Computer Communications 186 (2022) 166–173

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Fast and lean encrypted Internet traffic classification

Sangita Roy [a,b], Tal Shapira [b,*], Yuval Shavitt [b]

[a] Thapar Institute of Engineering and Technology, India
[b] School of Electrical Engineering, Tel-Aviv University, Israel

ARTICLE INFO

Keywords:
Deep learning
Internet traffic classification
ODE

ABSTRACT

Identifying the type of a network flow or a specific application has many advantages but becomes harder in recent years due to the use of encryption, e.g., by VPN. As a result, there is a recent wave of solutions that harness deep learning for traffic classification. These solutions either require a rather long time (15–60 Seconds) of flow data or rely on handcrafted features for solutions that classify flows faster.

In this work, we suggest a novel approach for classification that extracts the most out of the two simple yet defining features of a flow: packet sizes and inter-arrival times. We employ a model that uses the inter-arrival times to parameterize the derivative of the flow hidden-state using a neural network (Neural ODE). We compare our results with a solution that uses the same data without the ODE solver and show the benefit of this approach.

Our results can classify flows based on 20 or 30 consecutive packets taken from anywhere in one direction of a flow. This reduces the amount of traffic between the sampling point and the analyzer and does not require matching between two directions of the flow. As a result, our solution can classify traffic with good accuracy within a few seconds, and we show how to combine it with a more accurate (and a slower) classifier to achieve (mostly) fast and accurate classifications.

1. Introduction

To improve the quality of service (QoS), network traffic classification plays an important role in emphasizing the emergence of several services provided by the network. The quality includes content, time, bandwidth, speed, and money. The services demand proper monitoring of network traffic. Besides QoS implementation, traffic classification focuses on resource management and pricing, malware and anomaly detection, criminology, and traffic engineering.

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. Up until a few years ago, most of the presented techniques for classifying encrypted traffic relied on extracting statistical features (also called feature extraction) from a traffic flow. This is followed by the process of feature selection to eliminate irrelevant features, and finally use shallow methods of supervised learning, such as decision trees, SVM (Support Vector Machine), and KNN (K-Nearest Neighbors) for the classification.

Over the past few years, advances in deep learning [1] have driven tremendous progress in many fields due to its auto-learning ability. The increasing availability of GPUs considerably helps in the calculation of complex matrix manipulation and mathematical calculations. As a result, many solutions for traffic classification are based on deep learning network models: RNN, CNN, and auto-encoder (see our related work section). In these approaches, data from the flow packets are collected over a certain period of time, and then classification is applied to the flow of packets. Flow-based deep learning reached impressive results for many traffic classification tasks recently. However, they suffer from several engineering problems. Some approaches are required to obtain both directions of a communication session, which are not always available, and require considerable efforts to store and match. Other requires obtaining many features and/or obtaining a few features over a long period of time, which requires sending these many data items from the tapping point to the analysis server. Finally, in many cases, we need fast classification that will allow us to adjust the flow treatment quickly.

This paper introduces ODENet combined with LSTM to obtain a fast classification of network traffic using only two features from one direction of the communication: packet size and inter-arrival time. A sufficient number of packets for the classification can be obtained in a few seconds or even below one second. As a result, the solution lends itself to easy engineering. LSTM is used to solve the vanishing gradient problems which arise in RNN. The ODENet accelerates the numerical precision and provides parameterized derivatives of the hidden state using a neural network.

Our contribution is a generic approach for Internet traffic classification that takes advantage of all time and size-related information

* Corresponding author.
E-mail addresses: sangita.roy@thapar.edu (S. Roy), talshapira1@mail.tau.ac.il (T. Shapira), shavitt@eng.tau.ac.il (Y. Shavitt).

https://doi.org/10.1016/j.comcom.2022.02.003
Received 31 October 2021; Received in revised form 4 February 2022; Accepted 5 February 2022
Available online 11 February 2022

available in a network flow instead of using information from manually extracted features. Moreover, our model can deal with a small number of packets from a unidirectional flow instead of the entire bidirectional session. We use the same architecture for all the experiments in the paper — we did not make any 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 [2–5], 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. We make our code publicly available.[1]

The rest of the paper continues as follows. After describing related work in Section 2, we describe the dataset in Section 3. Section 4 describes the ODE method and our deep learning architecture. Section 5 presents our experiments and their results. Finally, the last section concludes the paper.

2. Related work

Internet traffic classification problems can be divided into three main categories: 1. Categorizing Internet traffic [6] into classes such as Video, VoIP, File Transfer, etc., 2. identifying Internet applications [7–9] such as YouTube, Facebook, Skype, etc., and 3. identifying user actions [10] in a specific application like sending a text message on apple iMessage [11] or watching a specific video on YouTube [12,13].

