FlowPic: Encrypted Internet Traffic Classification is as Easy as Image Recognition

Tal Shapira* and Yuval Shavitt†

School of Electrical Engineering, Tel-Aviv University
Email: *talshapira1@mail.tau.ac.il, †shavitt@eng.tau.ac.il

***Abstract*—Identifying the type of a network flow or a specific application has many advantages, but become harder in recent years due to the use of encryption, e.g., by VPN and Tor. Current solutions rely mostly on handcrafted features and then apply supervised learning techniques for the classification.**

We introduce a novel approach for encrypted Internet traffic classification by transforming basic flow data into a picture, *a FlowPic*, and then using known image classification deep learning techniques, Convolutional Neural Networks (CNNs), to identify the flow category (browsing, chat, video, etc.) and the application in use.

We show using the UNB ISCX datasets that our approach can classify traffic with high accuracy. We can identify a category with very high accuracy even for VPN and Tor traffic. We classified with high success VPN traffic when the training was done for a non-VPN traffic. Our categorization can identify with good success new applications that were not part of the training phase. We can also use the same CNN to classify applications with an accuracy of 99.7%. Overall, our approach achieves significant better performance than previous work, and can handle classification problems that were not studied in the past.

I. Introduction

Internet traffic classification has become a popular research field in recent years [1], [2], [3] as it is used for QoS implementations, traffic engineering, law enforcement, and even malware detection. However, due to the growing trends of Internet traffic encryption and an increase in usage of VPNs and Tors, this task is becoming much harder. Most of the presented techniques for classifying encrypted traffic rely on extracting statistical features (also called feature extraction) from a traffic flow. This is followed by a process of feature selection to eliminate irrelevant features, and finally use shallow methods of supervised learning, such as decision trees, SVM (Support Vector Machine), KNN (K-Nearest Neighbors), etc., for the classification.

Over the past few years, advances in deep learning [4] have driven tremendous progress in many fields. Convolutional Neural Networks (CNNs) have Especially brought breakthroughs in the field of image classification [5]. With these advances, the use of CNNs has become common in many domains [4] such as medical uses, sentence classification, visual document analysis, age and gender classification, and genomics. In addition, image classification reached maturity that allows it to be integrated in commonly used products, such as smart phones.

In this paper, we introduce a novel approach for classifying Internet traffic into categories (VoIP, video, file transfer, chat and browsing) and for application identification. We build on the excellent results achieved for image recognition by transforming Internet flows into images. For each flow, our method creates an image from the packet sizes and packet arrival times, we term it a *FlowPic*. These FlowPic images are fed into a CNN that classify them with astonishing high accuracy, e.g., we can classify a traffic category with an accuracy of over 96%, except for browsing (see Table IV). The method is so effective that it classifies traffic that passes through VPN with an accuracy above 99.2%, and achieves good results even for traffic that traverses Tor (over 89% except for file transfer). We further show in Table IV that even when our CNN is trained only with traffic that doesn't pass through VPN or Tor, we can still classify VPN traffic category with a good accuracy: 78.9% to 99.4% depending on the category. Furthermore, we trained the same model with 10 VoIP and video applications and classify them with an accuracy of 99.7%.

Finally, we seem to capture the intrinsic characteristics of a category behavior. We show that even if we train the network on a category while excluding some applications, e.g., by training without Facebook video data, we still manage to classify Facebook video to the correct category, with an accuracy of 99.9%. This means that the appearance of a new application may not break an already trained network.

Our contribution is a generic approach for Internet traffic classification, that takes advantage of all time and size related information available in a network flow, instead of using information from manually extracted features. Moreover, our model can deal with a short time window of a unidirectional flow instead of the entire bidirectional session. We use the exact same architecture for all the experiments in the paper: classification of traffic categories, encryption techniques and application identification - we made no attempt to gain extra accuracy by adapting the architecture to the exact problem.

Another advantage of our approach is that we do not rely on the packet payload content, and thus do not breach privacy. Unlike methods that classify based on the packet payload content [6], [7], [8], [9], our storage requirement is quite minimal, since for each packet we need to transfer only two words of data from the forwarding engine to where the analysis is done, which makes *real time classification* feasible.

978-1-7281-1878-9/19/$31.00 ©2019 IEEE

II. Related Work

Two types of Internet traffic classification problems were recently studied: Internet traffic categorization [1] also called traffic characterization such as VoIP, File Transfer, Video, etc., and Internet application identification [10], [3]. There are many different approaches for solving these classification problems. We choose to divide these approaches into three main categories: 1) Payload based traffic classification methods, also called deep packet inspection (DPI), are problematic because of their invasion of privacy. They are computationally expensive and are incapable of dealing with most of today's traffic due to use of encryption, 2) Port based methods - based on packet headers fields values, where the most commonly is the TCP/UDP port number. Port based methods, which are fast and simple, were widely used in the past, but with the increased use of dynamic ports and default ports, their efficiency declined, and 3) Statistics and machine learning based methods [11] - usually by manually extracting size and time related features and applying complex patterns or supervised learning algorithms as classifiers. In addition, there are also works that present hybrid approaches.

