adam* optimizer (Kingma and Ba 2014) and *mean squared error* as the loss function for 200 epochs, as described in Sect. 3.1. Next, in the fine-tuning phase, the whole network was trained for another 200 epochs using the *categorical cross entropy* loss function. Also, for implementing the proposed one-dimensional CNN, the *categorical cross entropy* and *Adam* were used as loss function and optimizer, respectively, and in this case, the network was trained for 300 epochs. Finally, it is worth mentioning that in both NNs, all layers employ Rectified Linear Unit (ReLU) as the activation function, except for the final softmax classifier layer.

To evaluate the performance of Deep Packet, we have used Recall (Rc), Precision (Pr) and F_1 Score (i.e., F_1) metrics. The above metrics are described mathematically as follows

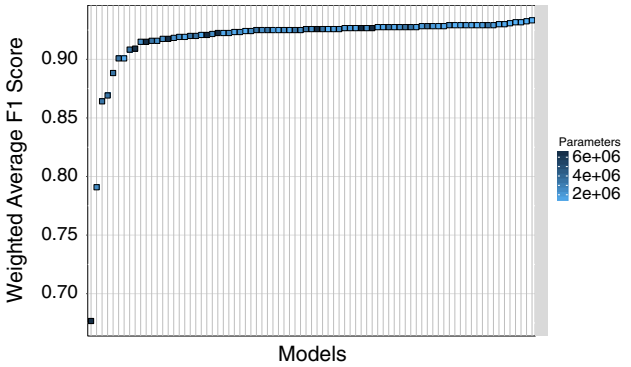
$$\mathrm{Rc} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}, \quad \mathrm{Pr} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}, \quad F_1 = \frac{2\cdot \mathrm{Rc}\cdot \mathrm{Pr}}{\mathrm{Rc}+\mathrm{Pr}}, \quad (3)$$

where TP, FP and FN stand for true positive, false positive and false negative, respectively.

As mentioned in Sect. 4, we used grid search hyper-parameters tuning scheme to find the best 1D-CNN structure in our work. Due to our computation hardware limitations, we only searched a restricted subspace of hyper-parameters to find the ones which maximize the weighted average F_1 score on the test set for each task. To be more specific, we changed filter size, the number of filters and stride for both convolutional layers. In total, 116 models with their weighted



(a) Aplication Identification.



(b) Traffic Characterization.

Fig. 6 Grid search on the hyper-parameters of the proposed 1D-CNN for **a** application identification, and **b** traffic characterization

average F_1 score for both application identification and traffic characterization tasks were evaluated. The result for all trained models can be seen in Fig. 6. We believe one cannot select an optimal model for traffic classification tasks since the definition of "optimal model" is not well defined and there exists a trade-off between the model accuracy and its complexity (i.e., training and test speed). In Fig. 6, the color of each point is associated with the model's trainable parameters; the darker the color, the higher the number of trainable parameters.

As seen in Fig. 6, increasing the complexity of the neural network does not necessarily result in a better performance. Many reasons can cause this phenomenon which among them one can mention to the vanishing gradient and over-fitting problems. A complex model is more likely to face the vanishing gradient problem which leads to under-fitting in the training phase. On the other hand, if a learning model becomes more complex while the size of training data remains the same, the over-fitting problem can be occurred. Both of these problems lead to a poor performance of NNs in the evaluation phase.

Table 3 shows the achieved performance of both SAE and 1D-CNN for the application identification task on the test set. The weighted average F_1 score of 0.98 and 0.95 for 1D-CNN and SAE, respectively, shows that our networks have entirely extracted and learned the discriminating features from the training set and can successfully distinguish each application. For the traffic characterization task, our proposed CNN and SAE have achieved F_1 score of 0.93 and 0.92, respectively, implying that both networks are capable of accurately classify packets. Table 4 summaries the achieved performance of the proposed methods on the test set.