Different classical approaches were used in the past for Internet traffic classification: 1. Payload based traffic classification methods [14, 15], also called deep packet inspection (DPI). These methods are problematic because they invade privacy, are computationally expensive, and are incapable of dealing with most of today's traffic due to the use of encryption and 2. Port-based methods — based on TCP/UDP packet header fields values, mainly the port number. These methods are fast and simple, but their efficiency declined with the increased use of dynamic and default ports.

Therefore, due to the mentioned problems, many statistical and machine learning-based methods [16] have been studied. These methods usually are done by manually extracting size and time-related features and applying complex patterns or supervised learning algorithms as classifiers. Besides these, some works present hybrid approaches [17], 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.

Many works focused on the process of feature generations [18–20]. These methods usually created a long list of handcrafted features extracted from bidirectional flows, such as RTT (round-trip delay time) statistics, packet size statistics, inter-arrival time statistics, frequencies, and so on. Then some [20] applied feature selection techniques to obtain an optimal feature set. Based on the obtained features, they applied machine-learning classifiers such as Naive Bayes Kernel estimator, SVM, decision trees, etc. More recently, Gil et al. [21] used statistical time-related features such as flow bytes per second, inter-arrival time, etc. They achieved accuracy levels above 80% by generating bidirectional flows of Non-VPN and VPN traffic and applying C4.5 and KNN as classifiers. Zhang et al. [22,23] used 20 simple unidirectional flow-based features and applied a bag of words (BoF) technique to model correlation information in traffic flows of the same application.

Payload-based methods were revisited in recent years with new approaches. Wang et al. [2,3] converted each packet payload to a normalized byte sequence and used it as input for a neural network. Moreover, using 1-D Convolution Neural Networks, the classification results over the ISCX VPN–nonVPN traffic dataset [21] have been improved relative to previous works. Lotfollahi et al. [24] applied almost the same method using CNNs and auto-encoders over the same dataset and achieved good performance. Aceto et al. [25,26] introduced a multimodal deep learning framework for mobile encrypted traffic classification. Their framework uses both payload data and protocols fields for the classification.

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 the internet categories to classify unknown applications. Moreover, these methods are problematic when encryption techniques are used, e.g., with VPN or Tor. Wang et al. [3] success in VPN classification is due to the usage of training data and test data based on the same encryption method and encryption keys.

Qin et al. [27] were among the first to understand the need to avoid manually extracted features. They used the Renyi cross-entropy to identify the similarity between a payload size distribution (PSD) of a given flow with the one generated for a specific application. Ertam and Avci [28] used a genetic algorithm (GA) for feature selection, then they applied a wavelet kernel-based extreme learning machines (ELM) over a dataset that contained 7 classes of regular traffic (non-VPN), and achieved accuracy over 95%.

Some recent works involved the use of deep learning. Lopez-Martin et al. [5] 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 port information and 84% without port information, which emphasized the weakness of their method. Chen et al. [29] converted flow data to a picture of the flow parameter auto-convolution and fed it to a neural network. However, they also used side information such as the target IP and had not described their method sufficiently to enable comparison. Zhang et al. [30] proposed an autonomous model update scheme to be able to handle new applications. Pacheco et al. [31] emulated satellite communications and presented a framework to classify heterogeneous Internet traffic with deep learning techniques for this type of communication. Iliyasu and Deng [32] addressed the challenges associated with establishing ground truth labels of large encrypted traffic datasets and introduced a semi-supervised approach using DCGAN. Their approach achieved good accuracy with a very small number of labeled samples.

A recent work by Shapira and Shavitt [33] suggested FlowPic, a transformation of the packet size and inter-arrival times into a picture, and used standard image classification deep learning to classify the Flow pictures into categories with high accuracy. They also classified in the same way applications.

3. 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) [21] as well as our a small packet capture (TAU) as used in [33].

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.

Since the UNB datasets do not contain enough flows for chats, we use the TAU's captured traffic of Whatsapp web chat, Facebook chat, and Google Hangout chat. We use a dataset only from the five categories that contain enough samples: Video, VoIP, Browsing, Chat, and File Transfer. For these categories, we have used two encryption techniques: non-VPN (Regular) and VPN (for all classes except Browsing). Notice that our categories are the same as in [33], but differ slightly from those suggested in [21,34]. Thus, we chose [33] to compare our results with. All the applications that were captured to create the dataset, for each traffic category and encryption technique, are shown in Table 1.

[1] https://github.com/talshapira/ODE-Flow.

Table 1
List of captured protocols and applications for each traffic category and encryption technique.

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

Table 2
The number of session blocks of regular traffic category.

