This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2021.3071441, IEEE Transactions on Network and Service Management

FlowPic: A Generic Representation for Encrypted Traffic Classification and Applications Identification

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, such as, traffic engineering, or to detect and prevent application or application types that violate the organization's security policy. The use of encryption, such as VPN, makes such identification challenging. 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 and application identification by transforming basic flow data into an intuitive picture, a *FlowPic*, and then using known image classification deep learning techniques, CNNs, to identify the flow category (browsing, chat, video, etc.) and the application in use.

We show that our approach can classify traffic with high accuracy, both for a specific application, or a flow category, even for VPN and Tor traffic. Our classifier can even identify with high success new applications that were not part of the training phase for a category, thus, new versions or applications can be categorized without additional training.

***Index Terms*—Internet traffic classification, applications identification, security management, image recognition, Convolutional Neural Networks.**

I. Introduction

INTERNET traffic classification has become a very popular research field in recent years [1], [2], [3], [4] 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 TOR, this task is becoming much harder. Most of the current 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 [5] have driven tremendous progress in many fields. Convolutional Neural Networks (CNNs) have especially brought breakthroughs in the field of image classification [6], which is the leading field of CNNs research. With these advances, the use of CNNs has become common in many domains [5] 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 does not 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 train the same model with 10 VoIP and video applications and classify them with an accuracy of 99.7%, and we train again with 5 VoIP and video applications that pass through VPN and classify them with an accuracy of 100.0%. Chen *et al.* [7] also suggested using a flow picture and feeding it to a CNN. Their picture captures the auto-convolution of the flow parameters and is not intuitive to a human eye.

Finally, it seems we 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 or a new version 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 (non-VPN, VPN, or Tor) 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 [8], [9], [10], [11], 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 *near real-time classification* feasible.

The rest of the paper continues as follows. After describing related work in Sec. II, we describe the dataset in Sec. III. Sec. IV describes the creation of the FlowPics and their basic structure. Sec. V describes the CNN architecture and training. Sec. VI presents our experiments and their results. Finally, the last two sections conclude the paper and discuss future research directions.

II. Related Work

Three types of Internet traffic classification problems were recently studied: 1) Internet traffic categorization [12] also called traffic characterization such as VoIP, File Transfer, Video etc., 2) Internet application identification [13], [14] such as Facebook, YouTube, Skype etc., and 3) user actions identification [15] in a specific application like sending text message on apple iMessage [16] or watching a specific video on YouTube [17], [18].

There are many different approaches for solving these classification problems, which can be divided into three main categories: 1) Payload based traffic classification methods [19], [20], also called deep packet inspection (DPI). These methods 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 [21] - 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 [22], which combine a classifier based on the well-known port numbers, packet payload signatures, and more. We will focus on describing the most relevant works, basically those that use statistics and machine learning methods.

There are many works that focused on the process of features generations. In 2005, Moore *et al.* [23] 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. At the same year, Moore and Zuev [24] applied a Naive Bayes Kernel estimator using their discriminators to categorize network traffic. Fahad *et al.* [25] investigated the task of feature selection (FS). They chose five well known FS techniques and proposed an integrated FS approach, that used all the others, to obtain an optimal feature set. They evaluated their techniques on the descriptors list of Moore *et al.* [23]. While they admitted that their integrated FS technique was computationally more expensive than the others, they provided increased robustness and performance that led to a significant gain in accuracy.

There are many works that use only flow-based features (i.e. time and size related features). Gil *et al.* [26] 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.* [27], [28] 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 [29] 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.* [11] 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 of 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.* [8], [9] 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 details), they improved the classification results over the ISCX VPN-nonVPN traffic dataset [26], relative to previous works. Lotfollahi *et al.* [10] applied almost the same method using CNNs and 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.* [9] 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.* [7] converted flow data to a picture of the flow parameter auto-convolution and fed it to a neural network, however, they did not describe their method sufficiently to enable comparison. They showed results for two experiments: classification of 5 protocols (FTP, HTTP, SSH, TFTP, and TLSV) and classification of 5 applications. The latter classification got an accuracy far below ours, but on a different, unpublished dataset.

After we published our initial results [1], several additional papers looked at related problems. Among them are Zhang *et al.* [30], which proposed an autonomous model update scheme to be able to handle new applications, and Pacheco *et al.* [31], which emulated satellite communications and presented a framework to classify heterogeneous Internet traffic with deep learning techniques for this type of communication.

Aceto *et al.* [4] reproduced different DL techniques and set into a systematic framework for comparison using mobile

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

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

encrypted traffic, and concluded that DL algorithms indeed constitute a promising approach, but yet to reach the maturity level of DL in other fields. Following their last work, Aceto *et al.* [32] introduced a multimodal deep learning framework for mobile encrypted traffic classification, named *MIMETIC*. However, their framework uses both payload data and protocols fields for the classification. Iliyasu and Deng [33] addressed the challenges associated with establishing ground truth labels of large encrypted traffic datasets and therefore introduced a semi-supervised approach using DCGAN. Their approach achieved good accuracy with a very small number of labeled samples.

A recent work by Hardegen *et al.* [34] proposed a flow data stream processing pipeline to solve a new type of problem. They trained deep learning models on NetFlow data to predict flow characteristics that can be used for network traffic optimization.

Qin *et al.* [35] 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 time interval. 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) [26] and "ISCX Tor-nonTor dataset" (ISCX-Tor) [36], as well as our own small packet capture (TAU) [1]. **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 [26], [36]. 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.

In order to prevent these kinds of mistakes in our dataset, we keep only the flows that belong to the correct category by manually removing sessions, which obviously did not match the label category, e.g., UDP packets within a TCP stream, or removing packets that are destined to IP addresses of other known services. As already mentioned, our dataset consists of pcap files, each file is labeled according to the corresponding application, encryption technique, and traffic 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}. We made our dataset publicly available[1].

B. Data Augmentation

In order to increase the number of samples for the training set, and in order to reduce overfitting, we divide each unidirectional flow to equal size blocks. In our experiments, we split each session to 60-second blocks [26] with an overlapping time of 45 seconds (A difference of 15 seconds between the beginning of two adjacent blocks). Our method is drawn from the use of object bounding box masks [37] (also known as sliding windows) as is done in object detection tasks. For example, 10 minutes of a VoIP conversation consists of 10 non-overlapping session blocks or 37 overlapping session blocks,

[1] https://www.eng.tau.ac.il/~shavitt/FlowPic.htm

each block is slightly different than the other. It should be noted, that the data augmentation process comes after splitting all sessions to a training set and a test set, ensuring that there is no overlapping in a single session between a training block and a test block. The total number of overlapping 60-second session blocks, for each traffic category and encryption technique is shown in Table II.