Table 3 Deep Packet performance for the application identification task

Application	CNN			SAE		
	Rc	Pr	F_1	Rc	Pr	F_1
AIM chat	0.76	0.87	0.81	0.64	0.76	0.70
Email	0.82	0.97	0.89	0.99	0.94	0.97
Facebook	0.95	0.96	0.96	0.95	0.94	0.95
FTPS	1.00	1.00	1.00	0.77	0.97	0.86
Gmail	0.95	0.97	0.96	0.94	0.93	0.94
Hangouts	0.98	0.96	0.97	0.99	0.94	0.97
ICQ	0.80	0.72	0.76	0.69	0.69	0.69
Netflix	1.00	1.00	1.00	1.00	0.98	0.99
SCP	0.99	0.97	0.98	1.00	1.00	1.00
SFTP	1.00	1.00	1.00	0.96	0.70	0.81
Skype	0.99	0.94	0.97	0.93	0.95	0.94
Spotify	0.98	0.98	0.98	0.98	0.98	0.98
Torrent	1.00	1.00	1.00	0.99	0.99	0.99
Tor	1.00	1.00	1.00	1.00	1.00	1.00
VoipBuster	1.00	0.99	0.99	0.99	0.99	0.99
Vimeo	0.99	0.99	0.99	0.98	0.99	0.98
YouTube	0.99	0.99	0.99	0.98	0.99	0.99
Wtd. average	**0.98**	**0.98**	**0.98**	**0.96**	**0.95**	**0.95**

Table 4 Deep Packet performance for the traffic characterization task

Class name	CNN			SAE		
	Rc	Pr	F_1	Rc	Pr	F_1
Chat	0.71	0.84	0.77	0.68	0.82	0.74
Email	0.87	0.96	0.91	0.93	0.97	0.95
File transfer	1.00	0.98	0.99	0.99	0.98	0.99
Streaming	0.87	0.92	0.90	0.84	0.82	0.83
Torrent	1.00	1.00	1.00	0.99	0.97	0.98
VoIP	0.88	0.63	0.74	0.90	0.64	0.75
VPN: Chat	0.98	0.98	0.98	0.94	0.95	0.94
VPN: File transfer	0.99	0.99	0.99	0.95	0.98	0.97
VPN: Email	0.98	0.99	0.99	0.93	0.97	0.95
VPN: Streaming	1.00	1.00	1.00	0.99	0.99	0.99
VPN: Torrent	1.00	1.00	1.00	0.97	0.99	0.98
VPN: VoIP	1.00	0.99	1.00	1.00	0.99	0.99
Wtd. average	**0.94**	**0.93**	**0.93**	**0.92**	**0.92**	**0.92**

Table 5 A comparison between Deep Packet and other proposed methods on "ISCX VPN-nonVPN" dataset

Paper	Task	Metric	Results	Alg.
Deep Packet	Application	Accuracy	0.98	CNN
Yamansavascilar et al. (2017)	Identification		0.94	k-NN
Deep Packet	Traffic	Precision	0.93	CNN
Gil et al. (2016)	Characterization		0.90	C4.5

5.1 Comparison

In the following, we compare the results of Deep Packet with previous results using the "ISCX VPN-nonVPN" dataset. Moreover, the Deep Packet is compared against some of the other machine learning methods in Sect. 5.1.2.

5.1.1 Comparison with previous results

As mentioned in Sect. 2, authors in Gil et al. (2016) tried to characterize network traffic using time-related features handcrafted from traffic flows such as the duration of the flow and flow bytes per second. Yamansavascilar et al. also used such time-related features to identify the end-user application (Yamansavascilar et al. 2017). Both of these studies evaluated their models on the "ISCX VPN-nonVPN traffic dataset," and their best results can be found in Table 5. The results suggest that Deep Packet has outperformed other proposed approaches mentioned above, in both application identification and traffic characterization tasks.

We would like to emphasize that the above-mentioned work have used handcrafted features based on the network traffic flow. On the other hand, Deep Packet considers the network traffic in the packet level and can classify each packet of network traffic flow which is a harder task, since there is more information in a flow compared to a single packet. This feature allows Deep Packet to be more applicable in real-world situations.