Session length	Classes				
	VoIP	Video	File Transfer	Chat	Browsing
10	7170	7481	7049	4889	7401
20	7334	6955	7178	2303	7347
30	6883	6770	6892	1379	6752

Table 3
The number of session blocks of VPN traffic category.

Session length	Classes			
	VoIP	Video	File Transfer	Chat
10	7270	7276	7235	5147
20	7168	7488	6989	2518
30	7157	7078	7004	1703

3.1. Dataset preparations

The dataset is made of captured 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 kinds of mistakes in our dataset, we follow the process done in [33] and keep only the flows that belong to the correct category. We split each pcap file into unidirectional flows, where each flow is defined by a 5-tuple {source IP, source port, destination IP, destination port, protocol}.

3.2. Data processing

To increase the number of training examples, we divide each unidirectional flow by the number of packets. For every class vs. all (VoIP, Video, File_Transfer, Chat, and Browsing), session blocks are of 10, 20, and 30 packets. Table 2 shows the number of session blocks of five classes of regular traffic, and Table 3 shows the number of session blocks of four classes of VPN traffic. For each packet, we generate two features to obtain the following time-series features: relative time and packet length. We normalized the data by assuming a maximum value of 1500 Bytes (which is the Ethernet MTU value) for packet length.

4. Methods

4.1. Preliminaries

Recurrent Neural Networks or **RNNs** [35] are a class of Artificial Neural Networks (ANNs) for processing sequential data. Unlike ANNs, RNNs perform the same task for every element of a sequence, with the output being dependent on the previous computations and is produced using the same update rule applied to the previous outputs.

The Long Short Term-Memory neural networks, or **LSTMs**, that was first introduced by Hochreiter and Schmidhuber [36], are a specific kind of RNNs that are capable of learning long-term dependencies. These networks are composed of units called memory cells that have an internal recurrence (a self-loop), in addition to the outer recurrence of the network.

RNN and LSTM are very popular deep learning models to analyze time-series data, but still, their performance is not very promising. A Residual network (ResNet) specifies a discrete sequence of finite transformation at hidden layers, but in contrast, Ordinary Differential Equation Network (ODENet) defines a continuous transformation of the hidden state. The neural ODE is first proposed by Chen et al. [37] and it focuses on the connection between neural network and differential equation.

4.2. Background

4.2.1. Ordinary Differential Equation Network (ODENet)

Ordinary Differential Equation Networks, also known as **ODENets** that were first introduced by Chen et al. have a major role in the field of deep learning time series data analysis. ODENet is comparatively a new family in deep learning networks. The numerical concept of ordinary differential equations is embedded into the deep learning methods to build a framework to handle massive time-series data reliably and efficiently.

The transformation into the hidden state of Residual networks can be expressed as:

$$h_{t+1} = h_t + f(h_t, \theta_t) \tag{1}$$

where h_t denotes the tth layer's hidden value, $t \in \{0, \dots, T\}$ and $h_t \in \mathbb{R}^D$.

In ODE, the above expression can be written as a Euler discrimination of differential equation considering more layers are added and smaller steps are considered:

$$\frac{dh_t}{dt} = f(h_t, t, \theta) \tag{2}$$

Neural networks with an ODE has several advantages:

1. ODE does not save all intermediate states of the hidden layers; hence it is memory efficient.
2. The Euler method, which is used in RNN or ResNet, achieves only first-order accuracy, but on the other hand, ODE can achieve higher accuracy.
3. ODE requires a smaller number of parameters since ODE uses a continuous function of time.
4. ODE performs well for accuracy and stability in small datasets as well as large ones.
5. Neural ODE is useful for irregular time series data.

4.3. The architecture

Our goal is to design an architecture that can classify Internet flows using a small number of packets. We wish to use only packet sizes and inter-arrival times. Thus, we introduce an LSTM classifier that is preceded by an ODENet.

LSTM networks are composed of units called memory cells that have an internal recurrence (a self-loop), in addition to the outer recurrence of the network. Each cell contains three 'gates'; 1. an input gate to control the flow of inputs into the memory cell, 2. an output gate to control the output flow of cell activations into the rest of the network, and 3. a forget gate, which controls the self-loop weight and set it to a value between 0 and 1 via a sigmoid unit.

