# Deep Packet: A Novel Approach For Encrypted Traffic Classification Using Deep Learning

Mohammad Lotfollahi, Ramin Shirali, Mahdi Jafari Siavoshani, Mohammdsadegh Saberian

*Sharif University of Technology, Teharn, Iran*

arXiv:1709.02656v1 [cs.LG] 8 Sep 2017

**Abstract**

Internet traffic classification has become more important with rapid growth of current Internet network and online applications. There have been numerous studies on this topic which have led to many different approaches. Most of these approaches exploit 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 integrate both feature extraction and classification phases into one system. Our proposed scheme, called "Deep Packet," can handle both *traffic categorization* in which the network traffic is categorize into major classes (e.g., FTP, P2P, etc.) and *application identification* in which one identify end-user applications (e.g., BitTorrent, Skype, etc.). Contrary to most of the current methods, Deep Packet can identify encrypted traffic and also distinguishes between VPN and non-VPN network traffic. After initial pre-processing phase on data, packets are fed to Deep Packet framework that embeds stacked autoencoder and convolution neural network in order to classify network traffic. Deep packet with CNN as its classification model achieved $F_1$ score of 0.95 in application identification and it also accomplished $F_1$ score of 0.97 in traffic categorization task. To the best of our knowledge, Deep Packet outperforms all of the classification and categorization methods on UNB ISCX VPN-nonVPN dataset.

*Keywords:* Application Identification, Traffic characterization, Deep Learning, Convolutional Neural Networks, Stacked autoencoder, Deep Packet.

## 1. Introduction

Network *traffic classification* is an important task in modern communication networks [1]. Due the rapid growth of high throughput network traffic demands, in order to properly manage network resources, it is vital to recognize different types of applications utilizing network resources. Consequently, accurate traffic classification and identification has become one of the prerequisites for advanced network management tasks such as providing appropriate Quality-of-Service (QoS), pricing, anomaly detection, etc. Traffic classification, and more specifically *application identification*, have attracted a lot of interests in both academia and industrial activities related to network management (e.g., see [2], [3], [4] and the references therein).

As an example of why network traffic identification is important, 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), video call, etc.) has changed the clients' demands to deviate from the aforementioned assumption. Thus, in order to provide a satisfactory experience for the clients, extra application-level knowledge is required to allocate adequate resources to such applications.

Emergence of new applications as well as interactions between various components on the Internet, have 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 important challenges of network traffic classification.

First, according to [5, 6], only 30% to 70% of the current generated traffic can be identified based on the connection's port numbers. In the classical approach, the connection's port numbers are compared to the list of standard ports proposed by the International Assigned Number Authority (IANA) for each application. However, many of the recent applications such as P2P applications, video call, etc., may use a port number initially assigned to a different protocol. For instance, using the standard web port 80 and secure shell (SSH) port 22 are very popular among such applications. This deviation in port assignment from IANA proposed list, leads to a poor performance of traditional port-based traffic classification methods.

Furthermore, the increasing demand for user's privacy and data encryption has tremendously raised the amount of encrypted traffic in today's Internet [4]. Encryption procedure turns the original data into a pseudo-random-like format in order to make it hard to decrypt. This causes the encrypted data scarcely contain any discriminative pat-

terns to identify network traffic. Therefore, accurate classification of encrypted traffic has become a challenge in modern network [2].

It is also worth mentioning that many of the proposed network traffic classification approaches, such as payload inspection as well as machine learning and statistical 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 [7]. These applications use protocol embedding to bypass traffic control systems [8]. These applications embed their content inside well-known protocols packets, e.g., Hypertext Transfer Protocol (HTTP), which are allowed to pass any network to escape this control procedures. The identification of these kind of applications is one of the important challenges in traffic classification task.

There have been abundant studies on the network traffic classification subject [9, 10, 5]. 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.) [11]. In contrast, this work proposes a method based on the ideas recently developed in the machine learning community, namely, deep learning, [12, 13], 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 the proposed method, there is no need for an expert to extract features related to network traffic. In the light of this approach, the cumbersome step of finding and extracting distinguishing features has been omitted.
- Our proposed method can identify traffic in both granular level (application identification and traffic categorization) with sate-of-the-art results compared to the works conducted on similar dataset [10, 14, 15, 16].
- Our approach can accurately classify one of the hardest class of applications, known to be P2P [11]. These kind of applications routinely use advanced port obfuscation techniques, embedding their information in well-known protocols' packets and using random ports in order to circumvent ISPs' controlling processes.

### *1.1. Related Works*

In this section, we provide an overview of the most important Internet traffic classification methods. In particular, we can categorize these approaches into four main category as follows: (i) port-based, (ii) graphical techniques, (iii) payload inspection, and (iv) statistical and machine learning. Here is a brief review of the most important and recent studies regarding each of the aforementioned approaches.

**Port-based approach:** Traffic identification via port number is the oldest and the most well-known method for this task [2]. 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 identification. 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 firewall and access control list (ACL) [17]. 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 [5, 6] only 30% to 70% of the current Internet traffic can be classified using port-based classification methods. Because of the aforementioned reasons, more complex traffic classification methods are needed to identify modern network traffic.

**Payload Inspection Techniques:** These techniques are based on the analysis of information in the application layer payload of packets [11]. Most of the payload inspection methods (also known as deep packet inspection (DPI)) exploit predefined patterns like regular expressions as signatures for each protocol (e.g., see [15] and [18]). The derived pattern is then used to distinguish protocols form each other. The need for updating patterns when a new protocol is released and user privacy issues are among the two most important weaknesses of this approach. Sherry et al. [19] proposed a new DPI system that can inspect encrypted payload without decryption, thus solved the user privacy issue, but it can only process HTTPS traffic.

**Graphical Techniques:** These techniques employ the interaction graphs of the hosts who are communicating with each other and analyze such graphs with graph theory techniques. The oldest graphical techniques are called Graphlets [9]. The Graphlets are graphs modeling the interactions among hosts at the application level layer. Every application has its own graphlets which are almost unique for that particular application [11]. Social network graphs derived from the host interactions are another kind of graphical techniques used in order to classify traffic. Iliofotou et al., [20], proposed a traffic dispersion graph (TDG) based method called "Graption" to classify P2P applications. They achieved 95% of accuracy covering Gnutella, e-Donkey, FastTrack, Soribada, MP2P, and BitTorrent P2P applications. Motif based classification is also another kind of Graphlet methods. Allan et al., [21], proposed using binary metric which measures weather the host is involved in using an application or not. The authors

achieved 85% accuracy classifying hosts over a protocol set including AIM, DNS, HTTP, MSDS, NETBIOS, SSH, and Kazaa.