We will focus on describing the most relevant works, basically those that use statistics and machine learning methods. As we mentioned, there are many works that focused on the process of features generations. In 2005, Moore *et al.* [12], [13] created a list, based on a bidirectional flow, containing 248 descriptors such as RTT (round-trip delay time) statistics, size-based statistics, inter-arrival time statistics, frequencies, and so on. They applied a Naive Bayes Kernel estimator to categorize network traffic. Fahad *et al.* [14] chose five well known feature selection (FS) techniques and proposed an integrated FS approach, to obtain an optimal feature set, using the descriptors list of Moore *et al.* [12].

Table I: List of captured protocols and applications for each traffic category and encryption technique.

	Non-VPN	**VPN**	**Tor**
VoIP	Google Hangouts, Facebook, VoipBuster, Skype	Google Hangouts, VoipBuster, Skype	Google Hangouts, Facebook, Skype
Video	Google Hangouts, Facebook, Netflix, Vimeo, YouTube, Skype	Netflix, Vimeo, YouTube	Vimeo, YouTube
File Transfer	FTPS, SCP, SFTP, Skype	FTPS, SFTP, Skype	FTP, SFTP, Skype
Chat	Google Hangouts, Facebook, AIM Chat, Skype, ICQ, WhatsApp Web	Google Hangouts, Facebook, AIM Chat, Skype, ICQ	Google Hangouts, Facebook, AIM Chat, Skype, ICQ
Browsing	Firefox, Chrome	-	Firefox, Chrome

Table II: Number of session blocks in each traffic category and encryption technique.

	Non-VPN	**VPN**	**Tor**
VoIP	3304	2872	2978
Video	1553	302	754
File Transfer	1174	242	1126
Chat	635	1061	422
Browsing	3191	-	2026

There are many works that use only flow-based features (i.e. time and size related features). Gil *et al.* [15] used statistical time-related features such as flow bytes per seconds, inter-arrival time, etc. They generated bidirectional flows of Non-VPN and VPN traffic, applied C4.5 and KNN as classifiers and achieved accuracy levels above 80%. Zhang *et al.* [16], [17] used 20 simple unidirectional flow-based statistical features and applied a bag of words (BoF) technique to model correlation information in traffic flows of the same application. They introduced a new robust traffic classification (RTC) scheme, that used their BoF-based traffic classifier, and showed that their performance is significantly better than the most common machine learning methods.

There are some recent works that involve the use of neural networks. Ertam and Avci [18] used a genetic algorithm (GA) for the selection of features out of 12 attributes, then applied a wavelet kernel based extreme learning machines (ELM) over a dataset that contained 7 classes of regular traffic (non-VPN), and achieved an accuracy over 95%. Lopez-Martin *et al.* [9] used a recurrent neural network (RNN) combined with a CNN to classify traffic based on 6 features for each packet in the session. They achieved an accuracy over 95% using ports information and 84% without ports information, which emphasized the weakness of their method. There are several works that introduced a new approach that are based on the packet payload content. Wang *et al.* [6], [7] converted each packet payload to a normalized byte sequence, and used it as an input for artificial neural networks. Moreover, using 1-D Convolution Neural Networks (CNNs, see Sec. V for more details), They improved the classification results over the ISCX VPN-nonVPN traffic dataset [15], relative to previous works. Lotfollahi *et al.* [8] applied almost the same method using CNNs and stacked auto-encoders over the same dataset, and achieved good performance. Payload based methods work very well on the test data, however because these methods rely on raw data (bytes values), they may be overfitted to the bytes structure of the specific applications, and not be able to generalize the characteristic of internet categories to classify unknown applications. Moreover, these methods can not work in case of using encryption techniques such as VPN and Tor. Wang *et al.* [7] success in VPN classification is due to the usage of training data and test data that are based on the same encryption method and encryption keys. Chen *et al.* [19] converted flow data to a picture of the flow parameter auto-convolution and fed it to a neural network, however, they used side information such as the target IP, and had not described their method sufficiently to enable comparison.

Qin *et al.* [20] were among the first to understand the need to avoid handcrafted feature selection (i.e., manually extracted features), and used instead the payload size distribution (PSD) probability of the packets in a bidirectional flow. Then, they employed the Renyi cross-entropy to identify the similarity between one PSD and that of a specific application. As we show later in Sec. IV-A, in our work we extend the use of PSD and construct a 2-dimensional histogram for flow representation, that is a PSD of the packets for each delta time.

Thus we utilize both time-related and size-related features in the flow.

III. The Dataset

In order to examine our method, we use labeled datasets of packet capture (pcap) files from the Uni. of New Brunswick (UNB): "ISCX VPN-nonVPN traffic dataset" (ISCX-VPN) [15] and "ISCX Tor-nonTor dataset" (ISCX-Tor) [21], as well as our own small packet capture (TAU). **ISCX-VPN** consists of captured traffic with a total of 7 traffic types (VoIP, Chat, etc.) for both regular traffic sessions (Non-VPN) and sessions over VPN. **ISCX-Tor** consists of the same 7 captured traffic traffic types for both regular traffic sessions (Non-VPN) and sessions over Tor. Since the UNB datasets do not contain enough flows for chats, we captured traffic of Whatsapp web chat, Facebook chat and Google Hangout chat. We use a combined dataset only from the five categories that contains enough samples: VoIP, Video, Chat, Browsing, and File Transfer. For these categories we have 3 encryption techniques: non VPN, VPN (for all classes except Browsing) and TOR. Notice that our categories differ slightly from those suggested in [15], [21]. All the applications that were captured in order to create the dataset, for each traffic category and encryption technique, are shown in Table I.