The LSTM model is fed by the output from the ODENet as depicted in Fig. 1. We used a classical LSTM with 32 hidden dimensions. The LSTM layer is following Eq. (1). The n outputs (n is the number of packets taken from a flow) of the ODENet are fed as features to

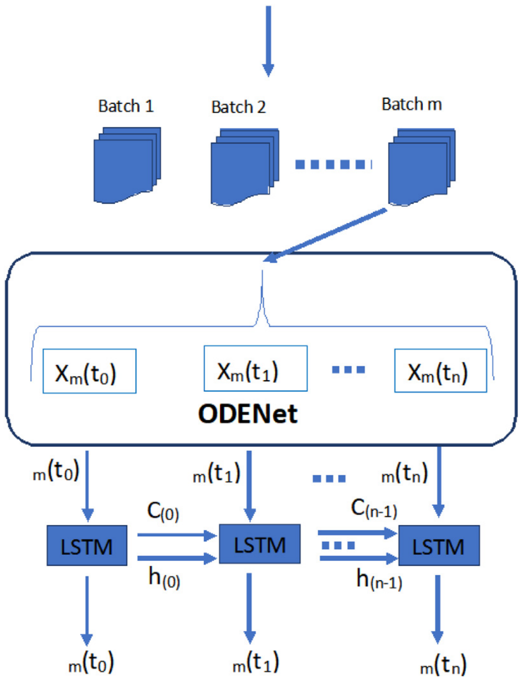


Fig. 1. ODE-LSTM hybrid layer.

Table 4
The architecture of our LSTM-ODE network.

Layer	Size
LSTMx2:	Input: n, 1 Output: n, 32
FC-1:	Output: 128
FC-2:	Output: 64
FC + Softmax:	Output: 5

the LSTM. The ODENet starts working with a single batch and its n successive data is labeled.

As depicted in Table 4, our LSTM-ODE architecture comprises four layers, not counting the input. Our input layer consists of n entities, which is the number of packets taken from a flow, followed by 2 LSTM layers with 32 hidden dimensions. Then we apply a sequence of 2 fully-connected (FC) layers of size 128 and 64, respectively. Finally, our output layer is the softmax layer, whose size depends on the number of classes: 5 for regular traffic classification and 4 for VPN traffic classification.

For comparison purposes, three different models have been used: 1-input classical LSTM, 2-input classical LSTM, and an LSTM with ODE. For a fair evaluation, we use the same architecture for the three models with the following changes:

1. For the 1-input architecture which we used for comparison, we simply replaced the input to the LSTM: instead of using the ODENet output, we use the packet size itself.
2. For the 2-input architecture, we doubled the input size to the LSTM model and used as features the packet size and inter-arrival times.

4.4. Training specification

The training is done by optimizing the *categorical cross entropy* [38] cost function, which 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 *SGD (Stochastic gradient descent)* optimizer. The loss function is defined as:

$$L(z(t_1)) = L\left(\int_{t_0}^{t_1} f(z_t, t, \theta)\, dt\right) = L(ODE(z(t_0), f, t_0, t_1, \theta)) \tag{3}$$

We build and run our networks using the PyTorch [39] library. To compare reliably between all sub-problems results, we run our network for 25 epochs (which took between 5 to 10 min for an epoch) of bath size 1 and save the result which achieve the best accuracy during the training process. In all experiments, our ODENet reaches convergence after running on 10 to 15 epochs.

5. 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 is challenging. Even the few papers that used the same datasets as we did [3,7,21,34] are not always directly comparable due to the 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.

5.1. Labeling datasets for different problems

After creating the pre-processed dataset, as mention before, we generate a balanced sub-dataset for each sub-problem. For class vs. all datasets, the specific class is equal to the number of samples in all other classes together, such that the ratio between the quantities of the other classes remains constant. We do it using a *random undersampling* method [40] in order to preserves the initial distribution of samples in each class.

5.1.1. 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. We construct class vs. all datasets for 2 encryption techniques: non-VPN or Regular and VPN (for all classes except Browsing).

5.1.2. Multiclass

We examine multiclass classification problem for *Traffic categorization*, which consists of an equal number of samples for all traffic categories that were mentioned before. We create multiclass datasets for 2 encryption techniques: non-VPN and VPN.

5.2. Evaluation criteria

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 dataset; we use 80% of the samples as a training set and 20% of the samples as a test set.

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** (Figs. 5 and 6). 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).

Table 5
One class vs. all regular traffic accuracy (%).

No. of	Method	Classes				
Packets		VoIP	Video	File_Transfer	Chat	Browsing
10	**ODE**	94.42	76.26	85.28	89.9	67.06
	1_input	78.18	71.94	83.44	70.20	77.26
	2_input	83.77	72.30	83.64	80.3	62.21
20	**ODE**	96.53	86.38	92.15	85.62	72.41
	1_input	83.78	87.77	87.09	64.05	82.6
	2_input	89.58	84.44	87.09	65.36	60.28
30	**ODE**	99.44	80.94	90.47	84.69	78.13
	1_input	86.30	79.15	64.40	76.53	77.94
	2_input	81.67	79.60	65.56	79.59	69.85
–	**FlowPic**	**99.6**	**99.9**	**98.8**	**96.2**	**90.6**

Table 6
One class vs. all VPN traffic accuracy (%).