**Statistical and machine learning approach:** The statistical approach comprises of the following two main methods:

- *Simple and complex statistical methods*: These methods have a biased assumption that the underlying traffic for each application have some statistical features which is almost unique for each application. Each statistical method uses its own functions and statistics. Crotti et al. [22] 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 et al. [23] 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%.

- *Machine learning based methods*: A vast number of machine learning approaches have been published to classify traffic. Auld et al.[16] proposed a Bayesian neural network that was trained to classify most well-known P2P protocols including Kazaa, BitTorrent, GnuTella, and achieved 99% accuracy. Moore et al. [24] achieved 96% of accuracy on the same set of applications using a Naive Bayes classifier and the kernel density estimator. Artificial neural network (ANN) approaches were proposed for traffic identification (e.g., see [25] and [26]). Moreover, it was shown in [26] that the ANN approach can outperform Naive Bayes methods. Two of the most important papers that have been published on "ISCX VPN-non VPN traffic" dataset are based on machin learning methods. Gil et al. [10] used time related features like the duration of the flow, flow bytes per second, forward and backward inter-arrival time and etc. to characterize the network traffic using k-nearest neighbor (k-NN) and C4.5 decision tree algorithms. They achieved approximately $91-92\%$ recall characterizing six major classes of traffic including Web browsing, email, chat, streaming, file transfer and VoIP using C4.5 algorithm. They also achieved approximately 88% recall using C4.5 algorithm on same dataset which is tunneled through VPN. Yamansavascilar et al. [14] manually selected 111 flow features described in [27] and achieved 93.94% of accuracy for 14 class of applications using k-NN algorithm.

  To the best of our knowledge, prior to our work, only one study based on deep learning ideas has been reported by Wangc [28]. They exploited stacked autoencoders (SAE) to classify some network traffic for a large family of protocols like HTTP, SMTP and etc. However, in their report, they did not mention to the dataset they used. Moreover, the methodology of their scheme, the details of their implementation, and the proper report of their result is missing.

The rest of paper is organized as follows. In Section 2, we present some essential background on machine learning which are necessary to our work. In Section 3, our methodology is presented. The results of our proposed method on network traffic identification is described in Section 4. In Section 5, we provide further discussion on experimental results. Finally, we conclude the paper in Section 6.

## 2. Background

In the following, we provide the background necessary to understand our proposed method for classification of network traffic using deep learning.

### *2.1. Neural Networks*

Neural networks (NNs) are computing systems made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs [29]. In practice, these networks are typically constructed from a huge number of building blocks called *neuron* where they are connected via links to each other. These links are called connections and to each connection a weight number is associated. During the training procedure, the NN is fed with possibly a huge number of data samples. The widely used learning algorithm used to train such networks (which is called *back-propagation*) adjusts the weights in order to achieve the desired output from the NN. It is known that with more data samples, the supervuied learning models including NNs can become better and more powerful in doing the classification task [30]. Intuitively, this happens because by observing more samples the NN can better approximate the underlying distribution governing the data. Hence, it is advised to use NN framework when there exist sufficient data to train and test the network.

### *2.2. Deep Learning*

The deep learning framework can be considered as a special kind of NNs with many (hidden) layers. Nowadays, with rapid growth of computational power and the availability of graphical processing units (GPUs), training deep NNs have become plausible. Therefore, the researchers from different scientific fields consider using deep learning algorithms in their respective area of research. It is worth mentioning that deep learning has achieved state-of-the-art results in many fields such as speech recognition [31], machine vision [32], and natural language processing [33].

### 2.3. Autoencoder

The autoencoder NN is an unsupervised learning framework that uses backpropagation to reconstruct the input at the output while minimizing the reconstruction error (i.e., according to some criteria). Suppose we have a training set $\{x^1, x^2, \ldots, x^n\}$ where for each training data we have $x^i \in \mathbb{R}^n$. Then, the autoencoder objective is to set $y^i = x^i$ for $i \in [1 : n]$. Considering this objective function, the autoencoder tries to learn a compressed representation of the dataset, i.e., it approximately learns the identity function, $F_{W,b}(x) \simeq x$, where $W$ and $b$ are the whole network weights and biases vectors. The first half of autoencoder

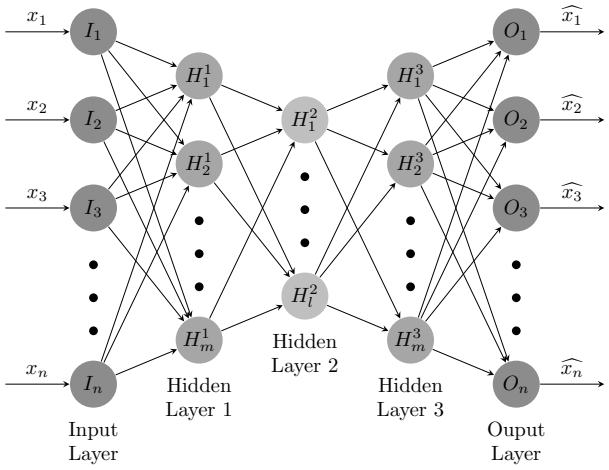



Figure 1: The general structure of an autoencoder. An autoencoder tries to reconstruct its input at the output, with minimum reconstruction error. As the architecture of this neural network suggests, the autoencoder first decreases the input dimension by employing successive layers, transforming the input data to a compact representation. Then, this compact version of data is used to reconstruct the input with minimum error.

which reduces the dimension of data is called encoder. On the other hand, the part of autoencoder which tries to reconstruct the data from the low-dimensional representation made by the encoder is called decoder. Fig. 1 shows a typical autoencoder with $n$ inputs and outputs.

In practice, to obtain a better performance, a more complex architecture and training procedure, called stacked autoencoder (SAE), is proposed [34]. 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 [35]. First, this method trains each layer of network while freezing the weights of other layers. After training all the layers, in order to have more accurate results, fine-tuning is applied. At the fine-tuning phase, the backpropagation algorithm is used to adjust all layers' weights. Moreover, for classification task, an extra soft-max layer can be applied as the final layer. Fig. 2, depicts the training procedure of a stacked autoencoder.