A. Dataset Preparations

The dataset is made of capture files, each corresponds to a specific application, a traffic category and an encryption technique. However, all these captures also contain sessions of different traffic categories, since while performing one action in an application, many other sessions occur for different tasks simultaneously. For example, while using VoIP over Facebook, there is another STUN session taking place at the same time for adjusting and maintaining the VoIP conversation, as well as an HTTPS session of the Facebook site. To prevent these kind of mistakes in our dataset, we keep only the flows that belong to the correct category. We split each pcap file to unidirectional flows, where each flow is defined by a 5-tuple {source IP, source port, destination IP, destination port, protocol}.

B. Data Augmentation

In order to increase the number of examples for the training set, and in order to reduce overfitting, we divide each unidirectional flow to 60 seconds blocks [15]. The total number of session blocks, for each traffic category and encryption technique is shown in Table II.

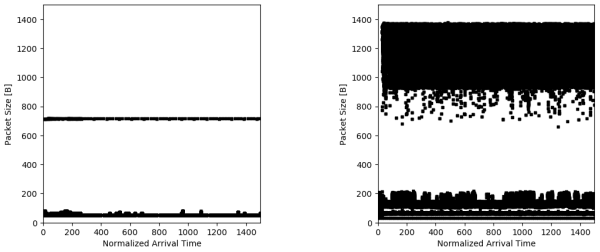
We performed a thorough analysis of the sensitivity of our approach to the FlowPic construction, by running experiments with different block sizes: 15, 30, 60 and 120 Sec for class vs. all tasks (see Sec. VI-A for further information). The differences in the average accuracy were up to 1.25% for different block sizes, which is not significant.

IV. Image Transformation Phase

A. FlowPic Construction

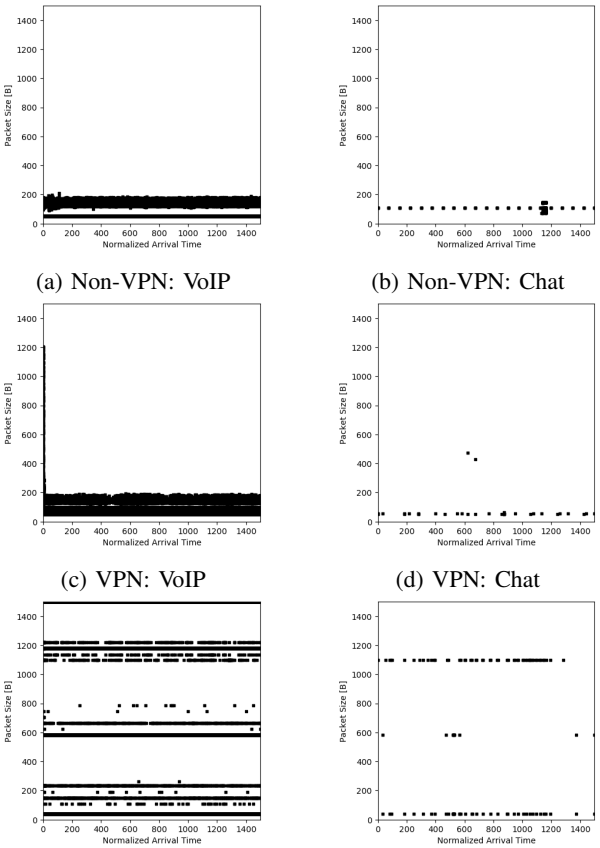
As a pre-processing stage, we extract records from each flow, which comprised of a list of pairs, {IP packet size, time of arrival} for each packet in the flow. Then, we merge all lists of the same traffic category and the same encryption technique to a single set.

Our goal is to construct an image that is built from a flow-based two-dimensional histogram. As mentioned in Sec. II,



(a) Netflix (b) Skype

Figure 1: Examples of FlowPics for Netflix and Skype video applications. Note that for illustration purposes, black pixels represent any value between 1 and 255, while white pixels represent the value 0 (namely, there is no arrival of packets with the corresponding size in the corresponding delta time.).



(a) Non-VPN: VoIP (b) Non-VPN: Chat

(c) VPN: VoIP (d) VPN: Chat

(e) Tor: VoIP (f) Tor: Chat

Figure 2: Examples of FlowPics, where the left column corresponds to VoIP traffic, the right column corresponds to chat traffic, and each row corresponds to a different encryption technique.

this image can be seen as an array of payload size distributions (PSDs) [20], where each PSD belongs to a specific time interval of the unidirectional flow. At the first stage, we plot all record pairs by defining our X-axis as the packet arrival time, and Y-axis as the packet size. An absolute majority of the packet sizes do not cross the 1500 bytes (which is the Ethernet MTU value), thus we disregard all packets with size greater than 1500 (less than 5% of all packets), and limit our Y-axis to be between 1 to 1500. For the X-axis, first we normalize all time of arrival values by subtracting the time of arrival of the first packet in the flow. For simplicity, we set the 2D-histogram to be a square image. For this purpose, we normalized all time of arrival values to be between 0 to 1500 (namely, 60 seconds is mapped to 1500). Then we insert all normalized pairs to a two dimensional histogram, where each cell holds the number of packets that arrive at the corresponding time interval and have the corresponding size. We store each 1500x1500 histogram in an image matrix and term it a *FlowPic*. Later on, we use these images as inputs to our model.