No. of	Method	Classes			
Packets		VoIP	Video	File_Transfer	Chat
10	**ODE**	97.62	80.17	86.64	90.62
	1_input	77.58	82.52	88.18	78.54
	2_input	85.71	82.52	88.36	83.15
20	**ODE**	94.54	78.22	41.91	86.7
	1_input	88.32	99.17	40.17	63.76
	2_input	90.39	98.96	41.04	76.61
30	**ODE**	99.50	84.35	71.94	96.1
	1_input	89.11	84.01	76.02	68.83
	2_input	91.79	83.67	76.19	81.17
–	**FlowPic**	**99.9**	**99.9**	**99.9**	**99.2**

5.3. Results on class vs. All problems

In many cases, there is a need to distinguish a single traffic category from the rest. Tables 5 and 6 show a summary of the results of class vs. all classification problems by comparing different traffic categories with and without VPN, as described in Table 1. For each traffic category, our ODENet was trained over 2 training sets according to different encryption techniques. Each one of the trained networks was tested on 2 test sets, consisting of samples from the trained class with one of the above types of encryption techniques: Non-VPN (Regular) and VPN. Table 7 shows the average results of each one of the encryption techniques based on the number of packets.

To quantify the contribution of the ODENet, we compare the ODENet results in Tables 5 and 6 with an LSTM that uses the same architecture and the same data but without the ODE. We used the packet data in two ways. In the experiments whose results are labeled as '1-input', we use the packet sizes as input to the LSTM architecture that is fed in the ODENet experiment. In the experiments whose results are labeled as '2-input', we use the packet sizes and the inter-arrival times[2] as input to the LSTM architecture that is similar to the one fed in the ODE experiment but double its size to accommodate the additional parameter. There is a significant difference between the three experiments for almost all cases: ODE is usually a clear winner, and '2-input' is usually doing better than '1-Input'.

In General, the ODE results are very good with the correct classification of more than 4 out of every 5 flows, using only 10, 20, or 30 packets from one direction of the flow. However, when comparing these results with a solution like FlowPic [33] that reported average accuracy of **97.0%** for non-VPN (Reg) traffic and **99.7%** for VPN (not including browsing traffic, as mentioned before), they do not seem attractive. We note that the FlowPic system requires at least 15 Seconds to make an evaluation while the ODE solution requires less than a Second for VoIP, Video, and FTP flows and a median of 10 s or less for chat and browsing (for 20 packets), as presented in Fig. 4.

[2] Note that while in ODE we use the relative time, for the '2-input' we use the inter-arrival time for better results.

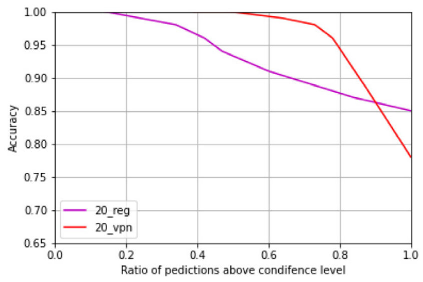


Fig. 2. Accuracy vs. classification ratio for video traffic identification using ODE.

Table 7
One class vs all average accuracy (%).

Method	Encryption	No. of Packets		
		10	20	30
ODE	**Non-VPN**	82.58	86.62	86.73
	VPN	88.76	75.34	87.97
1-input	**Non-VPN**	76.20	81.06	76.86
	VPN	81.71	72.86	79.49
2-input	**Non-VPN**	76.44	77.35	75.25
	VPN	84.94	76.75	83.21

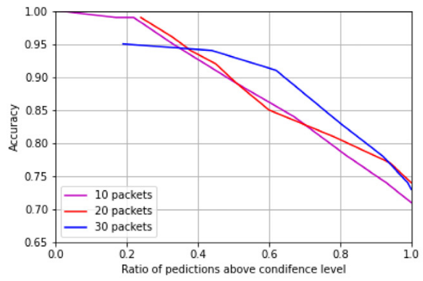
One way to improve the classification accuracy is to use the ODE results only if the classification confidence is above a threshold. Fig. 2 shows this trade-off for video traffic when we use 20 packets. Table 5 shows that the accuracy is 86.4% for regular traffic when we classify all the flows. Fig. 2 shows that if we are willing to classify 60% of the flows, our classification accuracy is over 91%. For VPN, the improvement is even better, if we are willing to set the threshold such that we classify 78.4% of the flow, our accuracy improves from 78.4% to 96.6%, and we can set it such that we get 100% accuracy for classifying almost half the flows. Note that for VPN traffic, we reach 100% accuracy for almost 40% of the data, while for regular traffic, this happens for less than 20%. We will revisit this point later when discussing multi-class categorization.