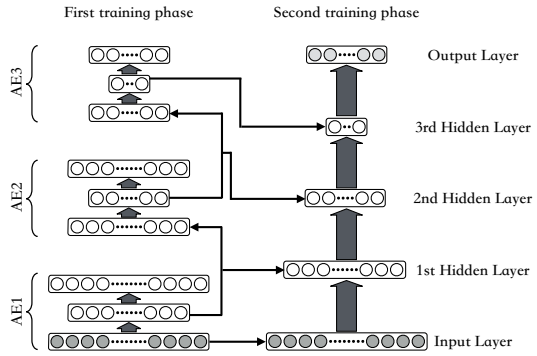



Figure 2: In a stacked autoencoder, each layer is trained as an standalone autoencoder, while freezing the weights of other layers. Afterward, fine-tuning is applied to the whole NN to improve its performance.

### 2.4. Convolutional Neural Network

The convolutional neural networks (CNNs) are another deep learning model in which feature extraction from the input data is done using layers comprised of convolutional functions. The structure of convolutional networks is inspired by visual mechanism of living organisms [36]. Similar to other kinds of deep learning models, feature extraction plays an important role in convolutional networks. Features extracted in shallower layers of the convolutional network will be fed to the successive convolutional layers in order to extract more abstract features.

CNNs exploit neurons with local connections to the input which means every neuron is only connected to to a limited numbers of adjacent elements in the input. This feature is also inspired by visual system of animals. In other words, in a CNN, neurons in layer $l$ are connected to a limited number of input neurons in layer $l-1$. The subset of neurons in layer $l-1$ which act as the input for a neuron in the successive layer are called the *receptive field* of that particular neuron [36]. It is important to mention that each neuron is unresponsive to variations outside of its receptive field. Another important component of a CNN is *pooling* mechanism. It is common to periodically insert a pooling layer in-between successive convolutional layers. The function of such layers is to progressively reduce the spatial size of the data representation in order to reduce the amount of parameters and computations in the network.

CNNs have been applied to different fields including natural language processing [37], computational biology [38], and machine vision [32]. One of the most interesting application of CNNs is in face recognition[39]. In face recognition task, consecutive convolutional layers are used in order to extract features from each picture. The extracted features in shallow layers are simple like edges and

curves. Features in deep layers of networks are more abstract than the ones in shallow layers. It is worth to mention that visualizing the extracted features in the middle layers of the network are not always meaningful like face recognition task. For example in one-dimensional CNN which we used to classify traffic network, the feature vectors are just some real numbers which make no sense at all.

## 3. Methodology

In this work, we developed a framework that comprises of two deep learning methods, namely, convolutional NN and stacked autoencoder NN, for both application identification and traffic characterization tasks. Prior to training the NNs, we have to prepare the network traffic data so that it can be fed into NNs properly. To this end, we preform a pre-processing phase on the dataset. Fig. 3 displays the general scheme for proposed method. At test phase, after loading *packet capture* (pcap) files, user selects the classification task which is required, namely application identification or traffic categorization. Afterwards, appropriate type of pre-trained neural network selection is done by user to do required task. The dataset, implementation and design details of the pre-processing phase and the architecture of proposed NNs will be explained in the following.

### *3.1. Dataset*

For this work, we use "ISCX VPN-nonVPN traffic dataset" [10], that consists of captured traffic of different applications in pcap format files. In this dataset, the captured packets are separated into different pcap files labeled according to the application produced the packets (e.g., Skype) 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 [10].

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, is the most prominent aspect of VPNs [40]. 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, file transfer and chat.

Moreover, this dataset includes captured traffic of Tor software. This traffic is presumably is generated while using Tor browser and it has labels like 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 a volunteer-operated servers. Tor was proposed to protect users against Internet surveillance known as "traffic analysis." In order 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 [41].

### *3.2. Pre-processing*

The dataset was captured at 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 are essential for forward the frames in network, but it is uninformative considering application identification or traffic characterization. Hence, at 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 a 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 with TCP headers in length. The remaining packets are then transformed from bits to bytes, as many fields in the layers' headers are made up 4 or 2 bytes. It also helps to reduce in input size for the NNs.

Fig. 4 illustrates the packet length for the dataset's packets. As the histogram shows, packet length varies a lot thorough the dataset, while employing NNs necessitates using a fixed-size input. Hence, truncation at a fixed length and zero-padding is required inevitably. In order too find the fixed length for truncation, we inspected the packets' statistics. We observed that approximately 96% of packets have a length of less than 1480 bytes. This is not far from our expectation, as most of the computer networks are constrained by Maximum Transmission Unit (MTU) size of 1500 bytes. We keep the IP header and the first 1480 bytes of each IP packet which results in a 1500 bytes vector as input for our proposed NNs. Needless to say, packets with IP payload less than 1480 bytes, are zero-padded at the end. For better performance, all the packet bytes are divided to 255, maximum value for a byte, so that all the input values are in $[0-1]$ range. All of the these pre-processing steps take place when user loads a pcap file into Deep Packet toolkit.

#### *3.2.1. Labeling Dataset*

As mentioned before at 3.1, the dataset's pcap files are labeled according to the applications and activities they were engaged to. However for application identification and traffic characterization tasks, we need to redefine the labels, with respect to each task. For application identification all pcap files labeled as a particular application disregarding the activities and VPN or non-VPN condition, are aggregated into a single file. This leads to 17 distinct labels shown at Table 2a. As for traffic characterization, we aggregated the captured traffic of different applications involved in the same activity into a single pcap