Finally, it worth mentioning that independently and parallel to our work (Lotfollahi et al. 2017), Wang et al. proposed a similar approach to Deep Packet for traffic characterization on "ISCX VPN-nonVPN" traffic dataset (Wang et al. 2017). Their best-reported result achieves 100% precision on the traffic characterization task. However, we believe that their result is seriously questionable. The proving reason for our allegation is that their best result has been obtained by using packets containing all the headers from every five layers of the Internet protocol stack. However, based on our experiments and also a direct inquiry from the dataset providers (Gil et al. 2016), in "ISCX VPN-nonVPN" traffic dataset, the source and destination IP addresses (that are appeared in the header of network layer) are unique for each application. Therefore, their model presumably just uses this feature to classify the traffic (in that case a much simpler classifier would be sufficient to handle the classification task). As mentioned before, to avoid this phenomenon, we mask IP address fields in the pre-processing phase before feeding the packets into our NNs for training or testing.

Table 6 The comparison between Deep Packet and other machine learning methods in application identification

Classifier	Pc	Pr	F1
Decision tree	0.90	0.90	0.90
Random forests	0.91	0.90	0.90
Logistic regression	0.91	0.91	0.91
Naive Bayes	0.40	0.34	0.26

Table 7 The comparison between Deep Packet and other machine learning methods in traffic characterization

Classifier	Pc	Pr	F1
Decision tree	0.75	0.75	0.74
Random forests	0.80	0.80	0.79
Logistic regression	0.79	0.79	0.78
Naive Bayes	0.48	0.32	0.27

5.1.2 Comparison with previous methods

In this section, we compare Deep Packet with four machine learning algorithms. The comparison was performed by feeding pre-possessed packets similar to what we feed to Deep packet. We used scikit-learn (Pedregosa et al. 2011) implementation of the decision tree with depth two, random forests with depth four, logistic regression (with $c = 0.1$) and naive Bayes with default parameters. Table 6 indicates our method outperforms four alternative algorithms in application identification task for the test data. Similarly, Table 7 illustrates Deep Packet performs better in traffic characterization task.

These comparisons confirm the power of deep neural network for the network traffic classification where a huge amount of data have to be analyzed.

6 Discussion

Evaluating the SAE on the test set for the application identification and the traffic characterization tasks result in

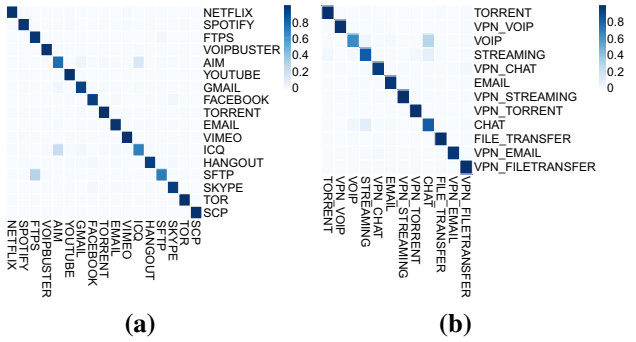
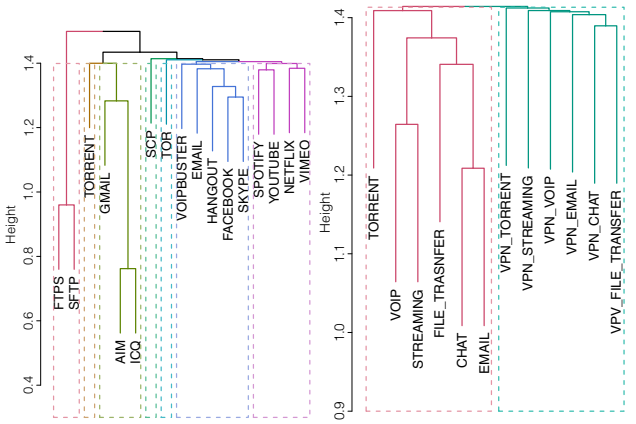


Fig. 7 Row-normalized confusion matrices using SAE on **a** application identification, and **b** traffic characterization



(a) Application identification using SAE. **(b)** Traffic characterization using SAE.

Fig. 8 Hierarchical clustering, performed on row-normalized confusion matrices of the proposed SAE network. Note that the height of fusion, provided on the vertical axis, indicates the (dis)similarity between two observations. The higher the height of the fusion, the less similar the observations are

row-normalized confusion matrices shown in Fig. 7. The rows of the confusion matrices correspond to the actual class of the samples, and the columns present the predicted label; thus, the matrices are row-normalized. The dark color of the elements on the main diagonal suggests that SAE can classify each application with minor confusion.