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

C. Sensitivity Analysis

To evaluate the effect of the data augmentation process on the results, we compared the results of class vs. all tasks (see Sec. VI-A for further information) for data with only non-overlapping 60-second session blocks with the overlapping-session dataset described above. By training on overlapping samples, we don't achieve any improvement for classes with many non-overlapping samples; VoIP, Video and Browsing; However, we achieve an improvement of 0.5% in the classification accuracy for chat and file transfer, which do not contain enough non-overlapping samples.

We performed a thorough analysis of the sensitivity of our approach to the *FlowPic* construction, by running experiments with different block sizes. We tested our method over non-overlapping block sizes of 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. For the rest of the paper, we choose a block size of 60 seconds.

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, this image can be seen as an array of payload size distributions (PSDs) [35], 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.

The result of this process is a 1500x1500 histogram, where the sum of the values in the histogram is equal to the total number of packets in the original time window (without those we ignored). We store each 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 multiple 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: there are categories that their FlowPics behave totally different between different encryption techniques. For example, as can be noticed in Figure 2, 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. When a VPN or Tor flow is a mixture of more than one flow, we assume that the labeled flow is sufficiently dominant, such that the other flows can be viewed as noise.

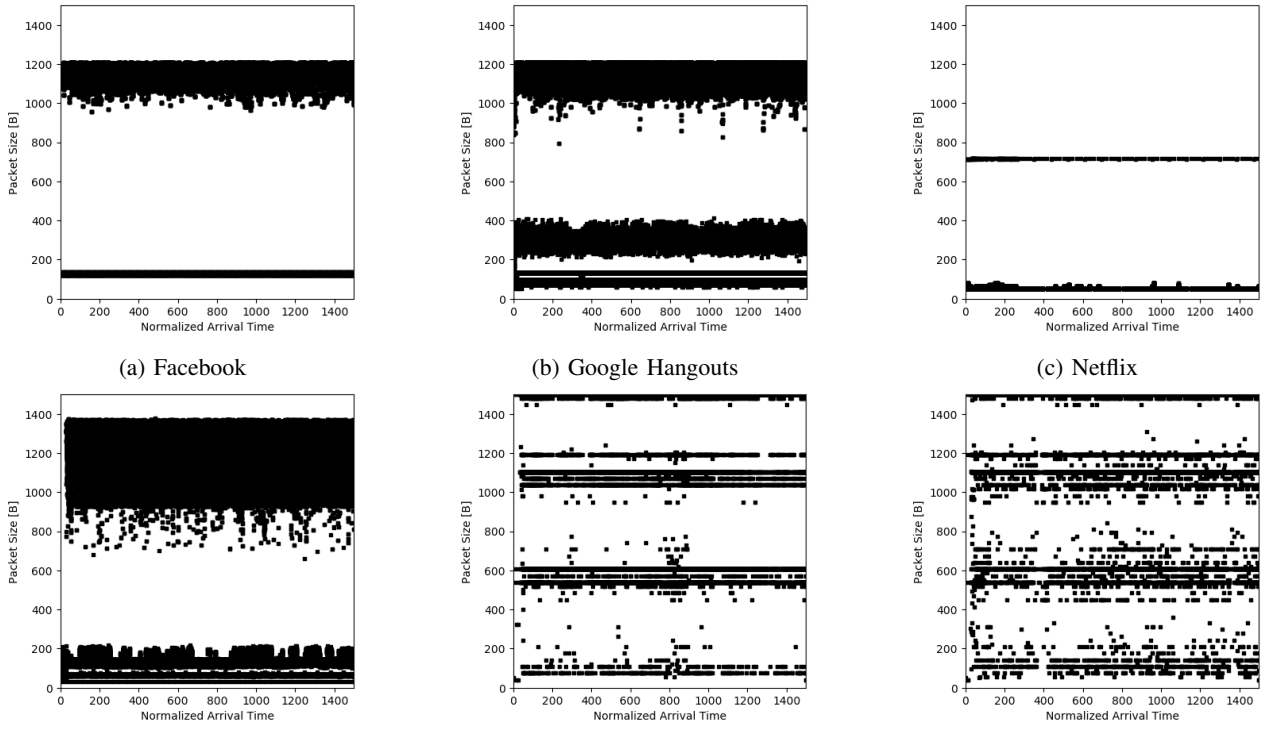
Another noticeable behavior, is that a Tor's encryption traffic flow is composed of discrete packet sizes, as opposed to many packet sizes in a non-VPN traffic. This is caused due to the use of a block cipher encryption. Exploring FlowPic images reinforces our initial motivation. While there are some powerful features that are easy to identify, such as the fact that VoIP traffic consists of lots of high-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.

V. Convolutional Neural Networks

A. Background

Convolutional Neural Networks, also known as **ConvNets** or **CNNs**, that were first introduced by LeCun *et al.* [38],

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2021.3071441, IEEE Transactions on Network and Service Management



(a) Facebook (b) Google Hangouts (c) Netflix

(d) Skype (e) Vimeo (f) YouTube

Figure 1: Examples of FlowPics for several 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.).

have a major role in the field of deep learning [39]. Compared to standard feedforward neural networks with similarly-sized layers, CNNs have much fewer trainable parameters and so they are much easier to train.

The convolutional layer produces layers that are called features maps. 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. Parameter sharing makes the representation approximately invariant to small translations of the input, causes the layer to have a property called *equivariance* to translation. By making the filter smaller than the input (an attribute called *local receptive fields*), 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 (also referred to as *spatial sub-sampling*). Its function is to reduce the spatial size of the representation, in order to reduce the amount of parameters and computations in the network, and hence to also reduce overfitting. The most popular pooling function is the *max-pooling* [40], which outputs the maximum value in a rectangular neighborhood of the previous layer.