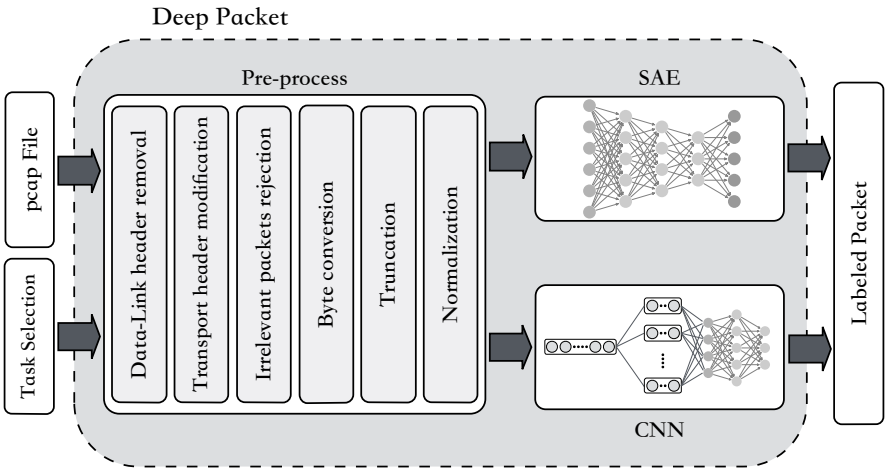



Figure 3: General illustration of Deep Packet toolkit. Firstly, user loads pcap files and selects the classification task between application identification or traffic categorization. At pre-processing phase pcap files are processed. First the header for data-link layer (Ethernet header) is removed. Then, the header of transport layer for UDP packets is modified. Afterwards, irrelevant packets like DNS or TCP handshaking packets are discarded. Next, the packets are converted from bits to bytes. Due to necessity of fixed input size for NNs, truncation or zero-padding is performed. To increase the performance, all the packet bytes are divided to 255, maximum value of a byte, so that all the input values are in $[0-1]$ range. The byte-vectorized packets are then used for training stacked autoencoder or convolutional NN in a supervised fashion, regarding the packets labels. For the test phase all of the aforementioned procedures are done and pre-trained neural networks are used for classification.

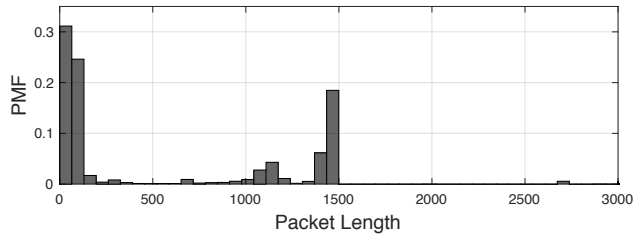



Figure 4: Empirical probability mass function of the packet length shows that almost 96% of the packets have length of less than 1500 bytes.

| Name | Size |
|---|---|
| AIM chat | 5K |
| Facebook | 2502K |
| FTPS | 7872K |
| Gmail | 12K |
| Hangouts | 3766K |
| ICQ | 7K |
| Netflix | 299K |
| SCP | 825K |
| SFTP | 864K |
| Skype | 4549K |
| Spotify | 40K |
| Torrent | 108K |
| Tor | 202K |
| Voipbuster | 807K |
| Vimeo | 146K |
| YouTube | 251K |

(a)

| Name | Size |
|---|---|
| Chat | 733K |
| Email | 31K |
| File Transfer | 319K |
| Streaming | 320K |
| Torrent | 86K |
| VoIP | 320K |
| VPN: Chat | 68K |
| VPN: File Transfer | 148K |
| VPN: Email | 17K |
| VPN: Streaming | 160K |
| VPN: Torrent | 79K |
| VPN: VoIP | 239K |

(b)

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

file, ignoring the VPN or non-VPN condition same as the application identification labeling procedure. This leads to a 12-classes dataset, as shown at Table 2b.

By observing Table 1, one would instantly notice that the dataset is significantly imbalanced and the number of samples vary remarkably among different classes. Sampling, explained at [42], is a simple yet powerful technique to overcome this problem. Using under-sampling method, we randomly remove the major classes' samples (classes having more samples) until the classes are fairly balanced.

### *3.3. Architectures*

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. In order to prevent over-fitting problem, after each layer the dropout technique with 25% dropout probability is applied [43]. 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 categorization tasks, at the final layer of the proposed SAE, a softmax classifier with 17 and 12 neurons is added respectively. Fig. 5a illustrates the architecture of the proposed SAE in a nutshell.

The second proposed scheme, based on one-dimensional CNN is briefly depicted at Fig. 5b. As shown in the figure, the proposed CNN consists of a one dimensional convolu-

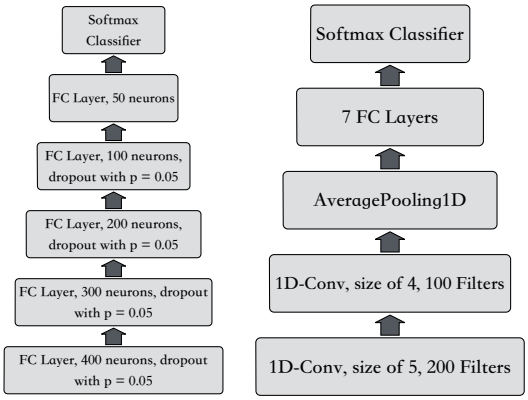



Figure 5: A minimal illustration of the neural networks architectures in the proposed method (a) SAE architecture (b) One-dimensional CNN architecture.

tional layer with 200 filters with kernel size of 5 which is applied on the byte-vactorized packet as input. This layer is followed by another one dimensional convolutional layer with 80 filters of kernel size 4. Afterwards, a one dimensional average pooling layer with kernel size of 2 is applied which is followed by a 25% dropout layer. Afterwards, the output of previous layer is fed into a fully-connected network consisting of seven layers. Finally, a softmax classifier is applied for the classification task, similar to SAE architecture. For a detailed description of one-dimensional CNN refer to Appendix .

## 4. Experimental Results

To implement our proposed NNs, we have used Keras library [44] using Tensorflow [45] backend. Each of the proposed models was trained and evaluated against the independent test set that was derived from dataset. We randomly split the dataset into three separate sets: first one which includes 60% of samples is used for training and adjusting weights and biases; second one containing 20% of samples is used for validation during the training phase; finally the third set made up of 20% of data points is used for testing the model. In order to avoid over-fitting problem, we have used *early stopping* technique. It stops the training procedure, once the loss function value for the validation set remains stationary for several epochs, thus prevent the network to over-fit on the train samples. The procedure of training and testing each network was performed 5 times, with different train, test and validation sets.

For training SAE, first each layer was trained in greedy layer-wise fashion, described at Section 2.3, using *Adam* [46] optimizer and *mean squared error* as the loss function for 200 epochs. At the fine-tuning phase *categorical cross entropy* loss function was employed and the network was trained for another 200 epochs. It is worth to mention that all layers exploit Rectified Linear Unit (ReLU) as the activation function, except for the final softmax classifier layer. As for implementing one dimensional CNN, *categorical cross entropy* and *Adam* were used for loss function and optimizer respectively. The network was trained for 500 epochs, with batch size of 1000 packets.