5.4. Results on traffic categorization problems

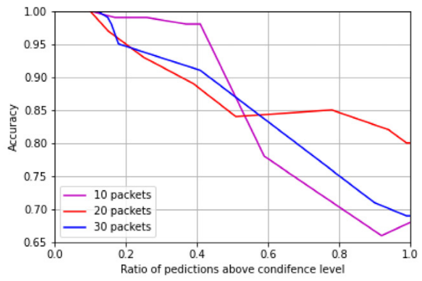
Fig. 5 shows the confusion matrix for traffic categorization for the case of 20 packets. The average accuracy for regular traffic is about 75% and for VPN traffic 80% (see Fig. 3. Note that for regular traffic with 20 and 30 packets classification cannot reach 100% accuracy, thus the lines do not start from the beginning of the abscissa). Clearly, VoIP and Video are easier to classify than the rest. The lack of browsing flows in the VPN dataset may explain the higher accuracy for this experiment than the one for regular traffic. The corresponding results for solutions that examine longer windows of time are better, for example, FlowPic [33] achieves accuracies of 85% and 98.4%, respectively. However, FlowPic requires at least 15 s of data, while here, we can obtain classification at a tenth of the time.

We would like to be able to enjoy both fast classification and high accuracy. To this end, we suggest the following algorithm, we first classify flows based on ODE but require a high threshold for classification and classify only sessions with a sufficient number of packets in a few seconds. If this threshold is reached and the flow has a sufficient number of packets, we have a fast and accurate classification. Otherwise, we wait for slower and more accurate solutions. For example, wait 15 s and classify based on FlowPic.

Fig. 3 shows the trade-off between the percentage of flows that are classified with ODE and the achieved accuracy. For the 20 packet example, for evaluation of regular traffic, we get 75% accuracy by

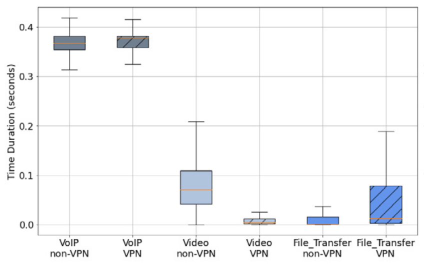


(a) Regular

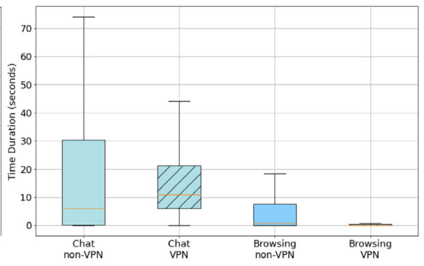


(b) VPN

Fig. 3. Accuracy vs. classification ratio for Multiclass traffic classification using ODE.

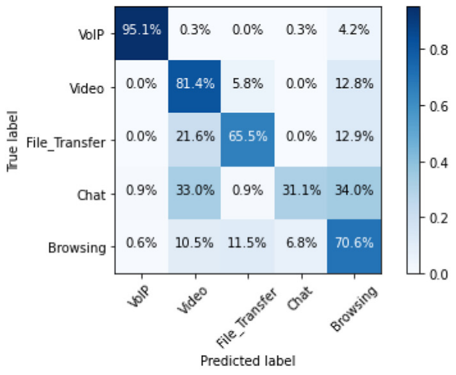


(a) VoIP, Video, File Transfer

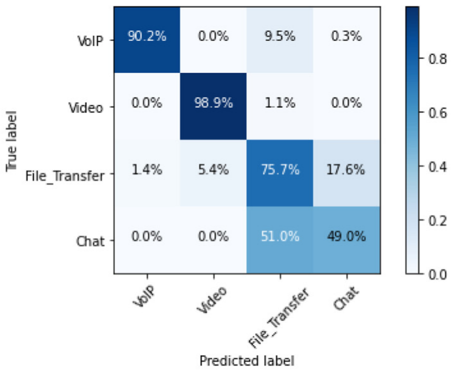


(b) Chat, browsing

Fig. 4. Time duration statistics for 20-packet flows.