B. FlowPics Exploration

Figure 1 illustrates some difficulties of internet traffic classification. Each category of Internet traffic consists of many applications and services, each of which behaves differently (The examples shown in Figure 1 are for video streaming.). Netflix, for example, transmits packets with almost fixed sizes (depending on the video), while applications such as Skype, Facebook and Google Hangout transmits much widely distributed sizes. A second problem that can be recognized while examining Figure 1, is that a video flow is not limited to display elements only, but also includes audio streaming that behave the same as VoIP, and a small packet streaming for coordination and control that looks like chat transmission. In contrast, on Skype for example, there is a separation between the video flow and the audio flow.

Figure 2 shows FlowPics of two traffic categories passing through different encryption services. It is easy to notice the effect of the choice of encryption technique, on the flow behavior for each traffic category. For example, a chat flow, without any use of VPN or Tor, contains a small number of small sizes low-frequency packets, whereas through a VPN, couple of flows are combined into one session, and whereas through Tor, all flows being transmitted from a user, are combined into one massive encrypted session.

Another noticeable behavior, is that Tor's encryption traffic flow is composed of discrete packet sizes, as opposed to many packet sizes in a non-VPN traffic. This is caused due to the use of a block cipher encryption. Exploring FlowPic images reinforces our initial motivation. While there are some powerful features that are easy to identify, such as the fact that VoIP traffic consists of lots of low-frequency small packets, there are also some unique patterns that can not be explicitly represented by numerical features. Therefore, the use of deep learning methods will result in a significant advantage.

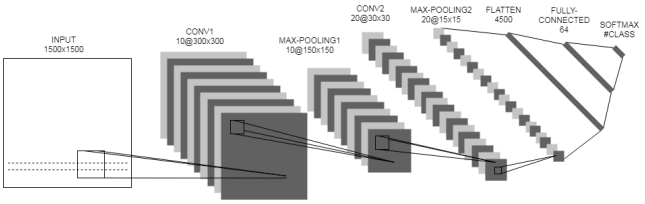


Figure 3: An illustration of the architecture of our LeNet-5 style CNN.

V. Convolutional Neural Networks

A. Background

Convolutional Neural Networks, also known as **ConvNets** or **CNNs**, that were first introduced by LeCun *et al.* [22], have a major role in the field of deep learning [23]. Compared to standard feedforward neural networks with similarly-sized layers, CNNs have much fewer trainable parameters and so they are much easier to train. In additions, CNNs make the explicit assumption that the input comes in the form of multiple arrays (mostly as an image). A typical CNN contains two main layers: *convolutional layers* and *pooling layers*.

The convolutional layer produces layers that are called features map. Each unit in a feature map is connected to a local region of the previous layer through a set of weights called *filter* (or kernel). The result of the convolution in each unit is then passed through an activation function. All units in a feature map share the same filter (known as *parameter sharing*). As a results, the number of filters determines the number of feature maps in the next layer. By making the filter smaller than the input, we need to store much fewer parameters, which both improves the efficiency of the learning and reduces overfitting.

The pooling function replaces the output of a certain layer with a summary statistics of the nearby outputs, in order to reduce the amount of parameters and computations in the network, and hence to also reduce overfitting. *Max-pooling*, which outputs the maximum value in a rectangular neighborhood of the previous layer, is the most popular.

A typical CNN usually ends with a number of fully-connected layers, in which neurons between two adjacent layers are fully pairwise connected. The last fully-connected layer is called the *output layer*. The most common output layer is the *softmax layer*, which outputs a K-dimensional probability distribution vector of real values in the range [0, 1] (K is the number of classes), each value represent a class score.

The training procedure is done by feeding the network with many labeled examples and applying gradient descent learning [24] (also known as back-propagation) to adjust the weights of the network, in order to minimize the cost function.

B. The Architecture

Our goal is to find a single architecture that yields satisfactory results for many kind of internet traffic classification problems. We chose to apply a LeNet-5 style architecture [24],

with some optimization techniques [5]. We use the same architecture for all sub-problems.

As depicted in Figure 3, our LeNet-5 style architecture comprises seven layers, not counting the input, where the ReLU activation function [25] is applied to the output of every convolutional and fully-connected layer. As mentioned before, our input is a 2-dimensional 1500x1500 matrix (pixel image). The first layer is a 2-dimensional convolutional layer (labeled as CONV1) with 10 filters of size 10x10 with a stride of 5 (leading to an overlap of 5). Each neuron in CONV1 is connected to a 10x10 neighborhood in the input. The outputs of CONV1 are 10 feature maps of size 300x300. CONV1 contains a total number of 1,010 trainable parameters (1000 weights and 10 bias parameters).

