A Few Shots Traffic Classification with mini-FlowPic Augmentations

Eyal Horowicz
eyalhorowicz@mail.tau.ac.il
Tel Aviv University
Israel

Tal Shapira*
talshapirala@gmail.com
Reichman University
Israel

Yuval Shavitt
shavitt@eng.tau.ac.il
Tel Aviv University
Israel

ABSTRACT

Internet traffic classification has been intensively studied over the past decade due to its importance for traffic engineering and cyber security. One of the best solutions to several traffic classification problems is the FlowPic approach, where histograms of packet sizes in consecutive time slices are transformed into a picture that is fed into a Convolution Neural Network (CNN) model for classification.

However, CNNs (and the FlowPic approach included) require a relatively large labeled flow dataset, which is not always easy to obtain. In this paper, we show that we can overcome this obstacle by replacing the large labeled dataset with a few samples of each class and by using augmentations in order to inflate the number of training samples. We show that common picture augmentation techniques can help, but accuracy improves further when introducing augmentation techniques that mimic network behavior such as changes in the RTT.

Finally, we show that we can replace the large FlowPics suggested in the past with much smaller mini-FlowPics and achieve two advantages: improved model performance and easier engineering. Interestingly, this even improves accuracy in some cases.

1 INTRODUCTION

Internet traffic classification has been extensively studied in the past decade [1, 4, 21] as it is used for QoS implementations, traffic engineering, law enforcement, and even malware detection. However, due to the growing usage 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 either shallow methods of supervised learning or deep learning models for the classification.

One of the most promising approaches to traffic classification is to convert the flow features into an image or a pseudo-image (namely, a 2D matrix) and then use techniques borrowed from image classification. Generally, these solutions can be divided into approaches that examine large portions of the packets, usually the packet headers [1, 8], but sometimes also the payload [7, 12, 22], and others that extract only packet size, timing and direction [3, 9, 11, 17–20, 23]. The latter is more attractive due to their low cost of implementation.

However, many of the previous approaches require sufficiently large labeled datasets to train the deep learning model. Such datasets are hard to obtain, in fact, the ISCX and QUIC datasets that are used in this paper are the only publicly available labeled application flow datasets, and they both have limitations. Thus, it is desirable to have a solution that overcomes this difficulty.

Following the difficulties in obtaining labeled flow datasets, several approaches were suggested. Rezaei and Liu [15] suggested transfer learning where they first use many unlabeled sessions in order to learn flow statistical features and then continue to train the same model with a small batch of 5-20 samples per class. Shapira and Shavitt [19] used overlapping time intervals (time shifts) to extract more samples from each session. Iliyasu and Deng [9] suggested using samples generated by DCGAN generators to increase the number of samples from each session. Rezaei and Liu [16] used 10-100 samples per class and to overcome the small number of samples they learned multiple tasks simultaneously.

A recently suggested approach to deal with labeling scarcity is to use *contrastive representation learning*, where unsupervised samples are embedded into a latent space, such that similar sample pairs stay close to each other while dissimilar ones are far apart [2]. This way samples can be separated

*Work partly done while at Tel Aviv University.

IMC '22, October 25–27, 2022, Nice, France

ACM ISBN 978-1-4503-9259-4/22/10…$15.00
https://doi.org/10.1145/3517745.3561436

with simple methods such as a linear classifier. This approach was not suggested for traffic classification, yet.

In this paper, we utilized contrastive unsupervised representation learning for traffic classification and extend it to the task of few-shot learning, where only a few labeled samples are used per class. We select the FlowPic approach as our flow image representation since it was shown to perform well [19, 20, 23], and it is easy to implement.

In order to perform unsupervised contrastive learn representation, we need good data augmentations that alter images such that the result is an image that may represent a flow in the class. Unfortunately, common augmentations, which perform well on natural images, can not be applied directly on FlowPic images, since most of them would generate a FlowPic that is significantly different from the original label class. For example, a dog image that is rotated by 90 degrees, is still a valid dog picture, but a FlowPic that is rotated by 90 degrees may not represent any legitimate flow.

In this paper, we suggest augmentations that are based on manipulations of the flow data before the FlowPic is built, e.g., changing the RTT. As a result, our augmentations perform consistently well, gaining up to 17% improvement in classification accuracy. Furthermore, we show how to reduce the implementation cost with mini FlowPics and show that they can achieve excellent classifying results.