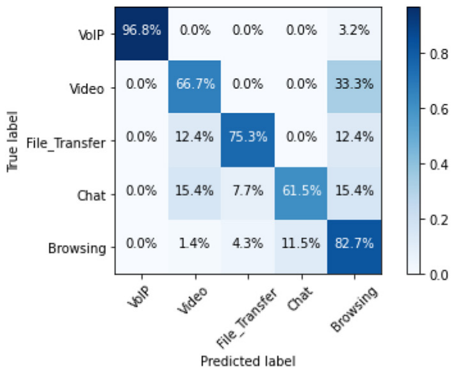


(a) 20 packets reg

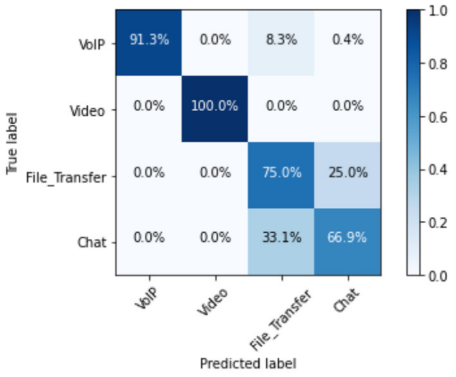


(b) 20 packets VPN

Fig. 5. Confusion matrix of Regular and VPN Multiclass.



(a) 20 packets reg



(b) 20 packets VPN

Fig. 6. Confusion matrix of Regular and VPN Multiclass using 0.8 as a minimal threshold.

classifying all packets. Using a threshold, we can increase the accuracy to 85% by classifying only 60% of the packets, and if we are willing to classify only half the packets, we get a 90% accuracy.

Fig. 6 shows the confusion matrix when we set the threshold at 0.8. At this point, we classify over 51% of the packets for regular traffic and above 50% of the packets for VPN traffic. Comparing the confusion matrix in Figs. 5 and 6 there is a clear improvement in accuracy with the threshold.

6. Conclusion

In this paper, we introduce a novel approach for encrypted internet traffic classification, which is based only on time and size-related information. The main advantage of our approach is the ability to get a classification based on only 10–30 packets, namely within a few seconds. It is also easy to deploy since it only requires two words for each packet, its size, and inter-arrival time. Finally, since the classification is only based on meta-data and does not require access to the packet payload, it is easier to deploy from the legal aspects, as well.

Our classification accuracy is very good for some applications, like identifying VoIP, but less for other applications. Thus, we suggest a hybrid approach where we classify fast a portion of the flows with high accuracy and wait for additional packets to arrive for the rest.

Our results may also be used for other types of problems where events are not evenly spaced in time, for example, in the study of cascading failures in the interdependent networks [41].

CRediT authorship contribution statement

Sangita Roy: Conception and design of study, Acquisition of data, Analysis and/or interpretation of data, Writing – original draft, Writing – review & editing. **Tal Shapira:** Conception and design of study, Acquisition of data, Analysis and/or interpretation of data, Writing – original draft, Writing – review & editing. **Yuval Shavitt:** Conception and design of study, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

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, Israel. All authors approved the version of the manuscript to be published.

References

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

[2] Z. Wang, The applications of deep learning on traffic identification, in: BlackHat USA, 2015.

[3] W. Wang, M. Zhu, J. Wang, X. Zeng, 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), 2017, pp. 43–48.

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

[5] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, J. Lloret, Network traffic classifier with convolutional and recurrent neural networks for internet of things, IEEE Access 5 (2017) 18042–18050, http://dx.doi.org/10.1109/ACCESS.2017.2747560.

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

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

[8] J. Muehlstein, Y. Zion, M. Bahumi, I. Kirshenboim, R. Dubin, A. Dvir, 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), 2017, pp. 1–6.

[9] S. Rezaei, X. Liu, Multitask learning for network traffic classification, in: 2020 29th International Conference on Computer Communications and Networks (ICCCN), 2020, pp. 1–9, http://dx.doi.org/10.1109/ICCCN49398.2020.9209652.

[10] M. Conti, L.V. Mancini, R. Spolaor, N.V. Verde, Analyzing android encrypted network traffic to identify user actions, IEEE Trans. Inf. Forensics Secur. 11 (1) (2016) 114–125.

[11] S.E. Coull, K.P. Dyer, Traffic analysis of encrypted messaging services: Apple imessage and beyond, SIGCOMM Comput. Commun. Rev. 44 (5) (2014) 5–11.

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

[13] R. Dubin, A. Dvir, O. Pele, O. Hadar, I know what you saw last minute - Encrypted HTTP adaptive video streaming title classification, IEEE Trans. Inf. Forensics Secur. 12 (12) (2017) 3039–3049.

[14] M. Finsterbusch, C. Richter, E. Rocha, J.A. Muller, K. Hanssgen, A survey of payload-based traffic classification approaches, IEEE Commun. Surv. Tutor. 16 (2) (2014) 1135–1156.