The next layer is the first max-pooling layer with 10 feature maps of size 150x150, where each unit in each feature map is connected to a 2x2 neighborhood in the corresponding feature map in CONV1. Layer CONV2 is the second convolutional layer with 20 filters of size 10x10 with a stride of 5, contains a total number of 20,020 trainable parameters. The outputs of CONV2 are 20 feature maps of size 30x30. The next layer is the second 2x2 max-pooling layer results with 20 feature maps of size 15x15. The next layer is a standard flatten layer that converts the 20 feature maps to a one dimensional layer of size 4500. Our next layer is a fully-connected layer of size 64, contains a total number of 288,064 trainable parameters. Finally, our output layer is the softmax layer whose size depends on the number of classes in each sub-problem. The last layer contains $65 \times m$ trainable parameters, where m is the number of classes. Note that the parameter m is the only difference in the architecture between the different problems"

C. Training Specifications

In order to reduce overfitting, in addition to the data augmentation procedure, we use the *dropout* [26] technique to prevent complex co-adaptations on the training data. This is done during the training time by randomly setting to zero the output of each neuron with a predefined probability. During the test evaluation we use all neurons, but multiply their output by the 'dropout' probability. In our CNN (Figure 3), we use dropout with a probability of 0.25 in CONV2 and dropout with a probability of 0.5 in the fully-connected layer.

The training was done by optimizing the *categorical cross entropy* [27] cost function, that is a measure of the difference between the softmax layer output and a one-hot encoding vector of the same size, representing the true label of the sample. For the optimization process we use the *Adam* [28] gradient-based optimizer. The Adam optimization algorithm is an extension to the stochastic gradient descent algorithm. We used the default hyper-parameters as provided in Kingma *et al.* [28] and set our batch size to 128.

We build and run our networks using the *Keras* [29] library with *Tensorflow* [30] as its backend. In order to compare reliably between all sub-problems results, we run our network for 40 epochs (which took between 5 to 10 minutes for an epoch) of 10 batches each, and save the result which achieve the best accuracy during the training process. In all experiments, our CNN reaches convergence after running on 10 to 25 epochs.

Table III: A summary of our results for different traffic classification problems, with comparison to best known previous results over the ISCX dataset. It is important to note that these are not accurate comparisons, due to the use of different categories, different classification entities, unbalanced datasets, etc.

Problem	FlowPic Acc. (%)	Best Previous Result	Remark
Non-VPN Traffic Categorization	85.0	84.0 % Pr., Gil *et al.* [15]	Different categories. [15] used unbalanced dataset
VPN Traffic Categorization	98.4	98.6 % Acc., Wang *et al.* [7]	[7] Classify raw packets data. Not including browsing category
Tor Traffic Categorization	67.8	84.3 % Pr., Gil *et al.* [15]	Different categories. [15] used unbalanced dataset
Non-VPN Class vs. All	97.0 (Average)	No previous results	
VPN Class vs. All	99.7 (Average)	No previous results	
Tor Class vs. All	85.7 (Average)	No previous results	
Encryption Techniques	88.4	99. % Acc., Wang *et al.* [7]	[7] Classify raw packets data, not including Tor category
Applications Identification	99.7	93.9 % Acc., Yamansavascilar *et al.* [10]	Different classes

VI. Experiments and Results

In this section we report our experimental results. Due to the lack of standard datasets, the comparison of our results to previous works (Table III) is challenging. Even the few papers that used the same datasets as we did [15], [21], [7], [10] are not always directly comparable due to selection of categories, evaluation criteria, etc. Moreover, unlike previous works, we created balanced datasets for reliable evaluation. We will discuss these differences while presenting the results.

A. Labeling Datasets for Different Problems

After creating the pre-processed dataset as mention before, we generate sub-dataset for each sub-problem. Table II shows that the dataset is imbalanced, a problem that can lead to a poor classification performance. Thus, we created **a balanced dataset** for each sub-problem. For class vs. all datasets, the specific class is equal to the number of samples in all others classes together, such that the ratio between the quantities of the other classes remains constant. We do it using a *random undersampling* method [31] in order to preserves the initial distribution of samples in each class.

*1) **Multiclass**:* We examine three kinds of multiclass classification problem:

a. *Traffic categorization* - consists of an equal number of samples for all traffic categories that were mentioned

before. We create multiclass datasets for 3 encryption techniques: non-VPN, VPN and Tor.

b. *Multiclass encryption techniques* - consists of 3 classes representing the encryption techniques; Non-VPN (contains only regular sessions), VPN and Tor.

c. *Application identification* - in order to demonstrate the strength of our method, we create a dataset consists of 10 classes representing VoIP and video applications over non-VPN encryption technique, as listed in Table I.

*2) **Class vs. All:*** A class vs. all dataset consists of samples of the specific traffic category, and an equal number of samples of all other traffic categories with an equal share. As done in the multiclass traffic categories datsets, we construct class vs. all datasets for 3 encryption techniques: non-VPN, VPN (for all classes except Browsing) and TOR.

As mentioned before, each of the above sub-problems was trained using its own training set and evaluated using its own test set. We randomly split each sub-problem datatset; we use 90% of the samples as a training set and 10% of the samples as a test set.