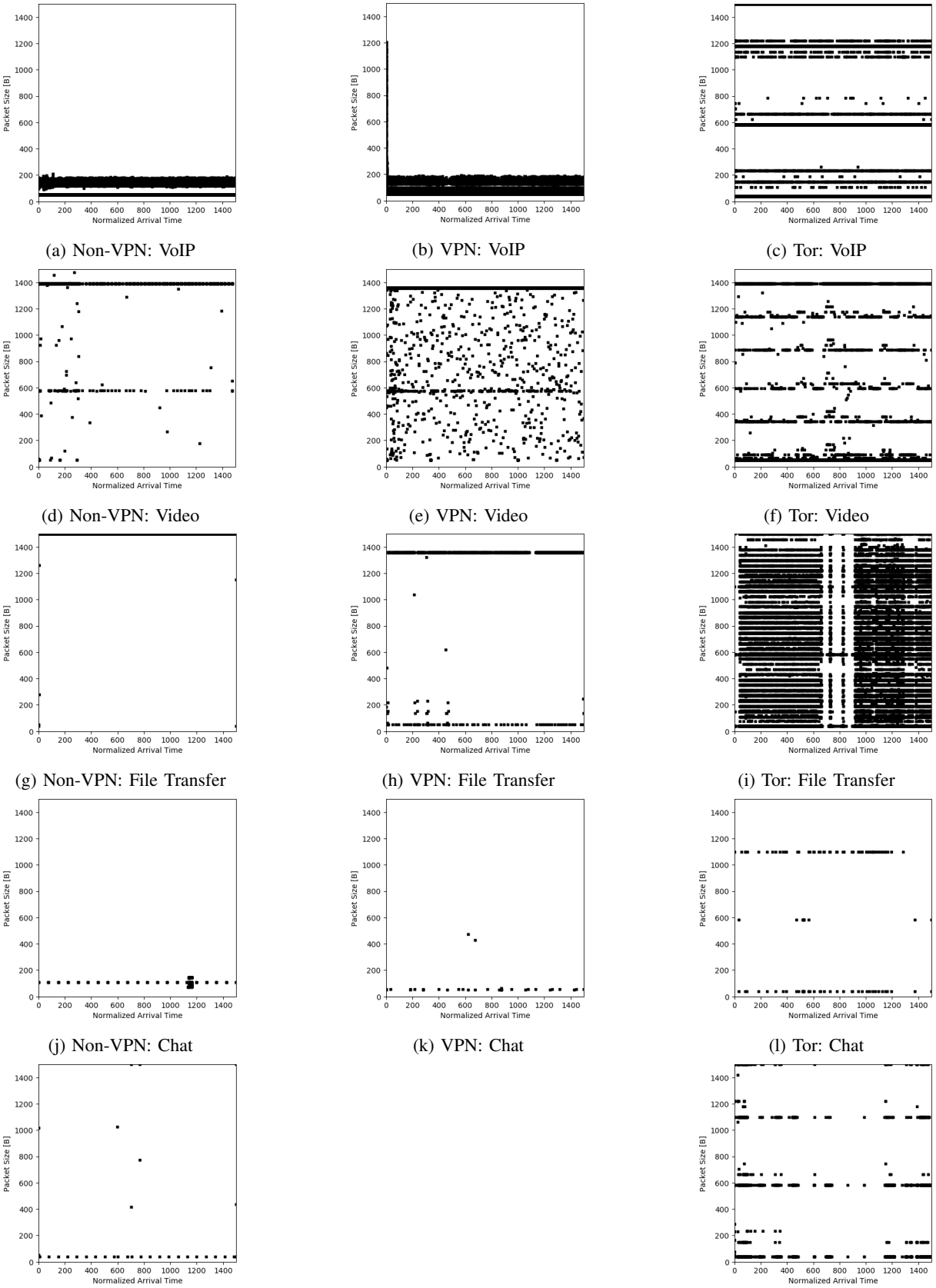
A typical CNN usually ends with several *fully-connected* layers, in which neurons between two adjacent layers are fully pairwise connected, where the last fully-connected layer is called the *output layer*. The most common output layer is the *softmax layer*, which is a generalization of the logistic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

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 [41], with some optimization techniques [6]. 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 [42] 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).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2021.3071441, IEEE Transactions on Network and Service Management



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

(d) Non-VPN: Video (e) VPN: Video (f) Tor: Video

(g) Non-VPN: File Transfer (h) VPN: File Transfer (i) Tor: File Transfer

(j) Non-VPN: Chat (k) VPN: Chat (l) Tor: Chat

(m) Non-VPN: Browsing (n) Tor: Browsing

Figure 2: Examples of FlowPics, where each row corresponds to a different traffic category and each column corresponds to a different encryption technique. Note that for illustration purposes, black pixels represent any value between 1 and 255, while white pixels represent the value 0 (A value of 0 meaning that there is no arrival of packets with the corresponding size in the corresponding delta time.).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2021.3071441, IEEE Transactions on Network and Service Management

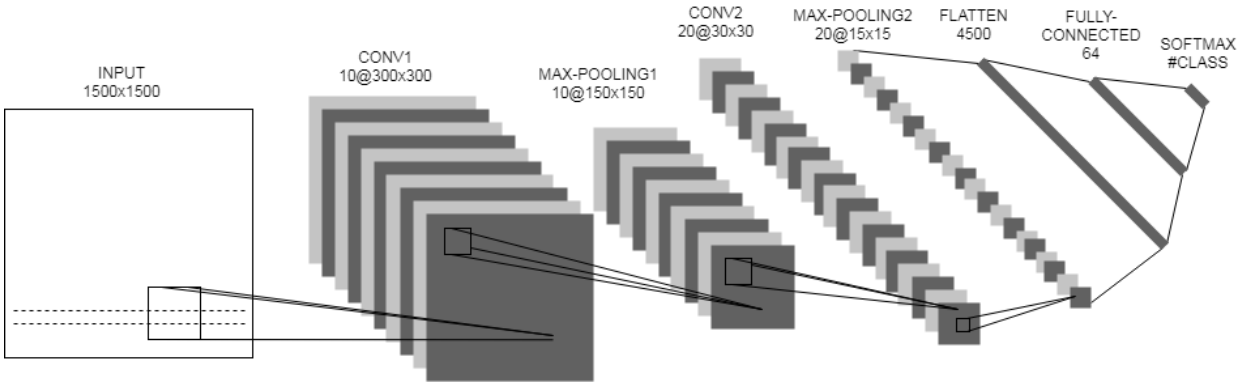


Figure 3: An illustration of the architecture of our LeNet-5 style CNN. The network consists of an input of 1500x1500 image, following by two convolutional blocks stacked on top of each other, each block consists of a convolutional layer followed by a max-pooling layer. Next, the output is flattened into a one-dimensional vector and it is connected to a fully-connected layer. The ReLU activation function is applied to the output of every convolutional and fully-connected layer. We use dropout in the CONV2 and the fully-connected layer, and softmax as the final layer for classification.