By carefully observing the confusion matrices in Fig. 7, one would notice some interesting confusion between different classes (e.g., ICQ and AIM). *Hierarchical clustering* further demonstrates the similarities captured by Deep Packet. Clustering on row-normalized confusion matrices for application identification with SAE (Fig. 7a), using *Euclidean distance* as the distance metric and *Ward.D* as the agglomeration method uncovers similarities among applications regarding their propensities to be assigned to the 17 application classes. As illustrated in Fig. 8a, application groupings revealed by Deep Packet generally agree with the applications' similarities in the real world. Hierarchical clustering divided the applications into 7 groups. Interestingly, these groups are to some extent similar to groups in the traffic characterization task. One would notice that Vimeo, Netflix, YouTube and Spotify which are bundled together are all streaming applications. There is also a cluster including ICQ, AIM, and Gmail. AIM and ICQ are used for online chatting, and Gmail in addition to email services offers a service for online chatting. Another interesting observation is that Skype, Facebook, and Hangouts are all grouped in a cluster together. Though these applications do not seem much relevant, this grouping can be justified. The dataset contains traffic for these applications in three forms: voice call, video call, and chat. Thus, the network has found these applications similar regarding their usage. FTPS (File Transfer Protocol over SSL) and SFTP (File Transfer Protocol over SSH) which are both used for transferring files between two remote systems securely are clustered together as well. Interestingly, SCP (Secure Copy) has formed its cluster although it is also used for remote file transfer. SCP uses SSH protocol for transferring file, while SFTP and FTPS use FTP. Presumably, our network has learned this subtle difference and separated them. Tor and Torrent have their clusters which are sensible due to their apparent differences with other applications. This clustering is not flawless. Clustering Skype, Facebook, and Hangouts along with Email and VoipBuster are not correct. VoipBuster is an application which offers voice communications over Internet infrastructure. Thus, applications in this cluster do not seem much similar regarding their usage, and this grouping is not precise.

The same procedure was performed on the confusion matrices of traffic characterization as illustrated in Fig. 8b. Interestingly, groupings separate the traffic into VPN and non-VPN clusters. All the VPN traffics are bundled together in one cluster, while all of non-VPNs are grouped together.

As mentioned in Sect. 2, many of the applications employ encryption to maintain clients' privacy. As a result, the majority of "ISCX VPN-nonVPN" dataset traffics are also encrypted. One might wonder how it is possible for Deep Packet to classify such encrypted traffics. Unlike DPI methods, Deep Packet does not inspect the packets for keywords. In contrast, it attempts to learn features in traffic generated by each application. Consequently, it does not need to decrypt the packets to classify them.

An ideal encryption scheme causes the output message to bear the maximum possible entropy (Cover and Thomas 2006). In other words, it produces patternless data that theoretically cannot be distinguished from one another. However, due to the fact that all practical encryption schemes use pseudo-random generators, this hypothesis is not valid in practice. Moreover, each application employs different (non-ideal) ciphering scheme for data encryption. These schemes utilize different pseudo-random generator algorithms which leads to distinguishable patterns. Such variations in the pattern can be used to separate applications from one another.

Table 8 Tor traffic classification results

Class name	CNN			SAE		
	Rc	Pr	F_1	Rc	Pr	F_1
Tor: Google	0.00	0.00	0.00	0.44	0.03	0.06
Tor: Facebook	0.24	0.10	0.14	0.28	0.06	0.09
Tor: YouTube	0.44	0.55	0.49	0.44	0.99	0.61
Tor: Twitter	0.17	0.01	0.01	0.37	0.00	0.00
Tor: Vimeo	0.36	0.44	0.40	0.91	0.05	0.09
Wtd. average	**0.35**	**0.40**	**0.36**	**0.57**	**0.44**	**0.30**

Deep Packet attempts to extract those discriminative patterns and learns them. Hence, it can classify encrypted traffic accurately.

