Deep Learning for Encrypted Traffic Classification: An Overview

Shahbaz Rezaei and Xin Liu

The authors introduce a general framework for deep-learning-based traffic classification. They present commonly used deep learning methods and their application in traffic classification tasks. Then they discuss open problems, challenges, and opportunities for traffic classification.

Abstract

Traffic classification has been studied for two decades and applied to a wide range of applications from QoS provisioning and billing in ISPs to security-related applications in firewalls and intrusion detection systems. Port-based, data packet inspection, and classical machine learning methods have been used extensively in the past, but their accuracy has declined due to the dramatic changes in Internet traffic, particularly the increase in encrypted traffic. With the proliferation of deep learning methods, researchers have recently investigated these methods for traffic classification and reported high accuracy. In this article, we introduce a general framework for deep-learning-based traffic classification. We present commonly used deep learning methods and their application in traffic classification tasks. Then we discuss open problems, challenges, and opportunities for traffic classification.

Introduction

Traffic classification, the categorization of network traffic into appropriate classes, is important to many applications, such as quality of service (QoS) control, pricing, resource usage planning, and malware/intrusion detection. Because of its importance, many different approaches have been developed to accommodate the diverse and changing needs of different application scenarios. In particular, new advances in communications, including encryption and port obfuscation, have raised additional challenges in network classification.

Traffic classification techniques have evolved significantly over time. The first and easiest approach uses port numbers. However, its accuracy has declined because newer applications either use well-known port numbers to disguise their traffic or avoid using standard registered port numbers. Despite its inaccuracy, the port number is still widely used either alone or in tandem with other features in practice. The next generation of traffic classifiers, relying on payload, called data packet inspection (DPI), focuses on finding patterns or keywords in data packets. These methods are only applicable to unencrypted traffic and have high computational overhead. As a result, a new generation of methods, based on flow statistics, emerged. These methods rely on statistical or time series features, which enable them to handle both encrypted and unencrypted traffic. These methods usually employ classical machine learning (ML) algorithms, such as random forest (RF) and *k*-nearest neighbor (KNN). However, their performance heavily depends on human-engineered features, which limit their generalizability.

Deep learning obviates the need to select features by a domain expert because it automatically selects features through training. This characteristic makes deep learning a highly desirable approach for traffic classification, especially when new classes constantly emerge and patterns of old classes evolve. Another important characteristic of deep learning is that it has a considerably higher capacity of learning in comparison to traditional ML methods, and thus can learn highly complicated patterns. Combining these two characteristics, as an end-to-end approach, deep learning is capable of learning the nonlinear relationship between the raw input and corresponding output without the need to break the problem into small sub-problems of feature selection and classification.

Recent work has demonstrated the efficacy of deep learning methods for (encrypted) traffic classification. To achieve this goal, DL requires sufficient labeled data and adequate computation power. In this article, we overview the general framework for traffic classification. We provide general guidelines for classification tasks, including data collection and cleaning, feature selection, and model selection. Moreover, we discuss deep learning techniques and how they have been applied for traffic classification. Finally, open problems and future directions are discussed.

Overview of Classification Problems in Computer Networks

Figure 1 illustrates a general framework for traffic classification, comprising seven steps. Most existing work adopts all or part of the framework. We discuss the first four steps in this section and the last three in the next section, with a focus on DL-based approaches.

Problem Definition

The first step to build a traffic classifier is to clearly define the classification goal. Typical goals include QoS provisioning, resource usage planning, billing system customization, and intrusion/malware detection. To serve its goal, one can categorize traffic classes based on:

- Protocols (e.g., UDP or HTTP)
- Applications (e.g., Skype or WeChat)
- Traffic-types (e.g., browsing or downloading)
- Websites

Digital Object Identifier:
10.1109/MCOM.2019.1800819

Shahbaz Rezaei and Xin Liu are with the University of California, Davis.