The only few-shot solution that was suggested for traffic classification in the past [15] is not suitable for online classification. The solution requires to sample packets from the entire flow and thus is hard to implement in practice. FlowPic on the other hand achieves same accuracy, but it is very easy to implement, and mini FlowPic further reduces the cost of implementation.

2 METHODS

2.1 FlowPic Representation

FlowPic is a histogram-based image representation for a sample of an Internet traffic session. It is created by partitioning a time interval into equal-length time slots and generating histograms by counting the number of packets arriving in the time slots according to their size. This image can be seen as an array of payload size distributions (PSDs) [14].

The FlowPic is a 1500x1500 pixel picture. The X-axis represents the time slot, and each column is a histogram, where the pixel value at height Y is the number of packets of this size in this time slot. If more than max value (255) packets of a certain size arrive in a time slot, we set the pixel value to max value. Since the absolute majority of the packets are not larger than 1500 bytes, which is the Ethernet MTU value, we disregard all packets with sizes greater than 1500 (less than 5% of all packets) and limit our Y-axis to be between 1 to 1500. For simplicity, we set the 2D-histogram to be a square image. We store each such histogram in an image matrix and term it a *FlowPic*. These images are later classified by deep learning models.

2.2 Mini FlowPics

As described above, the original FlowPic images[20] are 1500x1500 pixels. The justification for 1500 in the Y-axis is the ability to count each packet size between 1 to MTU value, and for the X-axis is the creation of a simple square image. Furthermore, [20] tested blocks of 120, 60, 30, and 15 seconds and reported that the differences in the average accuracy were up to 1.25%. In this paper, for all experiments, we use 15 seconds as the block size.

Mini FlowPics use the same construction idea as FlowPic but uses significantly smaller images. The Y-axis is reduced by generating histograms that count the number of packets in ranges of lengths, and X-axis is reduced by partitioning the time interval to a smaller number of time slots.

For example, if we use 30x30 image each pixel would count $\frac{1500}{30} = 50$ packets sizes which were arrived within $\frac{15}{30} = 0.5$ second time slot. In the original FlowPic paper the time slots were 40 or 160 milliseconds long.

The Mini FlowPics are cheaper to construct, train and predict on, which is a significant engineering advantage. The Mini FlowPics are also easier to be captured by a human eye for classification or sanity checks as can be seen in Figure 2 because the pixels are darker due to the aggregation (the figures of the FlowPics in the original FlowPic paper were generated by transforming each non-zero pixel to black). In all the experiments reported in this paper we set the Mini FlowPic image size to be 32x32 pixels, decreasing image size (total pixels) by a factor of 2197. In some experiments, we additionally used FlowPics of 64x64, which decreases the size by a factor of 549.

2.3 Data Augmentation for FlowPic Images

Training a deep learning (DL) model requires a large dataset of labeled data, or else one may risk overfitting. To overcome this, data augmentations were suggested to increase the number of labeled samples without actually collecting new data. Data augmentation artificially creates new training data by applying transformations to input data while preserving the class labels.

In the field of Computer Vision, there are many known image transformations, which were reported to work well in classifying a variety of standard image datasets. Some of the most popular transformations are flipping, color modification, cropping, rotation, noise injection, and random erasure.

The effect of transformations on regular images is easily understood by simply observing the new image. For example, when a picture of a dog is rotated or when we add some noise to the picture - it is still easy to observe the dog in the transformed picture. However, when one considers image transformations of FlowPics, they may alter significant characteristics of the picture, and it is impossible to tell which transformation is useful for augmentation by simply observing the outcome. For example, consider the popular rotation transformation. When rotating a FlowPic significantly, one changes the packet size distribution in time, and obviously does not help the learning process.

Thus, we suggest here to design FlowPic augmentations that attempt to mimic changes in the network conditions, such as changes in RTT, time translations, and packet loss. We will show later that such augmentation performs better than the classic ones, which were designed for natural images.

2.4 Contrastive Representation Learning

When there are only a few labeled samples per class (*few shots*) using augmentations cannot generate a sufficiently large, unbiased dataset. However, with sufficiently many unlabeled samples, we suggest to use *contrastive representation learning*.

The goal of contrastive representation learning is to learn a low-dimensional representation for each sample in the dataset such that in the embedding space similar sample pairs stay close together, while dissimilar ones are far apart. This method can be unsupervised utilized, as a result, with a few labeled samples, a simple linear classifier can produce good separation.

In this paper we utilize the SimCLR framework. To understand how it works we first define some notations:

- $\mathcal{T}$ a family of valid augmentations, i.e., transformations, which preserve label class.
- $f(\cdot)$ a CNN based encoder, that extracts representation vectors from data examples $\mathbf{x}$.
- $\mathbf{h} = f(\mathbf{x})$ the representation vector of sample $\mathbf{x}$.