To evaluate the performance of our proposed methods we have used Recall (Rc), Precision (Pr) and $F_1$ Score ($F_1$) metrics. The formulas for aforementioned metrics are stated as follows:

$$Rc = \frac{TP}{TP + FP}, \quad Pr = \frac{TP}{TP + FN}, \quad F_1 = 2 \cdot \frac{Rc \cdot Pr}{Rc + Pr} \quad (1)$$

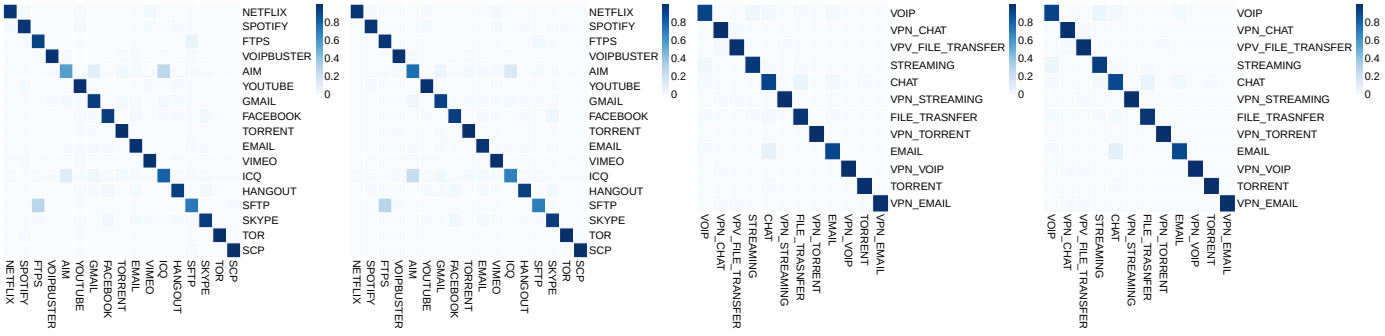
Here TP, FP and FN stands for True Positive, False Positive and False negative, respectively.

Table 2 shows the detailed performance of both SAE and CNN on the application identification task on the test set. The average $F_1$ score of 0.95 for one-dimensional CNN and 0.95 for SAE shows that our networks have completely extracted and learned the discriminating features from the training set and can successfully distinguish each application. One can infer the same from the row-normalized confusion matrices depicted at Fig. 9 in form of heatmaps. The dark color of the elements on the main diagonal suggests that our networks can classify each application with minor confusion. However there are lower $F_1$ scores in both networks for AIM, ICQ and email applications. This is due to the fact that those classes possess fewer samples, hence the NNs were not able to classify them as accurately as the others.

| class name | CNN | | | SAE | | |
|---|---|---|---|---|---|---|
| | Rc | Pr | $F_1$ | Rc | Pr | $F_1$ |
| AIM chat | 0.70 | 0.56 | 0.63 | 0.64 | 0.76 | 0.70 |
| Facebook | 0.95 | 0.95 | 0.95 | 0.95 | 0.94 | 0.95 |
| FTPS | 0.77 | 0.93 | 0.84 | 0.77 | 0.97 | 0.86 |
| Gmail | 0.91 | 0.95 | 0.93 | 0.94 | 0.93 | 0.94 |
| Hangouts | 0.99 | 0.94 | 0.96 | 0.99 | 0.94 | 0.97 |
| ICQ | 0.63 | 0.80 | 0.70 | 0.69 | 0.69 | 0.69 |
| Netflix | 1.00 | 0.98 | 0.99 | 1.00 | 0.98 | 0.99 |
| SCP | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| SFTP | 0.90 | 0.71 | 0.80 | 0.96 | 0.70 | 0.81 |
| Skype | 0.94 | 0.95 | 0.95 | 0.93 | 0.95 | 0.94 |
| Spotify | 0.97 | 0.98 | 0.97 | 0.98 | 0.98 | 0.98 |
| Torrent | 0.99 | 1.00 | 0.99 | 0.99 | 0.99 | 0.99 |
| Tor | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| Voipbuster | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 |
| Vimeo | 0.98 | 0.99 | 0.98 | 0.98 | 0.99 | 0.98 |
| YouTube | 0.98 | 0.99 | 0.99 | 0.98 | 0.99 | 0.99 |
| Average | 0.96 | 0.95 | 0.95 | 0.96 | 0.95 | 0.95 |

Table 2: Application identification results.

On the traffic characterization task, our CNN has achieved $F_1$ score of 0.97, while the proposed SAE has reached 0.97,




(a) Application identification using one-dimensional CNN

(b) Application identification using SAE

(c) Traffic characterization using one-dimensional CNN

(d) Traffic characterization using SAE

Figure 6: Confusion matrices on application identification and traffic characterization tasks for the proposed SAE and one-dimensional CNN. The rows of the confusion matrices correspond to the actual class and the columns present the predicted class, thus the matrices are row-normalized.

implying that both networks are capable of accurately classify packets. Table 4a summaries the proposed methods performance on the test set.

| class name | CNN | | | SAE | | |
|---|---|---|---|---|---|---|
| | Rc | Pr | $F_1$ | Rc | Pr | $F_1$ |
| Chat | 0.92 | 0.92 | 0.92 | 0.92 | 0.91 | 0.91 |
| Email | 0.93 | 0.91 | 0.92 | 0.93 | 0.91 | 0.92 |
| File Transfer | 0.93 | 0.99 | 0.96 | 0.94 | 0.98 | 0.96 |
| Streaming | 0.96 | 0.96 | 0.96 | 0.94 | 0.95 | 0.95 |
| Torrent | 1.00 | 1.00 | 1.00 | 0.99 | 0.99 | 0.99 |
| VoIP | 0.96 | 0.93 | 0.94 | 0.95 | 0.92 | 0.94 |
| VPN: Chat | 0.98 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 |
| VPN: File Transfer | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 |
| VPN: Email | 1.00 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 |
| VPN: Streaming | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| VPN: Torrent | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| VPN: VoIP | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| Average | 0.97 | 0.97 | 0.97 | 0.97 | 0.97 | 0.97 |

Table 3: Traffic characterization results.

As mentioned in Section 1.1, [38] tried to characterize network traffic using time-related features handcrafted from traffic flows. Yamansavascilar et al. [14] exploiteds time-related features to identify end-user application. Their best results can be found at Table 4. It can be observed that Deep Packet has outperformed the aforementioned works, in both application identification and traffic characterization.