B. Evaluation Criteria

We use the **accuracy** criteria to evaluate our model performance, which is defined as the proportion of examples for which the model produces the correct output of all predictions made. A formal definition of the accuracy for multiclass classification is

$$Accuracy = \frac{\sum_{i \in classes} TP_i}{\sum_{i \in classes}(TP_i + FP_i)},$$

where TP_i and FP_i are the true positive and the false positive of the class i, respectively. For visualizing the results of the multiclass problems, we use the normalized **confusion matrix** (Figures 4,5). In a confusion matrix, each row represents the actual class while each column represents the predicted class. In a normalized confusion matrix, each diagonal value represents the *recall* of the corresponding class, defined by $Rc = \frac{TP}{TP+FN}$ (where FN is the false negative).

C. Results on Traffic Categorization Problems

A summary of the results for all multiclass classification problems, as described in Sec. VI-A1, is presented in Table III. We discuss here the first four rows, which contain the results of the traffic categorization problems.

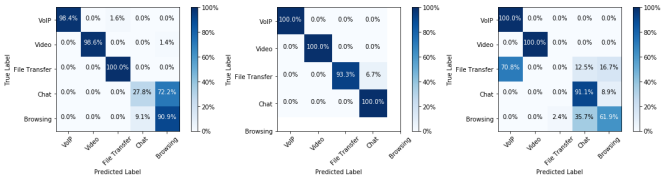
The most noticeable result is that it is significantly more difficult to classify correctly internet traffic categories over Tor, as the last leads to an accuracy of **67.8%**. Classification over VPN and Non-VPN achieves better performances: an accuracy of **98.4%** and **85.0%**, respectively.

As mentioned in Sec. II, the UNB group [15], [21] used time-related features to categorize traffic over the same ICSX datasets, and gained a best average *precision* (which defined by $Pr = \frac{TP}{TP+FP}$) of 84.0% for Non-VPN traffic categorization, 89.0% for VPN traffic categorization and 84.3% for Tor traffic categorization. However, they report the best results from two algorithms (C4.5, random forest); and for each dataset, results from a different algorithm was reported. Remember, that all of our results (for all problems) use the exact same CNN architecture (with different trained weights). Wang *et al.* [7] use the first 784 bytes of each flow to categorize traffic over the ISCX VPN-nonVPN dataset and achieved a best accuracy of 83.0% and 98.6% for non-VPN and VPN traffic, respectively, using different representations. Note, that they did not include the browsing category, citing difficulties in separating it from other categories. Fig. 4 shows that the difficulty in distinguishing between browsing and chat was the main cause of our accuracy degradation.

This experiment achieves an accuracy of **88.2%** for the Non-VPN test set, **98.4%** for the VPN test set and **67.8%** for the Tor test set. These results imply that the unique characteristics of different traffic categories over Tor are quite different from the characteristics of these categories over VPN or non-VPN traffic, and sometimes even completely different (such as for file transfer).

Table IV: Class vs. all classification accuracy performances. For each class, our CNN was trained over a training set of a certain encryption technique, and was tested on 3 test sets, each consists of samples with one of the above encryption techniques: Non-VPN, VPN and Tor.

Class	Accuracy (%)			
VoIP	Training/Test	Non-VPN	VPN	Tor
	Non-VPN	**99.6**	99.4	48.2
	VPN	95.8	**99.9**	58.1
	Tor	52.1	35.8	**93.3**
Video	Training/Test	Non-VPN	VPN	Tor
	Non-VPN	**99.9**	98.8	83.8
	VPN	54.0	**99.9**	57.8
	Tor	55.3	86.1	**99.9**
File Transfer	Training/Test	Non-VPN	VPN	Tor
	Non-VPN	**98.8**	79.9	60.6
	VPN	65.1	**99.9**	54.5
	Tor	63.1	35.8	**55.8**
Chat	Training/Test	Non-VPN	VPN	Tor
	Non-VPN	**96.2**	78.9	70.3
	VPN	71.7	**99.2**	69.4
	Tor	85.8	93.1	**89.0**
Browsing	Training/Test	Non-VPN	VPN	Tor
	Non-VPN	**90.6**	-	57.2
	VPN	-	-	-
	Tor	76.1	-	**90.6**



(a) Non-VPN (b) VPN (c) Tor

Figure 4: Confusion matrices of the 3 traffic categorization problems; over non-VPN, over VPN, and over Tor.

D. Results on Class vs. All Problems

In many cases, there is a need to distinguish a single traffic category from the rest. To the best of our knowledge, we are the first to report such results. Table IV shows a summary

of the results of class vs. all classification problems, by comparing different traffic categories over different encryption techniques, as described in Table I. For each traffic category, our CNN was trained over 3 training sets according to different encryption techniques. Each one of the trained networks was tested on 3 test sets, consist of samples from the trained class with one of the above types of encryption techniques: Non-VPN, VPN and Tor.

By averaging the results of each one of the encryption techniques, taking into account only the values of the diagonals for all traffic categories (where the test set consists of the same traffic category and encryption technique as the training set), we get an average accuracy of **97.0%** for non-VPN traffic, **99.7%** for VPN (not including browsing traffic, as mentioned before) and **85.7%** for Tor. By taking the average of the above averages we get a total average accuracy of **93.7%**. Clearly, except for identification of file transfer over Tor, our method succeeds well in characterizing and identifying different categories of Internet traffic that pass through different encryption techniques.