0163-6804/19/$25.00 © 2019 IEEE

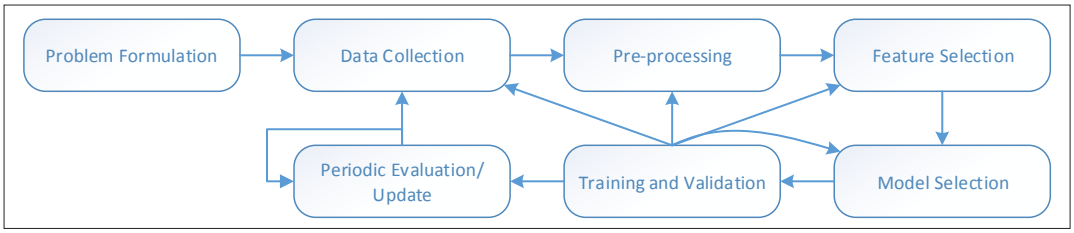

Figure 1. General framework to build a network classifier.

- User actions (e.g., posting comments or sending voice messages)
- Operating systems
- Browsers and so on

Hence, the goal is to label each flow with corresponding traffic classes. A flow is usually determined by a 5-tuple: source IP, destination IP, source port, destination port, and protocol.

Furthermore, traffic classification can also be categorized into two sub-classes: online and offline. Online classification usually refers to cases where flows need to be classified as quickly as possible, usually within the first few packets. For instance, for QoS provisioning and routing, classification needs to be online because the classification output is directly used for decisions on the current flow. For other applications, such as billing systems, classification can be offline.

While traffic classification applies to vastly different scenarios, most studies share two ubiquitous aspects. First, the input data for classification is raw packet data, part of it, or information directly derived from it. Second, similar ML algorithms are used. The focus of the article is on encrypted application/traffic-type classification. However, the same methodology may be used for other classification problems with minor modifications.

Data Collection

One of the most important requirements for training a deep learning model is a large and representative dataset. Although there are a few public datasets available for research purposes, there is no commonly agreed-upon dataset for most traffic-related classification problems. Possible reasons include:

- The number of possible traffic classes is enormous, and it is practically impossible for one dataset to contain all traffic types.
- There are no commonly accepted data collection and labeling methods.
- Different collection methods and scenarios result in different feature availability and distributions.

In practice, researchers often collect a dataset specific to their classification goal. To do so, the first step is to determine a data collection location. Data collection can happen at the client or server side of a communication channel, at the edge of the network, at the core of the network, or any place in between. Collection point dramatically affects available features, reliable labeling, and generalization.

Reliable Labeling: Correct labels are crucial to the performance of traffic classification methods. However, labeling data is not trivial. Some studies used free DPI modules, such as nDPI and libprotoident, to label captured data. In such cases, the accuracy of the labels, and thus any corresponding classification algorithms, is limited by that of the DPI methods. Furthermore, such methods do not usually work for all encrypted traffic types.

A controlled environment at the client side of the communication would be the easiest place to label the data. This solution is only practical when the capturing point is close enough to the data source to make sure that there is no other source to affect the labeling. Moreover, even in a fully controlled environment, it is not easy to distinguish and remove background traffic completely. It has been shown that 70 percent of smartphones' traffic is background traffic, and only 30 percent is directly related to user interactions [1]. Despite the limitations, capturing each class in a controlled environment is the most commonly adopted strategy in practice.

Available Features: Useful information in packets is not always available. Packets captured at wireless links or cellular communications are encrypted at layer 2, and consequently useful upper-layer header fields are not in plain text. Furthermore, at some capturing points, such as a router in the center of an Internet service provider (ISP), one may only capture one direction of a flow due to the asymmetric nature of routing in the Internet. Moreover, interarrival time may get distorted when traffic is aggregated, which is more severe at the core of ISPs. This phenomenon transforms the distribution of interarrival time, which depends on network conditions, traffic load, and time. Packet length may also change when the traffic passes through a tunnel or proxy. Finally, all these changes also affect the statistical features obtained from the entire flow. Hence, a model trained on a dataset captured at one capturing point may not be as accurate when used at another point.