It is noticeable from Table 3 that Tor traffic is also successfully classified. To further investigate this kind of traffic, we conducted another experiment in which we trained and tested Deep Packet with a dataset containing only Tor traffic. To achieve the best possible result, we performed a grid search on the hyper-parameters of the NN, as discussed before. The detailed results can be found in Table 8, which shows that Deep Packet was unable to classify the underlying Tor's traffic accurately. This phenomenon is not far from what we expected. Tor encrypts its traffic, before transmission. As mentioned earlier, Deep Packet presumably learns different pseudo-random patterns used in various encryption schemes used by applications. At this experiment, traffic was tunneled through Tor. Hence, they all experience the same encryption scheme. Consequently, our neural network was not able to separate them apart well.

7 Future work

The reasons why deep neural networks perform so well in practice are yet to be understood. In addition, there is no rigorous theoretical framework to design and analyze such networks. If there is some progress in these matters, it will have direct impact on proposing better deep neural network structures specialized for network traffic classification. Along the same line, one of the other important future direction would be investigating the interpretability (Du et al. 2018; Montavon et al. 2018; Samek et al. 2018) of our proposed model. This will include analyzing the features that the model has learned and the process of learning them.

Another important direction to be studied would be the robustness analysis of proposed schemes against noisy and maliciously generated inputs using adversarial attack algorithms (Yuan et al. 2017). Adversarial attacks on machine learning methods have been widely studied in some other fields (e.g., Akhtar and Mian 2018; Huang et al. 2017; Carlini and Wagner 2018) but not in network traffic classification.

Designing multi-level classification algorithms is also an interesting possible direction for future research. This means that the system should be able to detect whether a traffic is from one of the known previous classes or a new "unknown" class. If the packet is labeled as *unknown*, then it will be added to a database of unknown classes. Further, by receiving more unknown packets, one can use an unsupervised clustering algorithm to label them as discrete classes. Next, human experts will be able to map these unknown classes to well-known real-world applications. Thus, re-training the first level classifier would become possible with these new labeled classes. Re-training can be done with an online learning algorithm or using previously learned weights of the neural network as initialization for the newer network.

Finally, implementing the proposed schemes to be able to handle the real-world high-speed network traffic will be an important real challenge. This can be done for example by taking advantage of hardware implementation (e.g., see Vanhoucke et al. 2011; Zhang et al. 2015) and applying neural network simplification techniques (e.g., see Hubara et al. 2017; Lin et al. 2016).

8 Conclusion

In this paper, we presented Deep Packet, a framework that automatically extracts features from computer networks traffic using deep learning algorithms to classify traffic. To the best of our knowledge, Deep Packet is the first traffic classification system using deep learning algorithms, namely SAE and 1D-CNN that can handle both application identification and traffic characterization tasks. Our results showed that Deep Packet outperforms all of the similar works on the "ISCX VPN-nonVPN" traffic dataset, in both application identification and traffic characterization tasks, to the date. Moreover, with state-of-the-art results achieved by Deep Packet, we envisage that Deep Packet is the first step toward a general trend of using deep learning algorithms in traffic classification and more generally network analysis tasks. Furthermore, Deep Packet can be modified to handle more complex tasks like multi-channel (e.g., distinguishing between different types of Skype traffic including chat, voice call, and video call) classification, accurate classification of Tor's traffic, etc. Finally, the automatic feature extraction procedure from network traffic can save the cost of employing experts to identify and extract handcrafted features from the traffic which eventually leads to more accurate traffic classification.

Acknowledgements The authors would like to thank Farzad Ghasemi and Mehdi Kharrazi for their valuable discussions and feedback.

Compliance with ethical standards

Conflict of interest Mohammad Lotfollahi, Mahdi Jafari Siavoshani, Ramin Shirali Hossein Zade, and Mohammdsadegh Saberian declare that they have no conflict of interest

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

A Hyper-parameters

Here, in this section, we present all of the hyper-parameters of the proposed SAE and CNN for the traffic characterization and application identification tasks. These parameters are stated in Tables 9, 10, 11 and 12.

In the following, to have more compact tables, we have used some abbreviations, namely FC stands for "Fully connected," NoF stands for "Number of Features," DO stands for "Dropout," BN stands for "Batch Normalization," NL stands for "Nonlinearity," Str stands for "Strides," and Kl stands for "Kernel."