## 5. Discussion

By carefully observing the confusion matrices at Fig. 9, one would definitely notice some interesting confusions (e.g., ACQ and AIM). Hierarchical clustering further demonstrates the similarities captured by Deep Packet. Clustering on row-normalized confusion matrix for application identification with one-dimensional CNN (Fig. 6a), using "Euclidean" as the distance metric and *Ward.D* as the clustering method, uncovers similarities among applications in terms of their propensities to be assigned to the 17 application classes. Application groupings revealed by Deep Packet, generally agrees with the applications similarities in real world. Hierarchical clustering divided the applications into 7 groups. Interestingly, these groups are to some extent similar to groups in traffic characterization task. One would notice that Vimeo, Netflix, YouTube and Spotify which are bundled together, are all streaming applications. Streaming applications use network to transfer audio and video files. There is a cluster including ICM, AIM and Gmail. AIM and ICQ are used for online chatting and Gmail 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 contained traffic for aforementioned applications in 3 forms: voice call, video call and chat. Thus the network has found these application similar in terms of 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. Interestingly, SCP (Secure Copy) has formed its own cluster despite is also used for remote file transferring. 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 own clusters which is sensible due to their apparent differences with other applications. Yet this clustering is not flawless. Clustering Skype, Facebook and Hangouts along with Email and VoipBuster is not correct. VoipBuster is an application which offers voice communications over In-

| Paper | Task | Comparison Metric | Results | Algorithm |
|---|---|---|---|---|
| Deep Packet | Application Identification | Accuracy | 95.4% | CNN |
| Yamansavascilar et al. | | | 93.9% | K-NN |
| Deep Packet | Traffic Categorization | Precision | 97.0% | CNN |
| Gil et al. | | | 89.7% | C4.5 |

Table 4: Results comparisson of Deep Packet and other solutions for different tasks.

ternet infrastructure. Thus applications in this cluster do not seem much similar in terms of their usage and this grouping is not precise. The same procedure was also performed on confusion matrix for SAE on application identification task (Fig. 6b), and the outcome is pretty much the same. It can be found at Appendix .

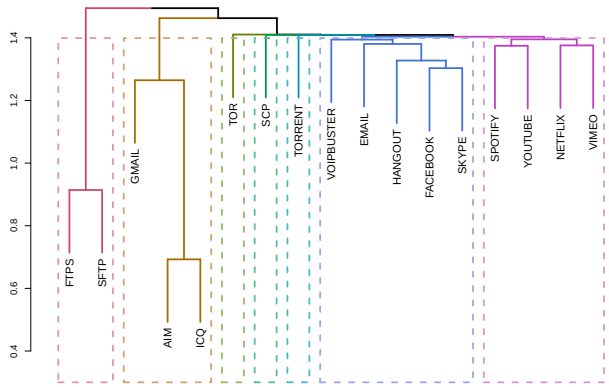



Figure 7: Hierarchical clustering, performed on row-normalized confusion matrix for application identification with one-dimensional CNN, confirms that the proposed one-dimensional CNN has captured the similarities among applications. The suggested clustering mostly consistent with the real-world similarities of applications. For instance Spotify, Netflix, YouTube and Vimeo, which are all streaming applications, formed a cluster.

Most of the applications exploit encryption in order to maintain clients' privacy. The dataset packets are believed to be encrypted. One might wonder how it is possible for Deep Packet to classify such packets. Unlike DPI methods Deep Packet does not inspect the packets for keywords. In contrast, it attempts to learn features in traffic for each application. Consequently it needs not to decrypt the packets to classify them. Ideal encryption technique causes the data to bear most possible entropy [47]. In other words, it produces pattern-less data that theoretically can not be discernible form on another. However due to the fact that all encryption schemes use pseudo-random generators, this hypothesis is not always true in practice. Each application employs its own ciphering scheme for packets encryption. These schemes utilize pseudo-random algorithms which bear distinct patterns. These patterns can be used to distinguish applications from one another. Our network attempts to extract those discriminative patterns and learn them. Hence, it is able to classify encrypted packets.

It is noticeable from Figs. 6a and 6b and Table 2 that Tor traffic was successfully classified. To further investigate this, we conducted another test in which we trained and tested our network with a dataset containing only Tor traffic. Detailed result can be found at Table 5, but it is obvious that our network was unable to classify Tors traffic accurately.

| 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 |
| Average | 0.35 | 0.40 | 0.36 | 0.57 | 0.44 | 0.30 |

Table 5: Tor traffic classification results.

This is not far from what we expected. Tor encrypts its packets, before channeling them. As discussed before, Deep Packet presumably learns pseudo-random patterns used in encryption scheme of each application. At this experiment traffic was tunneled through Tor, hence they all bear the same encryption scheme. Consequently our neural network was not able to tell them apart.

## 6. Conclusion

In this paper, we presented Deep Packet a framework that automatically extracts features from network traffic using deep learning algorithms in order to classify traffic. To the best of our knowledge, Deep Packet is the first traffic classification system using Deep learning algorithms, namely, SAE and one-dimensional CNN that can handle both application identification and traffic categorization tasks. Our results showed that Deep Packet outperforms all of the similar works on "ISCX VPN-nonVPN traffic dataset" both in traffic classification and traffic categorization 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 tasks. Deep Packet can be modified in order to handle more complex tasks

like multi-channel classification, accurate classification of Tor traffic, etc. Finally, the automatic feature extraction procedure from network traffic can save the cost for using experts in order to identify and extract handcrafted features from the traffic which eventually leads to more accurate traffic identification.

## 7. Refrences

## References


[1] S. Bagui, X. Fang, E. Kalaimannan, S. C. Bagui, J. Sheehan, Comparison of machine-learning algorithms for classification of vpn network traffic flow using time-related features, Journal of Cyber Security Technology (2017) 1–19.

[2] A. Dainotti, A. Pescape, K. C. Claffy, Issues and future directions in traffic classification, IEEE network 26 (1).

[3] M. Finsterbusch, C. Richter, E. Rocha, J.-A. Muller, K. Hanssgen, A survey of payload-based traffic classification approaches, IEEE Communications Surveys & Tutorials 16 (2) (2014) 1135–1156.

[4] P. Velan, M. Čermák, P. Čeleda, M. Drašar, A survey of methods for encrypted traffic classification and analysis, International Journal of Network Management 25 (5) (2015) 355–374.

[5] A. W. Moore, K. Papagiannaki, Toward the accurate identification of network applications, in: PAM, Vol. 5, Springer, 2005, pp. 41–54.

[6] A. Madhukar, C. Williamson, 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, 2006, pp. 179–188.