[15] T. Bujlow, V. Carela-Español, P. Barlet-Ros, Independent comparison of popular DPI tools for traffic classification, Comput. Netw. 76 (2015) 75–89.

[16] T.T.T. Nguyen, G. Armitage, A survey of techniques for internet traffic classification using machine learning, IEEE Commun. Surv. Tutor. 10 (4) (2008) 56–76.

[17] W. Lu, 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, 2014, pp. 49–54.

[18] A. Moore, D. Zuev, M. Crogan, Discriminators for Use in Flow-Based Classification, Tech. rep., Queen Mary Uni. of London, 2005.

[19] A.W. Moore, D. Zuev, Internet traffic classification using Bayesian analysis techniques, in: ACM SIGMETRICS, 2005, pp. 50–60.

[20] A. Fahad, Z. Tari, I. Khalil, I. Habib, H. Alnuweiri, Toward an efficient and scalable feature selection approach for internet traffic classification, Comput. Netw. 57 (9) (2013) 2040–2057.

[21] G. Draper-Gil, A.H. Lashkari, M.S.I. Mamun, A.A. Ghorbani, Characterization of encrypted and VPN traffic using time-related features, in: The 2nd International Conference on Information Systems Security and Privacy - Volume 1: ICISSP, 2016, pp. 407–414.

[22] J. Zhang, Y. Xiang, Y. Wang, W. Zhou, Y. Xiang, Y. Guan, Network traffic classification using correlation information, IEEE Trans. Parallel Distrib. Syst. 24 (1) (2013) 104–117.

[23] J. Zhang, X. Chen, Y. Xiang, W. Zhou, J. Wu, Robust network traffic classification, IEEE/ACM Trans. Netw. 23 (4) (2015) 1257–1270.

[24] M. Lotfollahi, M.J. Siavoshani, R.S.H. Zade, M. Saberian, Deep packet: A novel approach for encrypted traffic classification using deep learning, Soft Comput. 24 (3) (2020) 1999–2012.

[25] G. Aceto, D. Ciuonzo, A. Montieri, A. Pescapè, MIMETIC: Mobile encrypted traffic classification using multimodal deep learning, Comput. Netw. 165 (2019) 106944, http://dx.doi.org/10.1016/j.comnet.2019.106944.

[26] G. Aceto, D. Ciuonzo, A. Montieri, A. Pescapé, Toward effective mobile encrypted traffic classification through deep learning, Neurocomputing 409 (2020) 306–315, http://dx.doi.org/10.1016/j.neucom.2020.05.036.

[27] T. Qin, L. Wang, Z. Liu, X. Guan, Robust application identification methods for P2P and VoIP traffic classification in backbone networks, Knowl.-Based Syst. 82 (2015) 152–162.

[28] F. Ertam, E. Avci, A new approach for internet traffic classification: GA-WK-ELM, Measurement 95 (2017) 135–142.

[29] Z. Chen, K. He, J. Li, 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), 2017, pp. 1271–1276.

[30] J. Zhang, F. Li, H. Wu, 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, M. Gineste, A framework to classify heterogeneous internet traffic with machine learning and deep learning techniques for satellite communications, Comput. Netw. 173 (2020) 107213, http://dx.doi.org/10.1016/j.comnet.2020.107213.

[32] A.S. Iliyasu, H. Deng, Semi-supervised encrypted traffic classification with deep convolutional generative adversarial networks, IEEE Access 8 (2020) 118–126, http://dx.doi.org/10.1109/ACCESS.2019.2962106.

[33] T. Shapira, Y. Shavitt, FlowPic: A generic representation for encrypted traffic classification and applications identification, IEEE Trans. Netw. Serv. Manage. 18 (2021) 1218–1232.

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

[35] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagating errors, Nature 323 (6088) (1986) 533.

[36] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.

[37] R.T.Q. Chen, Y. Rubanova, J. Bettencourt, D.K. Duvenaud, Neural ordinary differential equations, in: 32nd Conference on Neural Information Processing Systems, 2018.

[38] D. Campbell, R.A. Dunne, 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.

[39] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, PyTorch: An imperative style, high-performance deep learning library, in: Advances in Neural Information Processing Systems 32, Curran Associates, Inc., 2019, pp. 8024–8035.

[40] S. Kotsiantis, D. Kanellopoulos, P. Pintelas, et al., Handling imbalanced datasets: A review, GESTS Int. Trans. Comput. Sci. Eng. 30 (1) (2006) 25–36.

[41] S. Hong, J. Zhu, L.A. Braunstein, T. Zhao, Q. You, Cascading failure and recovery of spatially interdependent networks, J. Stat. Mech. Theory Exp. 2017 (10) (2017) 103208, http://dx.doi.org/10.1088/1742-5468/aa8c36.