Table 9 SAE detailed architecture for the application identification task

Operation	NoF	DO	BN	NL
Input packet	–	–	–	–
FC	400	0.05	No	ReLU
FC	300	0.05	No	ReLU
FC	200	0.05	No	ReLU
FC	100	0.05	No	ReLU
FC	50	0.05	No	ReLU
FC	17	–	–	Softmax

Table 10 SAE detailed architecture for the traffic characterization task

Operation	NoF	DO	BN	NL
Input packet	–	–	–	–
FC	400	0.05	No	ReLU
FC	300	0.05	No	ReLU
FC	200	0.05	No	ReLU
FC	100	0.05	No	ReLU
FC	50	0.05	No	ReLU
FC	12	–	–	Softmax

Table 11 CNN detailed architecture for the application identification task

Operation	Kl	Str	NoF	DO	BN	NL
Input packet	–	–	–	–	–	–
Convolution	4	3	200	0.05	Yes	ReLU
Convolution	5	1	200	0.05	Yes	ReLU
Max pooling	–	2	–	0.05	No	–
Flattening	–	–	–	–	–	–
FC	–	–	200	0.05	No	ReLU
FC	–	–	100	0.05	No	ReLU
FC	–	–	50	0.05	No	ReLU
FC	–	–	17	–	–	Softmax

Table 12 CNN detailed architecture for the traffic characterization task

Operation	Kl	Str	NoF	DO	BN	NL
Input packet	–	–	–	–	–	–
Convolution	5	3	200	0.05	Yes	ReLU
Convolution	4	3	200	0.05	Yes	ReLU
Max pooling	–	2	–	0.05	No	–
Flattening	–	–	–	–	–	–
FC	–	–	200	0.05	No	ReLU
FC	–	–	100	0.05	No	ReLU
FC	–	–	50	0.05	No	ReLU
FC	–	–	12	–	–	Softmax

References

Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M, Ghemawat S, Goodfellow I, Harp A, Irving G, Isard M, Jia Y, Jozefowicz R, Kaiser L, Kudlur M, Levenberg J, Mané D, Monga R, Moore S, Murray D, Olah C, Schuster M, Shlens J, Steiner B, Sutskever I, Talwar K, Tucker P, Vanhoucke V, Vasudevan V, Viégas F, Vinyals O, Warden P, Wattenberg M, Wicke M, Yu Y, Zheng X (2015) TensorFlow: large-scale machine learning on heterogeneous systems. http://tensorflow.org/, software available from tensorflow.org

Akhtar N, Mian A (2018) Threat of adversarial attacks on deep learning in computer vision: a survey. arXiv:1801.00553

Alipanahi B, Delong A, Weirauch MT, Frey BJ (2015) Predicting the sequence specificities of dna-and rna-binding proteins by deep learning. Nat Biotechnol 33(8):831–838

Alshammari R, Zincir-Heywood AN (2011) Can encrypted traffic be identified without port numbers, ip addresses and payload inspection? Comput Netw 55(6):1326–1350

Auld T, Moore AW, Gull SF (2007) Bayesian neural networks for internet traffic classification. IEEE Trans Neural Netw 18(1):223–239

Bagui S, Fang X, Kalaimannan E, Bagui SC, Sheehan J (2017) Comparison of machine-learning algorithms for classification of vpn network traffic flow using time-related features. J Cyber Secur Technol 1(2):108–126