In each training step, we select a randomly sampled mini-batch of N samples from unlabeled training set and two separate transformations, t and t', from $\mathcal{T}$. Using transformations t and t', each of the N samples generates 2 augmented samples, totaling $2N$ for a batch. namely, $\tilde{\mathbf{x}}_i = t(\mathbf{x}), \quad \tilde{\mathbf{x}}_j = t'(\mathbf{x})$.

Since t and t' preserve the label class, each $\tilde{\mathbf{x}}_i$ and $\tilde{\mathbf{x}}_j$ are consider as one positive pair, and the other $2(N-1)$ augmented samples are treated as negative samples. The representation vector h of the augmented samples is produced by the CNN based encoder: $\mathbf{h}_i = f(\tilde{\mathbf{x}}_i), \quad \mathbf{h}_j = f(\tilde{\mathbf{x}}_j)$

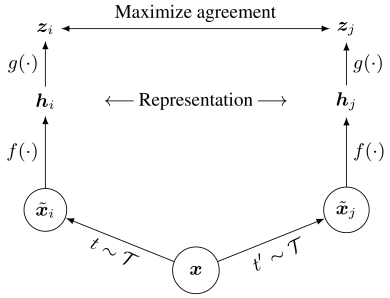


Figure 1: Illustration of SimCLR training process [2].

Extra projection layer of the representation h by a Linear or double Linear layer denoted by $g(\cdot)$ is applied resulting with the similarity vector z: $\mathbf{z}_i = g(\mathbf{h}_i), \quad \mathbf{z}_j = g(\mathbf{h}_j)$.

Finally the contrastive learning loss is defined using cosine similarity $\text{sim}(\mathbf{z}_i, \mathbf{z}_j) = \mathbf{z}_i \mathbf{z}_j / \|\mathbf{z}_i\| \|\mathbf{z}_j\|$

$$\mathcal{L}^{(i,j)}_{\text{SimCLR}} = -\log \frac{\exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbf{1}_{[k \neq i]} \exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_k)/\tau)}$$

where $\mathbf{1}_{[k \neq i]}$ is an indicator function: 1 if $k \neq i$ and 0 otherwise, τ denotes a temperature parameter. The final loss is computed across all positive pairs, both (i, j) and (j, i), in a batch and the networks $f(\cdot)$, $g(\cdot)$ weights are updated. Finally network $f(\cdot)$ taken as the representation extractor function, and $g(\cdot)$ is thrown away.

3 EXPERIMENTS AND RESULTS

3.1 Datasets

3.1.1 UC-Davis, QUIC Google Services. The QUIC dataset was captured at University of California at Davis lab [15], and consists of 5 Google services: Google Docs, Google Drive, Google Music, Youtube, and Google Search.

The dataset has already been pre-processed, all non-QUIC sessions and those that have fewer than 100 packets were removed. Each session is represented as a time series of the packet's arrival time, length, and direction. When we process this dataset we create a single FlowPic per session (composed of sampling only the first 15 seconds of the session).

The dataset includes thousands of flows that were triggered and labeled by automation scripts, and at least 15 flows per label class triggered and labeled by real human interaction.

For all experiments, for training set we use only 100 "triggered by script" flows per class, and for test set we follow the experiments by [16] randomly choosing 30 flows for each class for a "triggered by script" test set and 15 flows per class for "triggered by human" test set.

3.1.2 ISCX, VoIP, and Video Application Identification. We use a combination of the datasets from the Uni. of New Brunswick (UNB): "ISCX VPN-nonVPN traffic dataset" (ISCX-VPN) [5] and "ISCX Tor-nonTor dataset" (ISCX-Tor) [10].

These datasets consist of packet capture (pcap) files labeled by several encryption techniques (VPN, Tor, Regular), traffic types (VoIP, Video, etc.), and applications (Whatsapp, Facebook, etc.).

After filtering the ISCX dataset to include only sessions with at least 100 packets, most applications don't have more than a few sessions (Unlike the situation in the QUIC dataset). Luckily, the VoIP and video applications were recorded for several minutes in each session, thus, it is possible to extract many non-overlapping 15 second FlowPics out of each session.

Following [20], we create VoIP and Video Application Identification dataset, consists of 10 classes representing usage of VoIP application (Facebook, Hangouts, Skype, Buster) and video applications (Facebook, Hangouts, Netflix, Skype, Vimeo, Youtube) over non-VPN encryption technique.