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 classification sub-problem: 2 for Class vs. All, 5 for multiclass traffic categorization, 3 for encryption techniques multiclass classification and 10 for the multiclass application identification task. 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* [43] 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. This prevents the network from counting on the presence of a particular neuron during trainning, and therefore reduces overfitting. 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* [44] 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* [45] gradient-based optimizer. The Adam optimization algorithm is an extension to the stochastic gradient descent algorithm. Adam has recently been widely used in many deep learning applications because it efficiently achieves better results than other optimization algorithms. We used the default hyper-parameters ($\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 10^{-8}$) as provided in Kingma *et al.* [45] and set our batch size to 128.

We build and run our networks using the *Keras* [46] library with *Tensorflow* [47] as its backend. In order to compare reliably between all sub-problems results, we run our network for 40 epochs of 10 batches each, and save the result which achieve the best accuracy during the training process (which is commonly used as an *"early stopping"* technique). We trained all of our networks for 40 epochs, which took between 5 to 10 minutes for an epoch. As demonstrated in Figure 4, our network attains convergence after running on 10 - 15 epochs for the multiclass traffic categorization over VPN. More important, the test accuracy curve reaches saturation, in addition to a small variance between the learning curve and the test curve, which ensure we haven't reached overfitting and the stability of our results. In all experiments, our CNN reaches convergence after running on 10 to 25 epochs.

Using our method during test over a newly collected packet trace will cause an end-to-end latency, between the traffic capture and the classification, of $T_{BS} + T_{FC} + T_{ML}$, where T_{BS} is the customized block size (15, 30 or 60 Sec.), T_{FC} is the FlowPic construction time and T_{ML} is our CNN running time to perform classification. In our experiments, we have found that both T_{FC} and T_{ML} are 0.1 Sec., which are negligible compare to the block size.

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 [26], [36], [9], [13] are

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2021.3071441, IEEE Transactions on Network and Service Management

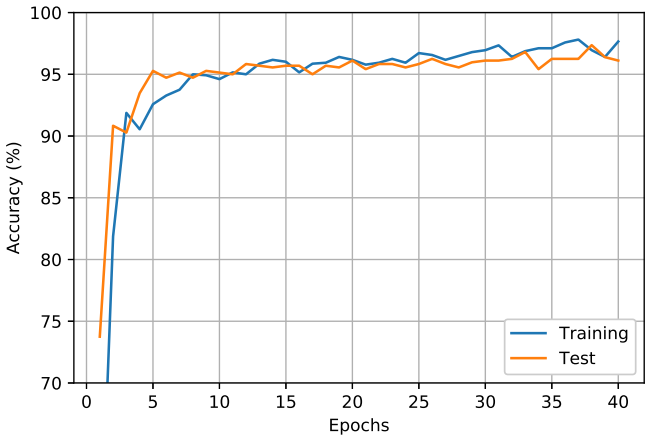


Figure 4: An example of training and test accuracy curves as a function of the number of epochs, of our CNN running for multiclass traffic categorizations over VPN. Each epoch consists of 1280 samples. Convergence is attained after 10-15 epochs.

not always directly comparable due to selection of categories, evaluation criteria, etc. Moreover, unlike previous works, we created balanced datasets for reliable evaluation in most of our experiments, but also report evaluation on imbalanced datasets. 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, since usually classifiers seek an accurate performance over a full range of instances, and therefore tend to classify most of the data into the majority class [48]. Thus, we created **a balanced dataset** for each sub-problem, by maintaining equality in quantities such that the number of samples in each class is equal to the number of samples in another class. 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 [48] with predefined distribution, by randomly removing samples from major classes, until the dataset become balanced. This preserves the initial distribution of samples in each class. Moreover, we can now reliably compare the method's performance for different traffic categories.

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. In addition, we create (we are the first to do this) a *merged* dataset in order to classify traffic, regardless of the encryption technique. Our motivation for creating the merged dataset is to examine how the encryption technique influences the traffic behavior. As we will show, flow-based information has a good representation of traffic types, such that our method can generically identify a traffic category with no dependence of the encryption technique. The merged dateset contains an equal number of sessions of each encryption technique for each traffic category.

b. *Multiclass encryption techniques* - consists of 3 classes representing the encryption techniques; Non-VPN (contains only regular sessions), VPN and Tor. For each of the above classes we create an equally balanced merged dataset containing all internet traffic types.

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, and two more datasets consists of 5 VoIP and video applications over VPN and 5 VoIP and video applications over ToR, as listed in Table I. Figure 1 illustrates differences between FlowPic images of video applications. We choose VoIP and video applications because of the large amount of examples we have for the training process, and the large number of applications .

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 well for a merged datasets. Each traffic category merged dateset contains an equal number of sessions for each encryption technique.

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** criterion to evaluate our model performance, since it is one of the most common evaluation criteria in the field of deep learning. For the imbalanced dataset evaluation, we also use the macro F1-score since it equalizes the contribution of all classes. Accuracy is defined as the proportion of samples for which the model produces the correct output of all predictions made. Classified test samples are divided into four categories:

- *True Positive (TP)* - the number of correctly classified positive samples.
- *True Negative (TN)* - the number of correctly classified negative samples.
- *False Positive (FP)* - the number of negative samples incorrectly classified as positive samples.
- *False Negative (FN)* - the number of positive samples incorrectly classified as negative samples.

Therefore, a formal definition of the accuracy for multiclass classification is

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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2021.3071441, IEEE Transactions on Network and Service Management

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. This is due to the use of different categories, different classification entities, imbalanced datasets, etc.