Bengio Y (2009) Learning deep architectures for ai. Found Trends Mach Learn 2(1):1–127
Bengio Y, Lamblin P, Popovici D, Larochelle H (2007) Greedy layer-wise training of deep networks. In: Advances in neural information processing systems, pp 153–160
Carlini N, Wagner D (2018) Audio adversarial examples: targeted attacks on speech-to-text. arXiv:1801.01944
Caudill M (1987) Neural networks primer, part i. AI Expert 2(12):46–52
Chollet F et al (2017) Keras. https://github.com/fchollet/keras
Chowdhury NMK, Boutaba R (2010) A survey of network virtualization. Comput Netw 54(5):862–876
Cover TM, Thomas JA (2006) Elements of information theory. Wiley Series in Telecommunications and Signal Processing. Wiley-Interscience, New Jersy
Crotti M, Dusi M, Gringoli F, Salgarelli L (2007) Traffic classification through simple statistical fingerprinting. ACM SIGCOMM Comput Commun Rev 37(1):5–16
Dainotti A, Pescape A, Claffy KC (2012) Issues and future directions in traffic classification. IEEE Netw 26(1):35-40
Dingledine R, Mathewson N, Syverson P (2004) Tor: the second-generation onion router. Tech. rep., Naval Research Lab Washington DC
dos Santos CN, Gatti M (2014) Deep convolutional neural networks for sentiment analysis of short texts. In: Proceedings of the 25th international conference on computational linguistics (COLING), Dublin, Ireland
Du M, Liu N, Hu X (2018) Techniques for interpretable machine learning. arXiv preprint. arXiv:1808.00033
Finsterbusch M, Richter C, Rocha E, Muller JA, Hanssgen K (2014) A survey of payload-based traffic classification approaches. IEEE Commun Surv Tutor 16(2):1135–1156
Gil GD, Lashkari AH, Mamun M, Ghorbani AA (2016) Characterization of encrypted and vpn traffic using time-related features. In: Proceedings of the 2nd international conference on information systems security and privacy (ICISSP 2016), pp 407–414
Hinton G, Deng L, Yu D, Dahl GE, Ar Mohamed, Jaitly N, Senior A, Vanhoucke V, Nguyen P, Sainath TN et al (2012) Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. IEEE Signal Process Mag 29(6):82–97
Huang SH, Papernot N, Goodfellow IJ, Duan Y, Abbeel P (2017) Adversarial attacks on neural network policies. arXiv:1702.02284
Hubara I, Courbariaux M, Soudry D, El-Yaniv R, Bengio Y (2017) Quantized neural networks: training neural networks with low precision weights and activations. J Mach Learn Res 18(1):6869–6898
Hubel DH, Wiesel TN (1968) Receptive fields and functional architecture of monkey striate cortex. J Physiol 195(1):215–243
Ioffe S, Szegedy C (2015) Batch normalization: accelerating deep network training by reducing internal covariate shift. In: International conference on machine learning, pp 448–456
Khalife J, Hajjar A, Diaz-Verdejo J (2014) A multilevel taxonomy and requirements for an optimal traffic-classification model. Int J Netw Manag 24(2):101–120
Kingma D, Ba J (2014) Adam: a method for stochastic optimization. arXiv preprint. arXiv:1412.6980
Kohout J, Pevný T (2018) Network traffic fingerprinting based on approximated kernel two-sample test. IEEE Trans Inf Forensics Secur. https://doi.org/10.1109/TIFS.2017.2768018
LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436–444
Lee H, Grosse R, Ranganath R, Ng AY (2009) Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In: Proceedings of the 26th annual international conference on machine learning, ACM, New York, NY, USA, ICML '09, pp 609–616
Lin DD, Talathi SS, Annapureddy VS (2016) Fixed point quantization of deep convolutional networks. In: Proceedings of the 33rd international conference on international conference on machine learning, vol 48, ICML'16, pp 2849–2858
Longadge R, Dongre S (2013) Class imbalance problem in data mining review. arXiv preprint. arXiv:1305.1707
Lotfollahi M, Shirali Hossein Zade R, Jafari Siavoshani M, Saberian M (2017) Deep packet: a novel approach for encrypted traffic classification using deep learning. CoRR abs/1709.02656. arXiv:1709.02656
Lotfollahi M, Wolf FA, Theis FJ (2018) Generative modeling and latent space arithmetics predict single-cell perturbation response across cell types, studies and species. bioRxiv p 478503
Lv J, Zhu C, Tang S, Yang C (2014) Deepflow: hiding anonymous communication traffic in p2p streaming networks. Wuhan Univ J Nat Sci 19(5):417–425
Madhukar A, Williamson C (2006) A longitudinal study of p2p traffic classification. In: Modeling, analysis, and simulation of computer and telecommunication systems, 2006. MASCOTS 2006. 14th IEEE international symposium on, IEEE, pp 179–188
Montavon G, Samek W, Müller KR (2018) Methods for interpreting and understanding deep neural networks. Digit Signal Process 73:1–15
Moore AW, Papagiannaki K (2005) Toward the accurate identification of network applications. PAM, Springer 5:41–54
Moore AW, Zuev D (2005) Internet traffic classification using Bayesian analysis techniques. ACM SIGMETRICS Perform Eval Rev ACM 33:50–60
Moore A, Zuev D, Crogan M (2013) Discriminators for use in flow-based classification. Tech. rep
Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: machine learning in Python. J Mach Learn Res 12:2825–2830
Perera P, Tian YC, Fidge C, Kelly W (2017) A comparison of supervised machine learning algorithms for classification of communications network traffic. In: Neural information processing. Springer, Cham, Lecture Notes in Computer Science, pp 445–454. https://doi.org/10.1007/978-3-319-70087-8_47
Prechelt L (1998) Early stopping-but when? Neural networks: tricks of the trade. Springer, pp 55–69
Qi Y, Xu L, Yang B, Xue Y, Li J (2009) Packet classification algorithms: from theory to practice. In: INFOCOM 2009, IEEE, IEEE, pp 648–656
Samek W, Wiegand T, Müller KR (2018) Explainable artificial intelligence: understanding, visualizing and interpreting deep learning models. ITU J ICT Discov Special Issue 1 Impact Artif Intell (AI) Commun Netw Serv 1(1):39–48
Sen S, Spatscheck O, Wang D (2004) Accurate, scalable in-network identification of p2p traffic using application signatures. In: Proceedings of the 13th international conference on world wide web, ACM, New York, NY, USA, pp 512–521
Sherry J, Lan C, Popa RA, Ratnasamy S (2015) Blindbox: deep packet inspection over encrypted traffic. ACM SIGCOMM Comput Commun Rev ACM 45:213–226
Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv preprint. arXiv:1409.1556
Socher R, Perelygin A, Wu JY, Chuang J, Manning CD, Ng AY, Potts CP (2013) Recursive deep models for semantic compositionality over a sentiment treebank. In: EMNLP
Sun R, Yang B, Peng L, Chen Z, Zhang L, Jing S (2010) Traffic classification using probabilistic neural networks. In: Natural computation (ICNC), 2010 sixth international conference on, IEEE, vol 4, pp 1914–1919
Ting H, Yong W, Xiaoling T (2010) Network traffic classification based on kernel self-organizing maps. In: Intelligent computing and inte-