[7] J. Lv, C. Zhu, S. Tang, C. Yang, Deepflow: Hiding anonymous communication traffic in p2p streaming networks, Wuhan University Journal of Natural Sciences 19 (5) (2014) 417–425.

[8] R. Alshammari, A. N. Zincir-Heywood, Can encrypted traffic be identified without port numbers, ip addresses and payload inspection?, Computer networks 55 (6) (2011) 1326–1350.

[9] T. Karagiannis, K. Papagiannaki, M. Faloutsos, Blinc: multilevel traffic classification in the dark, in: ACM SIGCOMM Computer Communication Review, Vol. 35, ACM, 2005, pp. 229–240.

[10] G. D. Gil, A. H. Lashkari, M. Mamun, A. A. Ghorbani, 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), 2016, pp. 407–414.

[11] J. Khalife, A. Hajjar, J. Diaz-Verdejo, A multilevel taxonomy and requirements for an optimal traffic-classification model, International Journal of Network Management 24 (2) (2014) 101–120.

[12] Y. Bengio, Learning deep architectures for ai, Found. Trends Mach. Learn. 2 (1) (2009) 1–127.

[13] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.

[14] B. Yamansavascilar, M. A. Guvensan, A. G. Yavuz, M. E. Karsligil, Application identification via network traffic classification, in: Computing, Networking and Communications (ICNC), 2017 International Conference on, IEEE, 2017, pp. 843–848.

[15] S. H. Yeganeh, M. Eftekhar, Y. Ganjali, R. Keralapura, A. Nucci, Cute: Traffic classification using terms, in: Computer Communications and Networks (ICCCN), 2012 21st International Conference on, IEEE, 2012, pp. 1–9.

[16] T. Auld, A. W. Moore, S. F. Gull, Bayesian neural networks for internet traffic classification, IEEE Transactions on neural networks 18 (1) (2007) 223–239.

[17] Y. Qi, L. Xu, B. Yang, Y. Xue, J. Li, Packet classification algorithms: From theory to practice, in: INFOCOM 2009, IEEE, IEEE, 2009, pp. 648–656.

[18] S. Sen, O. Spatscheck, D. Wang, 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, 2004, pp. 512–521.

[19] J. Sherry, C. Lan, R. A. Popa, S. Ratnasamy, Blindbox: Deep packet inspection over encrypted traffic, in: ACM SIGCOMM Computer Communication Review, Vol. 45, ACM, 2015, pp. 213–226.

[20] M. Iliofotou, H.-c. Kim, M. Faloutsos, M. Mitzenmacher, P. Pappu, G. Varghese, Graption: A graph-based p2p traffic classification framework for the internet backbone, Computer Networks.

[21] E. G. Allan, Jr., W. H. Turkett, Jr., E. W. Fulp, Using network motifs to identify application protocols, in: Proceedings of the 28th IEEE Conference on Global Telecommunications, GLOBECOM'09, IEEE Press, Piscataway, NJ, USA, 2009, pp. 4266–4272.

[22] M. Crotti, M. Dusi, F. Gringoli, L. Salgarelli, Traffic classification through simple statistical fingerprinting, ACM SIGCOMM Computer Communication Review 37 (1) (2007) 5–16.

[23] X. Wang, D. J. Parish, 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, 2010, pp. 98–103.

[24] A. W. Moore, D. Zuev, Internet traffic classification using bayesian analysis techniques, in: ACM SIGMETRICS Performance Evaluation Review, Vol. 33, ACM, 2005, pp. 50–60.

[25] R. Sun, B. Yang, L. Peng, Z. Chen, L. Zhang, S. Jing, Traffic classification using probabilistic neural networks, in: Natural computation (ICNC), 2010 sixth international conference on, Vol. 4, IEEE, 2010, pp. 1914–1919.

[26] H. Ting, W. Yong, T. Xiaoling, Network traffic classification based on kernel self-organizing maps, in: Intelligent Computing and Integrated Systems (ICISS), 2010 International Conference on, IEEE, 2010, pp. 310–314.

[27] A. Moore, D. Zuev, M. Crogan, Discriminators for use in flow-based classification, Tech. rep. (2013).

[28] Z. Wang, The applications of deep learning on traffic identification, BlackHat USA.

[29] M. Caudill, Neural networks primer, part i, AI Expert 2 (12) (1987) 46–52.

[30] C. Beleites, U. Neugebauer, T. Bocklitz, C. Krafft, J. Popp, Sample size planning for classification models, Analytica chimica acta 760 (2013) 25–33.

[31] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al., Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups, IEEE Signal Processing Magazine 29 (6) (2012) 82–97.

[32] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556.

[33] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, C. P. Potts, Recursive deep models for semantic compositionality over a sentiment treebank, in: EMNLP, 2013.

[34] P. Vincent, H. Larochelle, Y. Bengio, P.-A. Manzagol, Extracting and composing robust features with denoising autoencoders, in: Proceedings of the 25th international conference on Machine learning, ACM, 2008, pp. 1096–1103.

[35] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, Greedy layer-wise training of deep networks, in: Advances in neural information processing systems, 2007, pp. 153–160.

[36] D. H. Hubel, T. N. Wiesel, Receptive fields and functional architecture of monkey striate cortex, The Journal of physiology 195 (1) (1968) 215–243.

[37] C. N. dos Santos, M. Gatti, Deep convolutional neural networks for sentiment analysis of short texts, in: Proceedings of the 25th International Conference on Computational Linguistics (COLING), Dublin, Ireland, 2014.

[38] B. Alipanahi, A. Delong, M. T. Weirauch, B. J. Frey, Predicting the sequence specificities of dna-and rna-binding proteins by

deep learning, Nature biotechnology 33 (8) (2015) 831–838.
[39] H. Lee, R. Grosse, R. Ranganath, A. Y. Ng, Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations, in: Proceedings of the 26th Annual International Conference on Machine Learning, ICML '09, ACM, New York, NY, USA, 2009, pp. 609–616.
[40] N. M. K. Chowdhury, R. Boutaba, A survey of network virtualization, Computer Networks 54 (5) (2010) 862–876.
[41] R. Dingledine, N. Mathewson, P. Syverson, Tor: The second-generation onion router, Tech. rep., Naval Research Lab Washington DC (2004).
[42] R. Longadge, S. Dongre, Class imbalance problem in data mining review, arXiv preprint arXiv:1305.1707.
[43] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting., Journal of Machine Learning Research 15 (1) (2014) 1929–1958.
[44] F. Chollet, et al., Keras, `https://github.com/fchollet/keras` (2017).
[45] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems, software available from tensorflow.org (2015).
URL `http://tensorflow.org/`
[46] D. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980.
[47] C. Cachin, Entropy measures and unconditional security in cryptography (1997).