1) Classification of an Unknown Application: We show that our method is independent of a specific application characteristics. For this purpose, we create a video vs. all dataset, while excluding all Vimeo and YouTube samples from the training set. Although Vimeo and YouTube FlowPics are both totally different than all other videos, our network achieves an accuracy of **83.1%** over a test set consists of Vimeo and YouTube samples and an equal number of non-video samples. We repeat the process, but now we exclude all Facebook samples instead, and achieve an accuracy of **99.9%**. This can be explained by the fact that Facebook FlowPic looks similar to those of Google Hangouts and Skype. We repeat the same procedure to create a VoiP vs. all dataset while excluding all Facebook samples from the training set. We achieve an accuracy of **96.3%**. This is the first paper to report this type of results.

E. Results on Encryption Techniques Classification

Our network achieves an accuracy of **88.4%** classifying encryption techniques, regardless of the specific traffic category. In our results, all confusions occurs between Non-VPN and VPN, while Tor is almost hermetically separated with a recall of **97.7%** (and a precision of **100.0%**).

As already mentioned in Sec. II, the UNB group [15], [21] used time-related features to categorize traffic over the ISCX datasets. However, they separated the problem into two: classifying VPN from non-VPN and classifying Tor from non-Tor. They achieved a precision of 89.0% for VPN [15]; and 94.8% for Tor [21]. Note that while our CNN worked in harder conditions (we used both datasets together), we reached better results in both cases.

Wang *et al.* [7] use the first 784 bytes of each flow to categorize traffic; they only used the VPN and non-VPN datasets. Not surprisingly, they reached an accuracy of 99.9%, since the VPN encryption changes the payload data.

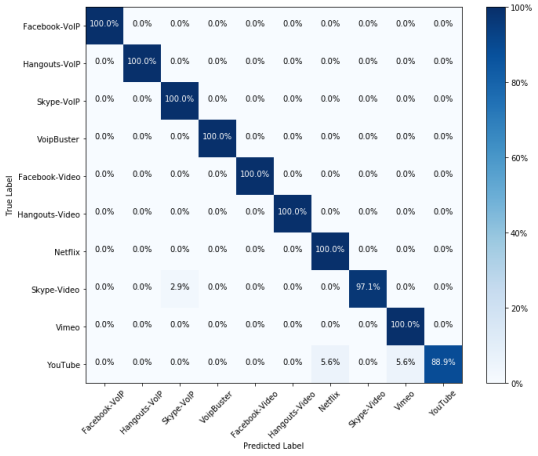


Figure 5: A confusion matrix of the VoIP and video applications identification problem.

F. Results on Application Identification

We tested the performances for classifying 10 classes of VoIP and Video applications over non-VPN traffic. Figure 5 shows that our method almost completely separates the various applications, resulting in an accuracy of **99.7%**. This result, on top of the previous results, demonstrates the ability of our approach to identify the unique strong features of Internet traffic flows.

Yamansavascilar *et al.* [10] constructed 111 flow features and used k-NN algorithm to achieve an overall accuracy of 93.9% classifying 14 classes of applications over the ISCX VPN-nonVPN dataset. For example, they achieved an accuracy of 45.1%, 80.10% and 86.30% classifying Skype-VoIP, Vimeo and YouTube, respectively. Using our method for classifying 10 VoIP and Video applications traffic over the ISCX dataset, we achieved an overall accuracy of 99.7%, and an accuracy (as defined in Sec. VI-B) of 99.7% ,99.4% and 98.9% classifying Skype-VoIP, Vimeo and YouTube, respectively.

In summary, a CNN with a non-complex architecture, achieves great results in a relatively short evaluation time for a variety of traffic classification problems.

VII. Conclusion

In this paper, we introduce a novel approach for encrypted internet traffic classification, both for categorizing traffic types and for identifying specific applications, based only on time and size related information. We showed that our method generically captures traffic characterization without overfitting a specific application. The cost of using only flow-based records is low in terms of memory resources, storage and running times, and therefore practical for deployment. In addition, our model relies only on a short time window of a unidirectional flow, and does not require reliance on the entire bidirectional session in order to successfully classify internet traffic. Note that one can improve classification by classifying several time windows (possibly with partial overlapping) and

use voting for better classification and vote spreading for classification confidence.

The key insight of our approach is the transformation of Internet traffic flows into FlowPic images. From this point, we take advantage of current advances in the field of image recognition using deep learning methods, and design a CNN architecture based on the LeNet-5 [24] to successfully classify our FlowPics. As we show, flow-based features are fairly immune to encryption techniques such as Tor or VPN, so our approach is able to distinguish between different Internet traffic categories that pass through different encryption techniques, and even distinguish between encryption techniques regardless of the traffic category itself.

Finally, as we demonstrate over the ISCX VPN-nonVPN [15] and ISCX Tor-nonTor [21] datasets, our method has the ability to successfully classify different applications of a particular traffic category, and also capable of identifying traffic category of an unfamiliar application, by learning enough samples of other application of the same traffic category.