grated systems (ICISS), 2010 international conference on, IEEE, pp 310–314
Vanhoucke V, Senior A, Mao MZ (2011) Improving the speed of neural networks on cpus. In: Deep learning and unsupervised feature learning workshop, NIPS 2011
Velan P, Čermák M, Čeleda P, Drašar M (2015) A survey of methods for encrypted traffic classification and analysis. Int J Netw Manag 25(5):355–374
Vincent P, Larochelle H, Bengio Y, Manzagol PA (2008) Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th international conference on machine learning, ACM, pp 1096–1103
Wang Z (2015) The applications of deep learning on traffic identification. BlackHat, USA
Wang X, Parish DJ (2010) Optimised multi-stage tcp traffic classifier based on packet size distributions. In: Communication theory, reliability, and quality of service (CTRQ), 2010 third international conference on, IEEE, pp 98–103
Wang W, Zhu M, Wang J, Zeng X, Yang Z (2017) End-to-end encrypted traffic classification with one-dimensional convolution neural networks. In: Intelligence and security informatics (ISI), 2017 IEEE international conference on, IEEE, pp 43–48
Yamansavascilar B, Guvensan MA, Yavuz AG, Karsligil ME (2017) Application identification via network traffic classification. In: Computing, networking and communications (ICNC), 2017 international conference on, IEEE, pp 843–848
Yeganeh SH, Eftekhar M, Ganjali Y, Keralapura R, Nucci A (2012) Cute: traffic classification using terms. In: Computer communications and networks (ICCCN), 2012 21st international conference on, IEEE, pp 1–9
Yosinski J, Clune J, Nguyen AM, Fuchs TJ, Lipson H (2015) Understanding neural networks through deep visualization. arXiv:1506.06579
Yuan X, He P, Zhu Q, Bhat RR, Li X (2017) Adversarial examples: attacks and defenses for deep learning. arXiv:1712.07107
Zhang C, Li P, Sun G, Guan Y, Xiao B, Cong J (2015) Optimizing fpga-based accelerator design for deep convolutional neural networks. In: Proceedings of the 2015 ACM/SIGDA international symposium on field-programmable gate arrays, ACM, pp 161–170