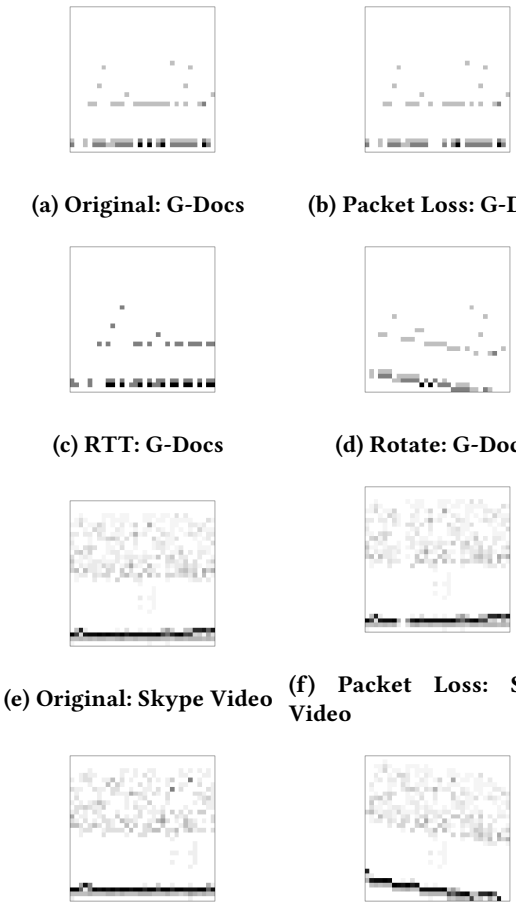
First, we split the sessions to test and train groups, such that they do not overlap. Then we keep the same setting of the QUIC dataset and sample FlowPics from the two groups, so the test set contains 30 FlowPic images per class and the train set has 100 FlowPics per class for the training set.

3.2 Supervised Training with FlowPic Augmentations

In this section, we show the strength of designing FlowPic augmentations by supervised experiments. First, we describe the different augmentations and the intuition which led us to build or use them.

We first describe various common **Computer Vision Augmentations** and analyze their expected effect on FlowPics and then continue by suggesting new augmentations based on network effects. Some of the results of the augmentation are illustrated on two FlowPics in Figure 2.

(1) **Rotate**: Image rotation is a popular augmentation since obviously, it preserves image content. However, when applied to FlowPics rotation changes the packet size distribution and thus seems inappropriate (see Figure 2(d) and (h)), unless maybe when only small angles are used. Thus, we tested this augmentation with angle rotation uniformly distributed in the range $[-10, 10]$ degrees. Another subtle issue with rotation is that areas without pixels in images rotation augmentation are usually filled with black, while in FlowPic the more appropriate filler color should be white.
(2) **Horizontal Flip**: This augmentation is acceptable for periodic signals, which is the case for many applications in FlowPic, such as Video and VoIP where the sessions are long and the image does not change significantly along the horizontal axis.
(3) **Color Jitter**: When used with small value Noise is equivalent to sending a slightly different number of packets at some sizes and we expect the model to disregard these small changes. Large noise values change the size distributions, and thus may have an adverse effect. The parameters we chose were brightness = 0.8, contrast = 0.8, saturation = 0.8, and hue = 0.2.
(4) **Crop and Resize**: This image augmentation will alter a FlowPic to a level that will not be considered as a variant of the original behavior. For example, zooming in is equivalent to generating significantly more traffic, since a single pixel that represent n packets of size y in time slot t may become 9 pixels representing $9n$ packets of sizes between $y-1$ and $y+1$ with three consecutive time slots.
(5) **Vertical Flip**: This transformation transforms large packets into small packets and wise versa, which does not maintain the same flow class characteristics.



(a) Original: G-Docs (b) Packet Loss: G-Docs (c) RTT: G-Docs (d) Rotate: G-Docs (e) Original: Skype Video (f) Packet Loss: Skype Video (g) RTT: Skype Video (h) Rotate: Skype Video

Figure 2: Examples of data augmentation applied on Mini FlowPics. We Show here Packet Loss of 1 second, Change RTT by a factor of 1.5, and Rotate by 10°

(6) **Translation** This transformation also changes the sizes of all packets and alters the application class.
(7) **Cutout** This transformation deletes all the packets of some size range at a specific time interval. The result may not be similar to the application class behavior.