Problem	**FlowPic Acc. (%)**	**Best Previous Result**	**Remark**
Non-VPN Traffic Categorization	Balanced - 85.0 Imbalanced - 93.8 (89.7 Pr.)	84.0% Pr., Gil *et al.* [26]	Different categories [26], used imbalanced dataset
VPN Traffic Categorization	Balanced - 98.4 Imbalanced - 97.6	98.6% Acc., Wang *et al.* [9]	[9] Classify raw packets data
Tor Traffic Categorization	Balanced - 67.8 Imbalanced - 86.9 (66.3 Pr.)	84.3% Pr., Gil *et al.* [26]	Different categories [26], used imbalanced dataset
Traffic Categorization over Merged Dataset	Balanced - 83.0	No previous results	
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	Balanced - 88.4	99.0% Acc., Wang *et al.* [9]	[9] Classify raw packets data, not including Tor category
Applications Identification	Balanced - 99.7 Imbalanced - 94.2	93.9% Acc., Yamansavascilar *et al.* [13]	Different classes
Applications Identification over VPN	Balanced - 100.0	No previous results	

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 5, 6, 7). 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 $RC = \frac{TP}{TP+FN}$. Furthermore, we use the top-2 categorical accuracy as a second criteria for the multiclass classification problems. Top-K categorical accuracy (or top K-error rates) [49] is widely used for evaluation of multiclass classification problems with a large number of classes.

The Macro F1-score, which we used for the imbalanced dataset evaluation, is defined as the average of the F1 scores across all classes, where the F1-score is defined for each class by: $F1 = 2 * \frac{PR*RC}{PR+RC}$ and the *precision* is defined by $PR = \frac{TP}{TP+FP}$.

C. Traffic Categorization Problems

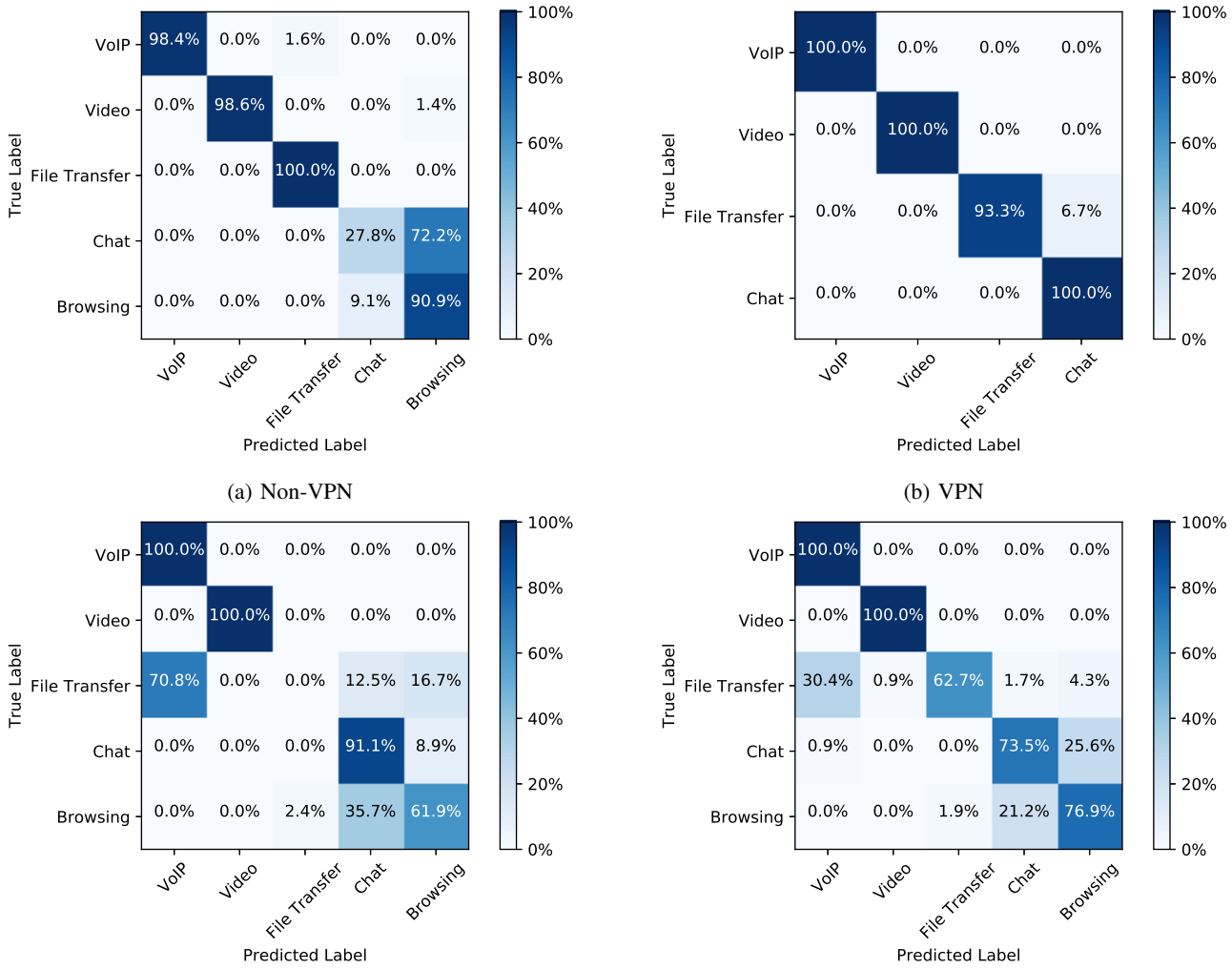
A summary of the results for all multiclass classification problems, as described in Sec. VI-A1, is presented in Table III. We discuss here the first four rows, which contain the results of the traffic categorization problems.

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. Note that for the imbalanced datasets our results are significantly better. Additional results on imbalanced datasets appear in Sec. VI-G and Table VI.

Table IV: Class vs. all classification accuracy performances for VoIP, Video, File Transfer, Chat and Browsing. 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 of the specific class with one of the following encryption techniques: Non-VPN, VPN and Tor. The diagonal values (in bold) represent the accuracy when the test set consists of the same traffic category and encryption technique as the training set.

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**

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2021.3071441, IEEE Transactions on Network and Service Management



(a) Non-VPN

(b) VPN

(c) Tor

(d) Merged

Figure 5: Confusion matrices of the 4 traffic categorization problems; over non-VPN, over VPN, over Tor and a merged dataset. The rows of the confusion matrices correspond to the true label for each class, and the columns correspond to the predicted label for each class.

Table V: Class vs. All classification accuracy performances for *merged datasets* of VoIP, Video, File Transfer, Chat and Browsing . Each traffic category merged dataset contains an equal number of sessions for each encryption technique.

Class	**Accuracy (%)**
VoIP	99.7
Video	99.9
File Transfer	77.4
Chat	89.3
Browsing	90.2

As mentioned in Sec. II, the UNB group [26], [36] 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.* [9] 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. 5 shows that the difficulty in distinguishing between browsing and chat was the main cause of our accuracy degradation.