We made no effort to optimize the CNN architecture, and simply used one that is known to work well for image recognition. Examining other known architectures may improve results, or simplify the training process. We can also optimize our CNN by changing hyperparmeters such as the number of layers, layer sizes, activation functions, etc. Furthermore, we can also reduce run-time and memory consumption by playing with input parameters like block length and the resolution of the number of bytes in a packet. For example, the input size to the system can be reduced to a coarser matrix of 300x300 by reducing the packet size resolution. Furthermore, we can also reduce the number of bits per pixel by using binning. At the extreme we can create use a single bit per pixel creating binary images, such that black pixels represent an arrival of at least one packet with the corresponding size at the corresponding delta time, while white pixels represent that there is no arrival of packets.

REFERENCES

[1] A. Dainotti, A. Pescape, and K. C. Claffy, "Issues and future directions in traffic classification," *IEEE Network*, vol. 26, no. 1, pp. 35–40, January 2012.

[2] P. Velan, M. Čermák, P. Čeleda, and M. Drašar, "A survey of methods for encrypted traffic classification and analysis," *International Journal of Network Management*, vol. 25, no. 5, pp. 355–374, 2015.

[3] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges," *IEEE Transactions on Network and Service Management*, 2019.

[4] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[6] Z. Wang, "The applications of deep learning on traffic identification," *BlackHat USA*, 2015.

[7] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," in *2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*, July 2017, pp. 43–48.

[8] M. Lotfollahi, R. S. H. Zade, M. J. Siavoshani, and M. Saberian, "Deep packet: A novel approach for encrypted traffic classification using deep learning," *CoRR*, vol. abs/1709.02656, 2017.

[9] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Network traffic classifier with convolutional and recurrent neural networks for internet of things," *IEEE Access*, vol. 5, pp. 18 042–18 050, 2017.

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

[11] T. T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE Communications Surveys Tutorials*, vol. 10, no. 4, pp. 56–76, Fourth 2008.

[12] A. Moore, D. Zuev, and M. Crogan, "Discriminators for use in flow-based classification," Queen Mary Uni. of London, Tech. Rep., 2005.

[13] A. W. Moore and D. Zuev, "Internet traffic classification using bayesian analysis techniques," in *ACM SIGMETRICS*, ser. SIGMETRICS '05, 2005, pp. 50–60.

[14] A. Fahad, Z. Tari, I. Khalil, I. Habib, and H. Alnuweiri, "Toward an efficient and scalable feature selection approach for internet traffic classification," *Computer Networks*, vol. 57, no. 9, pp. 2040 – 2057, 2013.

[15] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and 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 - Volume 1: ICISSP,*, INSTICC. SciTePress, 2016, pp. 407–414.

[16] J. Zhang, Y. Xiang, Y. Wang, W. Zhou, Y. Xiang, and Y. Guan, "Network traffic classification using correlation information," *IEEE Trans. on Parallel and Distributed Systems*, vol. 24, no. 1, pp. 104–117, Jan. 2013.

[17] J. Zhang, X. Chen, Y. Xiang, W. Zhou, and J. Wu, "Robust network traffic classification," *IEEE/ACM Trans. Netw.*, vol. 23, no. 4, pp. 1257–1270, Aug. 2015.

[18] F. Ertam and E. Avci, "A new approach for internet traffic classification: GA-WK-ELM," *Measurement*, vol. 95, pp. 135 – 142, 2017.

[19] Z. Chen, K. He, J. Li, and Y. Geng, "Seq2img: A sequence-to-image based approach towards ip traffic classification using convolutional neural networks," in *2017 IEEE International Conference on Big Data (Big Data)*, Dec 2017, pp. 1271–1276.

[20] T. Qin, L. Wang, Z. Liu, and X. Guan, "Robust application identification methods for p2p and voip traffic classification in backbone networks," *Knowledge-Based Systems*, vol. 82, pp. 152 – 162, 2015.

[21] A. H. Lashkari, G. D. Gil, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of tor traffic using time based features," in *Proceedings of the 3rd International Conference on Information Systems Security and Privacy - Volume 1: ICISSP,*, INSTICC. SciTePress, 2017, pp. 253–262.

[22] Y. L. Cun, O. Matan, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jacket, and H. S. Baird, "Handwritten zip code recognition with multilayer networks," in *10th International Conference on Pattern Recognition*, vol. ii, Jun. 1990, pp. 35–40 vol.2.

[23] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[24] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[25] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ser. ICML'10, 2010, pp. 807–814.

[26] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *CoRR*, vol. abs/1207.0580, 2012.

[27] D. Campbell, R. A. Dunne, and N. A. Campbell, "On the pairing of the softmax activation and cross–entropy penalty functions and the derivation of the softmax activation function," in *8th Australian Conference on Neural Networks*, Melbourne, Australia, 1997, pp. 181–185.

[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[29] F. Chollet *et al.*, "Keras," https://github.com/fchollet/keras, 2015.

[30] M. A. et al., "Tensorflow," https://www.tensorflow.org/, 2015.

[31] S. Kotsiantis, D. Kanellopoulos, P. Pintelas *et al.*, "Handling imbalanced datasets: A review," *GESTS International Transactions on Computer Science and Engineering*, vol. 30, no. 1, pp. 25–36, 2006.