Since applying the last 4 augmentations will results in FlowPics that are significantly different from the characteristics of the original class, we did not implement them. Instead, we suggest the following **Networking augmentations**:

(1) **Change RTT**: Round-trip time (RTT) in the Internet changes between a few milliseconds to a small fraction of a second. Due to TCP congestion control and application flow control, changes in the RTT translate almost linearly to the distribution of packets along the time axis.
We simulate a change in the RTT by multiply the arrival time of each packets by a factor α, and rebuild the FlowPic. The factor α is uniformly selected from $[\alpha_{min}, \alpha_{max}]$, namely $RTT_{new} = \alpha \cdot RTT_{old}$. we chose $\alpha_{min} = 0.5$ and $\alpha_{max} = 1.5$. Figures 2(c) and 2(g) depict the augmentation result.
(2) **Time Shift**: We simulate a Time Shift by adding a constant $b \in [b_{min}, b_{max}]$ to the arrival time of each packet and rebuild FlowPic. The constant b is uniformly sampled from $U[b_{min}, b_{max}]$, so $t_{new} = t_{old} + b$. we chose $b_{min} = -1$ and $b_{max} = 1seconds$.
(3) **Packet Loss** There are multiple ways to simulate packet loss that may take into account complex behavior such as the TCP congestion control. We simulate a simple loss process where we delete all packets in the time interval $[t - \Delta t, t + \Delta t]$. t is randomly selected in the session interval, and $\Delta t = 0.1$seconds. Note that this augmentation is a special case of a rectangle cutoff with maximal height. Figures 2(b) and 2(f) depict the augmentation result.

We first perform a supervised experiment that highlights the best augmentations for the learning process. The parameter selection is based on quite extensive parameter search that shows little sensitivity to parameters that are in the area we selected, but drop of accuracy when extreme values are used.

For this end, we supervised train our CNN architectures as described in Appendix A.1 on the datasets described in 3.1. For all experiments, we apply each of the augmentations 10 times on the 100 samples per class training set, which increase the training set to 1000 images per class.

We also train without any augmentation as baseline experiments and term it "no aug". For all experiments we allocated 20% of the images for validation, and early stopped the training when the validation loss stopped improving. Comparing the loss of the validation and the training makes sure there is no over-fitting. The result are tabulated in Tables 1–3.

	Mini FlowPic 32	Mini FlowPic 64	Full FlowPic
No aug	98.67	99.1	96.22
Rotate	98.6	98.87	94.89
Horizontal-flip	98.93	99.27	97.33
Color Jitter	96.73	96.4	94.0
Packets Loss	98.73	99.6	96.22
Time Shift	99.13	99.53	97.56
change RTT	**99.4**	**100.0**	**98.44**

Table 1: QUIC triggered by script - Accuracy of supervised training with different augmentations

	Mini FlowPic 32	Mini FlowPic 64	Full FlowPic
No aug	92.4	85.6	73.3
Rotate	93.73	87.07	77.3
Horizontal-flip	94.67	79.33	87.9
Color Jitter	82.93	74.93	68.0
Packets Loss	90.93	85.6	84.0
Time Shift	92.8	87.33	77.3
change RTT	**96.4**	**88.6**	**90.7**

Table 2: QUIC triggered by human - Accuracy of supervised training with different augmentations

	Mini FlowPic 32	Mini FlowPic 64	Full FlowPic
No aug	89.73	92.33	93.67
Rotate	86.3	87.135	93.0
Horizontal-flip	88.4	92.17	93.33
Color Jitter	86.9	86.6	91.3
Packet Loss	90.4	91.83	92.6
Time Shift	88.9	92.1	**95.33**
Change RTT	**91.0**	**93.17**	**95.33**

Table 3: Video and VoIP applications - Accuracy of supervised training with different augmentations

As Tables 1, 2, and 3 show, using the 100 labeled FlowPics per class is already a sufficient training dataset for achieving good results on traffic classifying tasks in the three datasets. Interestingly, we can see that for the "no aug" baseline the Mini FlowPics achieve better results on the QUIC datasets, but not on the Video and VoIP applications.

In all the nine experiments changing the RTT was the best performing augmentation. The improvement varies from 1% for the QUIC script dataset (where the "no aug" accuracy was already 98.7%) up to 17.4% improvement for the most challenging dataset, the QUIC human. While, the winning size architecture changes between datasets, choosing a mini FlowPic has both engineering advantages and is never too far from optimum.