## 8. Appendix

### *8.1. 1D CNN Architecture*

The architecture of one-dimensional CNN is shown at Fig. 8. It is made up of a one-dimensional convolutional layer with 200 filters of size 5. The filters are convolved with the input vector. The output of this layer would be a two-dimensional tensor with size of $1500 \times 200$. This tensor is then fed into the next convolutional layer which has 80 filters each having a $4 \times 200$ shape. Afterwards, a one-dimensional average pooling layer with pooling region of 2 is applied. Dropout technique with probability of 25% is used. Then the output which is a tensor with size of $2 \times 80$ is squashed (flattened) into a one-dimensional vector. This vector is then fed into several fully-connected layers. This fully-connected network consists of seven layers with 600, 500, 400, 300, 200, 100 and 50 neurons respectively. A softmax classifier is employed for performing classification tasks.

### *8.2. Clustering*

#### *8.2.1. Traffic Characterization Task*

We did a hierarchical clustering on row-normalized confusion matrices for traffic characterization, shown in Figs. 6c and 6d. The clustering were performed using "Euclidean" as the distance metric and *Ward.D* as the clustering method. Interestingly two groups revealed by clustering divides the traffic to VPN and non-VPN clusters. All the VPN traffics are bundled together in one cluster, while all of non-VPNs are grouped together.

#### *8.2.2. Application Identification Task*

Hierarchical clustering on Fig. 6b, using "Euclidean" as the distance metric and *Ward.D* as the clustering method, uncovers similarities among applications captured by the SAE. Application groupings are generally sensible with respect to applications similarities in real world. The clustering is depicted at Fig. 10.

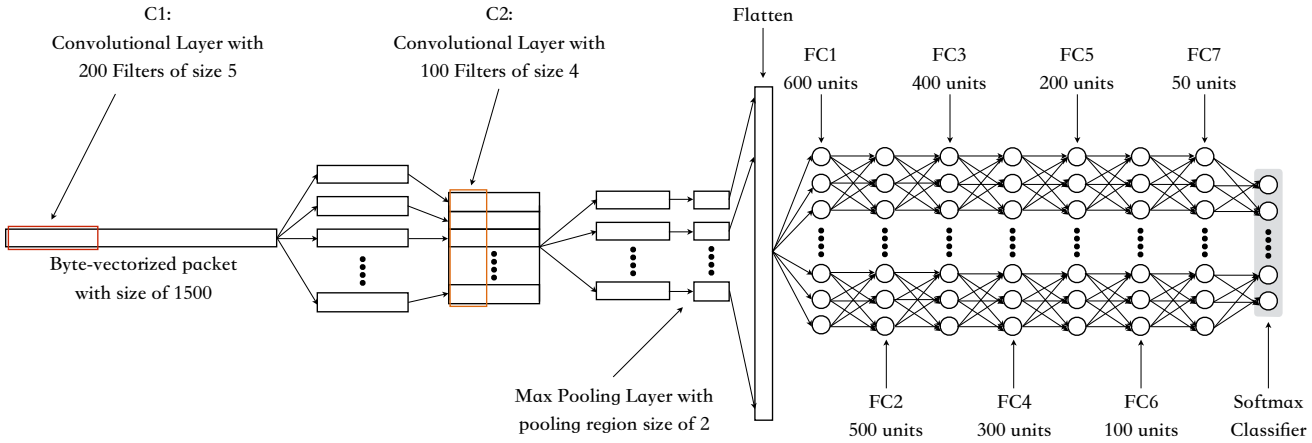
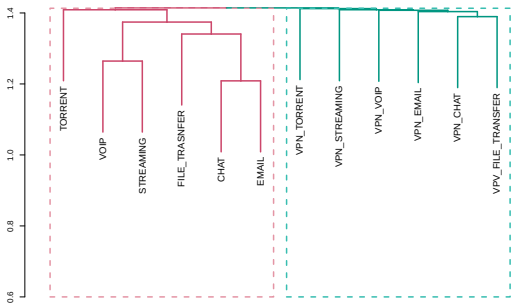
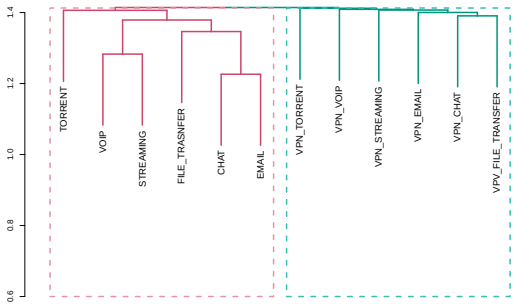



Figure 8: Complete architecture of one-dimensional CNN employed in Deep Packet. The CNN is made up of two one-dimensional convolutional layers stacked on top of each other, followed by a max pooling layer. Afterwards, the output is flattened into a one-dimensional vector and it is connected to a fully-connected network of neurons comprised of seven layers. For classification, a softmax classifier is used at the final layer.




(a) Clustering based on confusion matrix of SAE for traffic characterization task.




(b) Clustering based on confusion matrix of SAE for traffic characterization task.

Figure 9: Hierarchical clustering on row-normalized confusion matrices of SAE (Fig. 9a) and one-dimensional CNN (Fig. 9b). Noticeably all of the VPN traffics are bundled together whereas all of non-VPN traffics are grouped together.

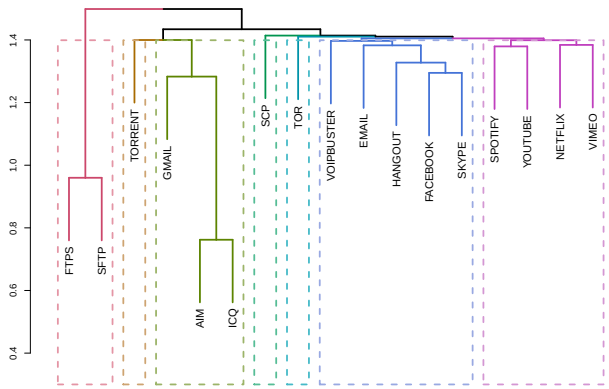



Figure 10: Hierarchical clustering on row-normalized confusion matrix for application identification with SAE. Interestingly FTP-based applications have formed a cluster, while SCP, which is SSH-based, has formed its own cluster.