The confusion matrix in Figure 5c shows that the reason for the performance degradation in classifying Internet traffic categories over Tor, is that our model failed to characterize file transfer traffic over Tor. We later report that even distinguishing file transfer from all the others, is hard over Tor traffic (see Table IV). Second thing to notice is that, apart from the results for Tor traffic categorization, a top-2 accuracy of **81.0%**, our

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2021.3071441, IEEE Transactions on Network and Service Management

model achieves a top-2 accuracy between **95%** to **100%** for all other multiclass classification tasks. As can be inferred from the confusion matrices in Figure 5, our network is sometimes confused between chat sessions and browsing sessions, since both contain a very small number of low-frequency packets, which makes them difficult to distinguish even by a human eye (Figure 2).

Furthermore, we apply the network that was trained using the merged training set (and achieve an accuracy of **83.0%** for the merged test set) over 3 test sets: Non-VPN, VPN and Tor. We do it in order to show how learning the generic behavior of each traffic category, regardless of the encryption technique, is reflected in the performance on each one of the encryption techniques. We are the first to report results from training all encryption techniques together. 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).

D. 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.

Examining the results, there are quite large differences between the characteristic of each one of the traffic categories using different encryption techniques. As a results, learning to identify a certain traffic category that passes through a certain encryption service, doesn't guarantee the ability to identify the same traffic category (and even the same application) through another encryption service. Not surprisingly, by comparing the results from Table IV, it is clear that Tor posses a much harder challenge than VPN.

Moreover, it can be easily deduced that the behavior of a traffic session over Tor, is much closer to the behavior of the same traffic type over VPN than regular (Non-VPN) traffic, and vice versa. This insight is consistent with the fact that Tor adds additional encryption layers, which envelop each other.

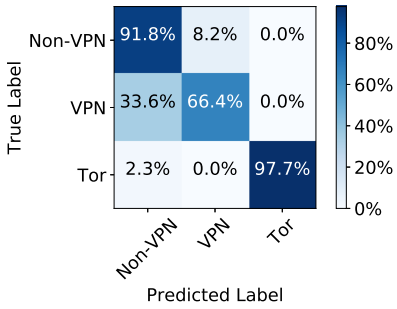


Figure 6: A confusion matrix of the multiclass encryption techniques classification problem.

Table V presents the results for class vs. all problems over merged datasets. The results show that our method is able to well identify the unique generic features of each traffic category regardless of the encryption technique. These results also show that there is a certain difficulty in identifying file transfer traffic, that can be explained by noticing at Figure 2 that a FlowPic of file transfer over Tor is totally different than a FlowPic of file transfer over non-VPN or VPN. This is due to the reason that the file transfer traffic that was examined, usually results with uniform packet size flows, whereas over Tor, many flows are combined to form a merged flow with different packet sizes.

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 FlowPics look similar to those of Google Hangouts and Skype (see Figure 1). 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. 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 [26], [36] 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 [26]; and

94.8% for Tor [36]. Note that while our CNN worked in harder conditions (we used both datasets together), we reached better results in both cases.

Wang *et al.* [9] 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.

F. Application Identification

We test the performances for classifying VoIP and Video applications over non-VPN traffic. Figure 7 shows that our method almost completely separates the various applications, resulting in an accuracy of **99.7%**. We repeat the same experiment for classifying 5 VoIP and Video applications over VPN traffic (The dataset had only have 5 applications over VPN with sufficient number of samples), and achieve an accuracy of **100.0%** (see Figure 8). We do not succeed to classify 5 classes representing VoIP and video applications over Tor, achieving only an accuracy of 44.3%. However, we do succeed to separate between the VoiP applications and the video applications almost hermetically. These results, 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.* [13] 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. A similar experiment using an imbalanced dataset achieves an accuracy of 94.2%.

G. Imbalanced Datasets

Up until now, we reported results of our models on balanced datasets since in many cases that we examined, they give more weight to less frequent classes, which are harder to classify. However, it is also important to test our results on the imbalanced dataset to see how the model performs in such a scenario, especially since previous works mostly looked at this scenario.

We evaluate four problems: Non-VPN traffic categorization, VPN traffic categorization, Tor traffic categorization, and application identification (10 VoIP and VPN classes). For evaluating each task, we conducted 5-fold stratified cross-validation while making sure that each session appears either in the training or the test set. Then, we split each session into 15-second non-overlapped blocks.

In our evaluation, we compared *FlowPic* with 5 ML models that have been widely used for Internet traffic classification in the literature [21], [26]: Decision Trees (DT), k-Nearest-Neighbors (k-NN), Support Vector Machine (SVM), Naive Bayes classifier (NB), and a two-layer neural network (MLP). We used the python *scikit-learn* package [50] to run the ML models with default parameters. We evaluate each of the ML models using the following 10 statistical flow-based features as input since these features have been widely used in previous works [7], [23], [24], [25], [26], [36]:

- Packet size statistics: mean, minimum, maximum, and standard deviation of packet sizes.
- Inter-arrival time statistics: mean, minimum, maximum, and standard deviation of time differences.
- Flow Bytes per second (BPS).
- Flow packets per second (PPS).

We extract these features for each 15-second block. Notice that since our comparison is based only on unidirectional flow, we include only features for one direction.

A summary of the results is presented in Table VI. As shown, except for one task, *FlowPic* outperforms the other algorithms both in terms of accuracy and macro F1-score. For the non-VPN (i.e., regular) traffic categorization, *FlowPic* achieves the second-best score, sightly below *DT*. This can be explained by the significant imbalance between the categories, as 'VoIP' has 2423 samples, while 'chat' has only 245 samples, which is about 10 times less.

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. Concluding remarks

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 behind 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 [41] 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 or a merged dataset, and even distinguish between encryption techniques regardless of the traffic category itself.