3.3 Learning a Representation for a Few Shots classification

In this section, we utilize the FlowPic Augmentations for Contrastive Representation Learning within SimCLR. Following the results of the previous experiment, we selected to use 'Change RTT' by $\alpha \sim U[0.5, 1.5]$ together with Time Shift by $b \sim U[-1, 1]$. In each training step, a double batch of 32 *unlabeled* images (taken from the pool of 100 unlabelled samples per class) is loaded after applying the two augmentations above.

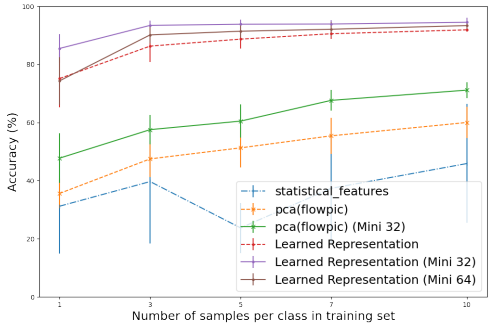


Figure 3: QUIC triggered by script - Accuracy as a function of the training set size

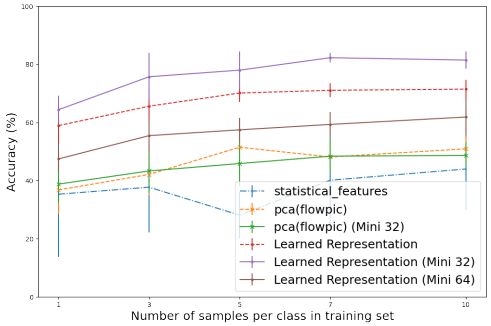


Figure 4: QUIC triggered by human - Accuracy as a function of the training set size

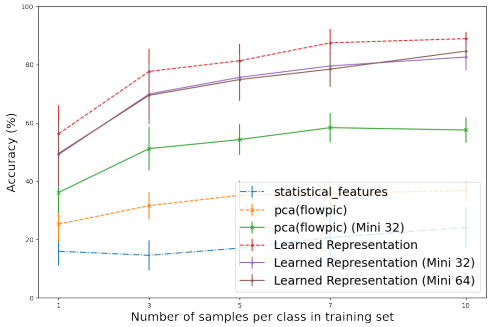


Figure 5: Video and VoIP applications - Accuracy as a function of the training set size

For the experiments described here, we use the same datasets from the last section, but this time the training process of the base encoder CNN network $f(\cdot)$ is fully unsupervised (see section 2 for details and Appendix A.2 for training specifications) yielding a 120 dimensional vector $h = f(FlowPic)$ for any FlowPic or mini FlowPic.

The evaluation of the learned representations is based on linear evaluation protocol, where a linear classifier is trained to classify the representation vectors.

Up until now, we performed unsupervised learning using the 100 samples per class. Next, we freeze the weights of $f(\cdot)$ and train the linear classifier (see Appendix A.3 for training specifications) using only a few labeled samples per class. We vary the number of samples per class, η, to evaluate how many such samples are required.

To evaluate our results, we replace our data representation h, with two other representations. The first representation used the following 10 statistical flow-based features since these features have been widely used in previous works [3, 6, 13]: Mean, minimum, maximum, and standard deviation of packet sizes and inter-arrival times; Flow Bytes per second (BPS); Flow packets per second (PPS). While these features were shown to perform well when plenty of labeled samples are available, here we limited them to η samples per class.

The second representation is also based on FlowPics and mini FlowPics (32x32), but here we apply PCA directly on the FlowPic to reduce its dimension to 120.

Figures 3, 4 and 5 depict the accuracy of the three methods, for the three datasets. Each marker is the average accuracy of 10 experiments where the training samples are randomly selected, and the whiskers show the STD.

Clearly, our suggested method performs very well (with the exception of mini FlowPic of 64x64 for QUIC Human dataset), and is much better than the other two methods. For the script-based QUIC datasets, which are the easiest to classify, we reach 93.4% accuracy with only 3 labeled samples per class. For the QUIC human dataset, we reach 82.3% for 7 samples, and for the ISCX Video and VoIP dataset we reach 87.5% for 7 samples.

It interesting to compare these results with previous works. For the QUIC dataset script, Rezaei and Liu [16] reached 93.3% accuracy with 10 labeled samples per class. Our method achieves 93.4% accuracy with only 3 samples, and 94.5% with 10 samples. In their previous paper [15], they got better results, but as they point out themselves: their approach takes sampled data packets which means that it needs to observe a large portion of a flow before performing the classification, which is not suitable for online applications.