Finally, as we demonstrate over the ISCX VPN-nonVPN [26] and ISCX Tor-nonTor [36] 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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2021.3071441, IEEE Transactions on Network and Service Management

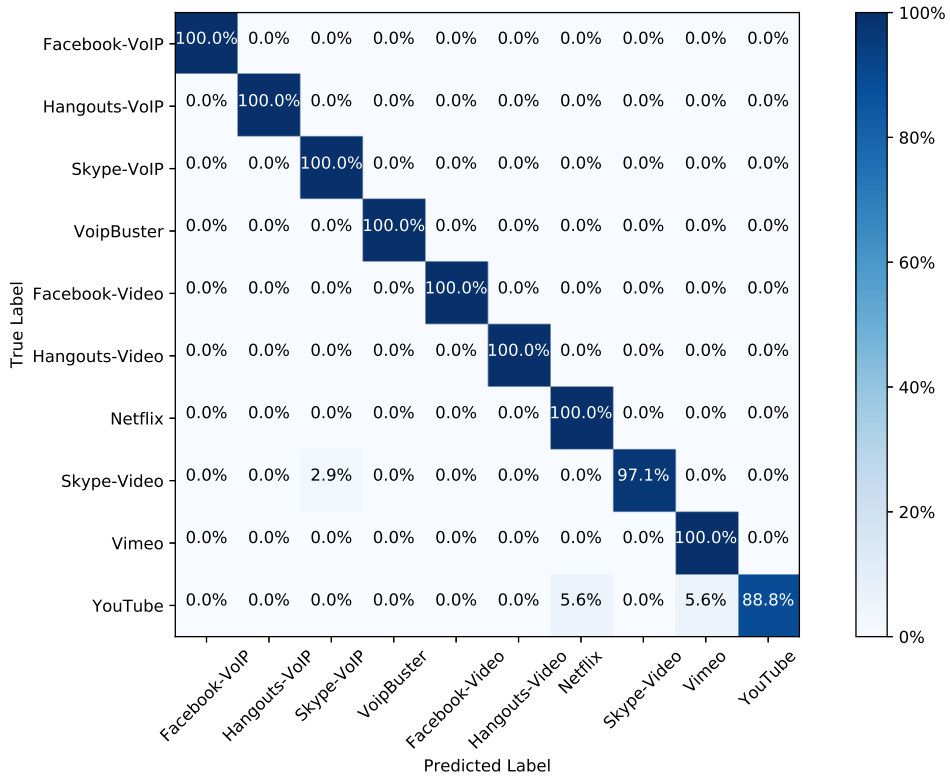


Figure 7: A confusion matrix of the VoIP and video applications identification problem.

Table VI: Comparison of *FlowPic* with common ML algorithms using imbalanced datasets for the following problems: Non-VPN traffic categorization (non-VPN), VPN traffic categorization (VPN), Tor traffic categorization (Tor), and application indetigication (Apps).

Problem	FlowPic		DT		KNN		SVM		NB		MLP	
	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1
Non-VPN	0.9376 ± 0.0284	0.8340 ± 0.0551	**0.9422 ± 0.0115**	**0.8532 ± 0.0196**	0.8777 ± 0.0089	0.7138 ± 0.0222	0.5187 ± 0.1171	0.3443 ± 0.0537	0.5187 ± 0.1171	0.3443 ± 0.0537	0.8403 ± 0.0538	0.6388 ± 0.1002
VPN	**0.9759 ± 0.0117**	**0.9170 ± 0.0466**	0.9492 ± 0.0130	0.8664 ± 0.0457	0.9162 ± 0.0133	0.7588 ± 0.0568	0.9154 ± 0.0202	0.6711 ± 0.0359	0.5249 ± 0.1850	0.4017 ± 0.0719	0.9320 ± 0.0510	0.8018 ± 0.1133
Tor	**0.8694 ± 0.0678**	**0.6493 ± 0.1051**	0.7755 ± 0.0870	0.6331 ± 0.0888	0.6203 ± 0.0753	0.4434 ± 0.0508	0.5196 ± 0.0577	0.2298 ± 0.0471	0.5841 ± 0.0411	0.2359 ± 0.0426	0.5874 ± 0.1467	0.4357 ± 0.0783
Apps	0.**9422 ± 0.0561**	**0.9355 ± 0.0573**	0.9180 ± 0.0677	0.9091 ± 0.0621	0.7964 ± 0.0667	0.7404 ± 0.0529	0.4230 ± 0.0409	0.2625 ± 0.0480	0.1880 ± 0.0832	0.1225 ± 0.0460	0.5189 ± 0.0711	0.4135 ± 0.0501

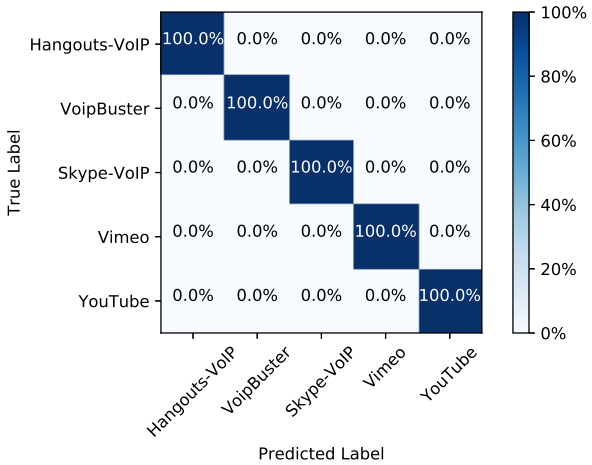


Figure 8: A confusion matrix of the VoIP and video applications identification over VPN Traffic.

enough samples of other application of the same traffic category.

VIII. Future Research Directions

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 hyper-parameters 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 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 time interval.

While our 15 seconds classification time is certainly not problematic in most cases, it may still be a limiting factor for some applications. Thus, an interesting research direction would be to look for methods that would produce a faster classification. While there are already several published solutions, we believe there is still room for improvement.

ACKNOWLEDGMENT

This research was funded in part by a grant on cyber research from the Israeli PMO and the Blavatnik Interdisciplinary Cyber Research Center at Tel Aviv University.

REFERENCES

[1] T. Shapira and Y. Shavitt, "Flowpic: Encrypted internet traffic classification is as easy as image recognition," in *IEEE Workshop on Network Intelligence: Machine Learning for Networking (NI 2019)*, 2019, pp. 680–687.

[2] 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.

[3] 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.

[4] 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.

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

[6] 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.

[7] 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 *IEEE International Conference on Big Data (Big Data)*, Dec 2017, pp. 1271–1276.

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

[9] 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.

[10] M. Lotfollahi, M. J. Siavoshani, R. S. H. Zade, and M. Saberian, "Deep packet: A novel approach for encrypted traffic classification using deep learning," *Soft Computing*, vol. 24, no. 3, pp. 1999–2012, 2020.

[11] 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.