4 CONCLUSIONS

This work presents a practical solution for traffic classification with only a few samples per class. In all our experiments we achieved good accuracy with less than 10 samples per class. This is the first time that unsupervised contrastive learning is suggested for traffic classification. We note that the winning size architecture changes between datasets, but choosing a mini FlowPic has both engineering advantages and is never too far from optimum. Understanding this behavior is left for future research.

ACKNOWLEDGEMENT

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] Giuseppe Aceto, Domenico Ciuonzo, Antonio Montieri, and Antonio Pescapé. 2019. Mobile Encrypted Traffic Classification Using Deep Learning: Experimental Evaluation, Lessons Learned, and Challenges. *IEEE Transactions on Network and Service Management* 16 (June 2019). Issue 2.

[2] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*. PMLR, 1597–1607.

[3] Z. Chen, K. He, J. Li, and Y. Geng. 2017. Seq2Img: A sequence-to-image based approach towards IP traffic classification using convolutional neural networks. In *IEEE International Conference on Big Data (Big Data)*. 1271–1276.

[4] A. Dainotti, A. Pescape, and K. C. Claffy. 2012. Issues and future directions in traffic classification. *IEEE Network* 26, 1 (January 2012), 35–40.

[5] Gerard Draper-Gil, Arash Habibi Lashkari, Mohammad Saiful Islam Mamun, and Ali A. Ghorbani. 2016. 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, 407–414.

[6] Adil Fahad, Zahir Tari, Ibrahim Khalil, Ibrahim Habib, and Hussein Alnuweiri. 2013. Toward an efficient and scalable feature selection approach for internet traffic classification. *Computer Networks* 57, 9 (2013), 2040 – 2057.

[7] He Huang, Haojiang Deng, Jun Chen, Luchao Han, and Wei Wang. 2018. Automatic Multi-task Learning System for Abnormal Network Traffic Detection. *International Journal of Emerging Technologies in Learning* 13, 4 (2018).

[8] Ren-Hung Hwang, Min-Chun Peng, Chien-Wei Huang, Po-Ching Lin, and Van Linh Nguyen. 2020. An Unsupervised Deep Learning Model for Early Network Traffic Anomaly Detection. *IEEE Access* 8 (2020), 30387–30399.

[9] Auwal Sani Iliyasu and Huifang Deng. 2019. Semi-supervised encrypted traffic classification with deep convolutional generative adversarial networks. *IEEE Access* 8 (2019), 118–126.

[10] Arash Habibi Lashkari, Gerard Draper Gil, Mohammad Saiful Islam Mamun, and Ali A. Ghorbani. 2017. 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, 253–262.

[11] Sam Leroux, Steven Bohez, Pieter-Jan Maenhaut, Nathan Meheus, Pieter Simoens, and Bart Dhoedt. 2018. Fingerprinting encrypted network traffic types using machine learning. In *IEEE/IFIP Network Operations and Management Symposium (NOMS)*. 1–5.

[12] Wenhao Li, Xiao-Yu Zhang, Haichao Shi, Feng Liu, Yunlin Ma, and Zhaoxuan Li. 2020. *A Glimpse of the Whole: Path Optimization Prototypical Network for Few-Shot Encrypted Traffic Classification.* Technical Report. arXiv–2010 pages.

[13] Andrew Moore, Denis Zuev, and Michael Crogan. 2005. *Discriminators for use in flow-based classification.* Technical Report RR-05-13. Queen Mary Uni. of London.

[14] Tao Qin, Lei Wang, Zhaoli Liu, and Xiaohong Guan. 2015. Robust application identification methods for P2P and VoIP traffic classification in backbone networks. *Knowledge-Based Systems* 82 (2015), 152 – 162.

[15] Shahbaz Rezaei and Xin Liu. 2019. How to achieve high classification accuracy with just a few labels: A semi-supervised approach using sampled packets. In *Industrial Conference on Data Mining (ICDM)* (NYC, NY, USA).

[16] Shahbaz Rezaei and Xin Liu. 2020. Multitask learning for network traffic classification. In *2020 29th International Conference on Computer Communications and Networks (ICCCN)*. IEEE, 1–9.

[17] Sangita Roy, Tal Shapira, and Yuval Shavitt. 2022. Fast and lean encrypted Internet traffic classification. *Computer Communications* 186 (2022), 166–173.

[18] Ola Salman, Imad H Elhajj, Ali Chehab, and Ayman Kayssi. 2019. A machine learning based framework for IoT device identification and abnormal traffic detection. *Transactions on Emerging Telecommunications Technologies* (2019), e3743.