[12] S. Zander, T. Nguyen, and G. Armitage, "Automated traffic classification and application identification using machine learning," in *The IEEE Conference on Local Computer Networks 30th Anniversary (LCN'05)l*, Nov 2005, pp. 250–257.

[13] 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.

[14] J. Muehlstein, Y. Zion, M. Bahumi, I. Kirshenboim, R. Dubin, A. Dvir, and O. Pele, "Analyzing HTTPS encrypted traffic to identify user's operating system, browser and application," in *2017 14th IEEE Annual Consumer Communications Networking Conference (CCNC)*, Jan. 2017, pp. 1–6.

[15] M. Conti, L. V. Mancini, R. Spolaor, and N. V. Verde, "Analyzing android encrypted network traffic to identify user actions," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 1, pp. 114–125, Jan 2016.

[16] S. E. Coull and K. P. Dyer, "Traffic analysis of encrypted messaging services: Apple imessage and beyond," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 5, pp. 5–11, Oct. 2014.

[17] R. Schuster, V. Shmatikov, and E. Tromer, "Beauty and the burst: Remote identification of encrypted video streams," in *26th USENIX Security Symposium*, Vancouver, BC, Canada, 2017, pp. 1357–1374.

[18] R. Dubin, A. Dvir, O. Pele, and O. Hadar, "I know what you saw last minute - encrypted http adaptive video streaming title classification," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 12, pp. 3039–3049, Dec 2017.

[19] M. Finsterbusch, C. Richter, E. Rocha, J. A. Muller, and K. Hanssgen, "A survey of payload-based traffic classification approaches," *IEEE Communications Surveys Tutorials*, vol. 16, no. 2, pp. 1135–1156, Second 2014.

[20] T. Bujlow, V. Carela-Español, and P. Barlet-Ros, "Independent comparison of popular DPI tools for traffic classification," *Computer Networks*, vol. 76, pp. 75 – 89, 2015.

[21] 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.

[22] W. Lu and L. Xue, "A heuristic-based co-clustering algorithm for the internet traffic classification," in *2014 28th International Conference on Advanced Information Networking and Applications Workshops*, May 2014, pp. 49–54.

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

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

[25] 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.

[26] 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.

[27] 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.

[28] 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.

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

[30] J. Zhang, F. Li, H. Wu, and F. Ye, "Autonomous model update scheme for deep learning based network traffic classifiers," in *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1–6.

[31] F. Pacheco, E. Exposito, and M. Gineste, "A framework to classify heterogeneous internet traffic with machine learning and deep learning techniques for satellite communications," *Computer Networks*, vol. 173, p. 107213, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128619313544

[32] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapè, "Mimetic: Mobile encrypted traffic classification using multimodal deep learning," *Computer Networks*, vol. 165, p. 106944, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128619304669

[33] A. S. Iliyasu and H. Deng, "Semi-supervised encrypted traffic classification with deep convolutional generative adversarial networks," *IEEE Access*, vol. 8, pp. 118–126, 2020.

[34] C. Hardegen, B. Pfülb, S. Rieger, and A. Gepperth, "Predicting network flow characteristics using deep learning and real-world network traffic," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2662–2676, 2020.

[35] 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.

[36] 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.

[37] C. Szegedy, A. Toshev, and D. Erhan, "Deep neural networks for object detection," in *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2013, pp. 2553–2561.

[38] 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.

[39] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.
[40] Y. T. Zhou, R. Chellappa, A. Vaid, and B. K. Jenkins, "Image restoration using a neural network," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 36, no. 7, pp. 1141–1151, Jul 1988.
[41] 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.
[42] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *the 27th International Conference on International Conference on Machine Learning*, ser. ICML'10, 2010, pp. 807–814.
[43] 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.
[44] 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.
[45] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.
[46] F. Chollet *et al.*, "Keras," https://github.com/fchollet/keras, 2015.
[47] M. A. et al., "Tensorflow," https://www.tensorflow.org/, 2015.
[48] 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.
[49] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.
[50] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

Tal Shapira (GS'19) was born in Rishon-Lezion, Israel, in 1991. He received the B.Sc. degree in physics and minor in mathematics from the Hebrew University of Jerusalem, Jerusalem, Israel, in 2012, and the M.Sc. degree in electrical engineering (magna cum laude) from Tel-Aviv University, Tel-Aviv, Israel in 2018.

From 2009 to 2012, he served as a cadet in the Talpiot elite Israeli Defense Forces program. From 2013 to 2015, he served in the Israeli Prime Minister Office as an RF and Antennas Engineer, from 2015 to 2018 as a Data Scientist and a Data Science Team Leader (Israeli Defense Award), and in the last two years of his military service as a head of a cybersecurity R&D group. After graduation, he worked as the Head of Deep Learning and Computer Vision Algorithms Group in an automotive startup called Guardian Optical-Technologies. In 2020, he co-founded RecoLabs Inc., where he serves as the Chief Scientist.

Shapira is a Ph.D. candidate in the School of Electrical Engineering at Tel-Aviv University since 2019. His research focuses on deep learning, computer networks, and cybersecurity.

Yuval Shavitt (S'88–M'97–SM'00) received the B.Sc. degree in computer engineering (cum laude), the M.Sc. degree in electrical engineering, and the D.Sc. degree from the Technion, Israel Institute of Technology, Haifa, in 1986, 1992, and 1996, respectively.

From 1986 to 1991, he served in the Israel Defense Forces first as a System Engineer and the last two years as a Software Engineering Team Leader. After graduation, he spent a year as a Postdoctoral Fellow at the Department of Computer Science at Johns Hopkins University, Baltimore, MD. Between 1997 and 2001, he was a Member of Technical Stuff at the Networking Research Laboratory at Bell Labs., Lucent Technologies, Holmdel, NJ. Starting from October 2000, he is a Faculty Member in the School of Electrical Engineering at Tel-Aviv University Tel-Aviv, Israel.

Prof. Shavitt served as a TPC member for many networking conferences, and on the executive committee of INFOCOM 2000, 2002, and 2003. He was a TCP co-chair for TMA 2011, keynote speaker at PAM 2012, an Editor of Computer Networks 2003-2004, and served as a Guest Editor for JSAC and JWWW. In 2014, he co-founded BGProtect LTD, a company that monitors the Internet for IP hijack attacks, where he serves as the CTO. His recent research focuses on Internet measurements, and deep learning solutions for various networking problems.