[19] Tal Shapira and Yuval Shavitt. 2019. FlowPic: Encrypted Internet Traffic Classification is as Easy as Image Recognition. In *IEEE Workshop on Network Intelligence: Machine Learning for Networking (NI 2019)*. 680–687.

[20] Tal Shapira and Yuval Shavitt. 2021. FlowPic: A Generic Representation for Encrypted Traffic Classification and Applications Identification. *IEEE Transactions on Network and Service Management* 18, 2 (2021), 1218 – 1232.

[21] Petr Velan, Milan Čermák, Pavel Čeleda, and Martin Drašar. 2015. A survey of methods for encrypted traffic classification and analysis. *International Journal of Network Management* 25, 5 (2015), 355–374.

[22] Wei Wang, Ming Zhu, Xuewen Zeng, Xiaozhou Ye, and Yiqiang Sheng. 2017. Malware traffic classification using convolutional neural network for representation learning. In *2017 International conference on information networking (ICOIN)*. IEEE, 712–717.

[23] Lixuan Yang, Alessandro Finamore, Feng Jun, and Dario Rossi. 2021. *Deep Learning and Traffic Classification: Lessons learned from a commercial-grade dataset with hundreds of encrypted and zero-day applications.* Technical Report 2104.03182. arXiv.

A CNN ARCHITECTURE

A.1 Supervised classifier CNN architecture

We use the same LeNet-5 based architecture, which was used by [20] for classifying FlowPics and a similar version for the Mini FlowPic. As depicted in Figures 6 and 7, our architectures comprise seven layers, the ReLU activation function is applied to the output of every convectional and fully-connected layer and dropout with probabilities of 0.25 and 0.5 are used in order to reduce overfitting.

For the supervised training, we use the categorical cross entropy cost function, Adam gradient-based optimizer, learning rate 0.01, batch size 32 and early stopping callback which monitor the validation loss with threshold of 0.001 in absolute term and patient of 5 epochs.

A.2 Unsupervised representation extractor CNN architecture

For the representation extractor $f(\cdot)$ we employed the 5 first layers of the CNN architectures described in A.1 and replaced the last 2 layers with 2 linear layers sized 120 and 30. Thus, resulting with a 120 dimensional representation

vector $h = f(FlowPic)$ and $z = g(h)$ dimensional similarity vector.

For the unsupervised training process we apply Loss $\mathcal{L}_{\text{SimCLR}}$ with temperature parameter $\tau = 0.07$, Adam gradient-based optimizer, learning rate 0.001, batch size 32 and early stopping callback that in each batch count all the $\tilde{\mathbf{x}}_i = t(\mathbf{x}), \quad \tilde{\mathbf{x}}_j = t'(\mathbf{x})$ which are in the top 5 closest samples to each other in the embedding space (this is correlative to the aim that similar sample pairs stay close together in the embedding space) with patient of 3 epochs of non-improvement for stopping the training.

A.3 linear classifier

We use a simple single layered Neural Network which fit to the formulae $y = xA^T + b$ where A, b are the classifier parameters, x is the input vector and y is the output. x is 120 (or 10 for the statistical features vectors) dimensional vector and y is 5-dimensional vector for Quic dataset and 10-dimensional vector for the ISCX dataset.

For the training, we use the categorical cross entropy cost function, Adam gradient-based optimizer, learning rate 0.01, batch size 32 and early stopping callback which monitor the training loss with threshold of 0.001 in absolute term and patient of 5 epochs.

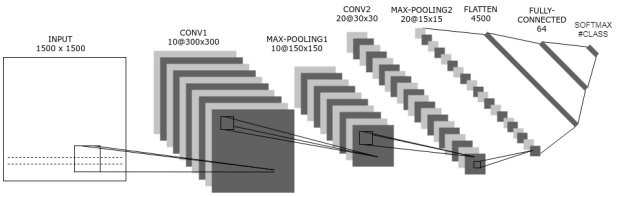


Figure 6: An illustration of the Deep Learning Architecture for FlowPic

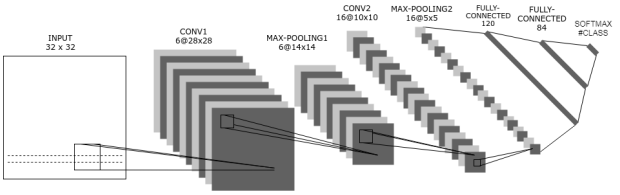


Figure 7: An illustration of Deep Learning Architecture for 32x32 Mini FlowPic