Representative Dataset: A representative dataset should contain diverse and abundant samples from each class. It has been shown that the accuracy drops by as much as 26 percent when the operating system (OS)/vendor is different in the training and test set [2]. Furthermore, a model may overfit to user-specific features rather than traffic-specific features if the dataset contains interactions of only one or a few users. It is also a big limitation on studies that capture the traffic generated by scripts [3] which probably have more deterministic behavior. In general, a dataset captured further away from the client side of the communication, for example, at the core of ISPs where diverse traffic is observable, is less exposed to this issue. The best way to guarantee that the trained model on the dataset is representative is to test the model on a test set that comes from a different device/user configuration than the training set.

One of the most important requirements for training a deep learning model is a large and representative dataset. Although there are a few public datasets available for research purposes, there is no commonly agreed-upon dataset for most traffic-related classification problems.

Data cleaning and pre-processing significantly affect the performance of classification. In a network environment, some relatively common events can change the packet-level feature distribution. For instance, packet retransmissions, duplicate acks, and out-of-order packets may change the traffic pattern of an application.

Paper	Category	DL method	Online	Features	Year
Wang-2018 [10]	Intrusion detection	CNN+LSTM	✓	Header+payload	2018
Rezaei [3]	APP/OS identification	CNN	×	Sampled time series	2018
Aceto [7]	APP classification	CNN/LSTM/SAE/MLP	✓	Header+payload	2018
Vu [12]	Traffic identification	AC-GAN	×	Statistical	2017
Wang-2017 [8]	Traffic identification	CNN	✓	Header+payload	2017
Seq2Img [9]	APP/protocol identification	RKHS+CNN	✓	Time series	2017
Lotfollahi [6]	APP/traffic identification	CNN/SAE	×	Header+payload	2017
Lopez-Martin [5]	Mixed-type classification	CNN+LSTM	✓	Header+time Series	2017
Hochst [11]	Traffic identification	Autoencoder	×	Statistical+header	2017

Table 1. Overview of DL methods used for traffic classification.

Dataset Pre-Processing

Data cleaning and pre-processing significantly affect the performance of classification. In a network environment, some relatively common events can change the packet-level feature distribution. For instance, packet retransmissions, duplicate acks, and out-of-order packets may change the traffic pattern of an application. Some studies reported improvement upon removing such packets [4], and some reported no difference [2]. That is because different datasets and features are used for classification. For example, methods that use statistical features of entire flows are probably immune to a few unrelated packets. On the other hand, methods that use the first few packets for classification might be affected more. This pre-processing step is often ignored due to its computational complexity.

Another pre-processing step that is crucial to the performance of DL methods is normalization. In this step, all input features are scaled to have a value in the range [-1, +1] (or [0, 1]). This allows the gradient-based methods to converge faster and equalizes the importance of all features when computing the distance between data points.

Features

State-of-the-art traffic classification methods use one or more categories of the following features.

Time Series: Time series features include packet length, inter-arrival time, and direction of consecutive packets. In many studies where these features were representative, the first few packets (up to the first 20 packets) have been shown to be enough for reasonable accuracy even for encrypted traffic [5]. Times series features of a set of sampled packets have also recently been shown to achieve good accuracy [3].

Header: This includes all useful header fields in a packet, typically layer 3 and layer 4 information, when unencrypted. In the pre-DL era, fields including port number, protocol, and packet length were carefully chosen by domain experts as representative features. In some recent DL-based approaches, entire packets are taken as input [6].

Payload Data: Even for encrypted traffic, information above the layer 4 header exists that can be exploited for classification. For instance, some studies have achieved high accuracy using TLS 1.2 handshake packets that contain plain text data.

Statistical Features: There are numerous statistical features that can be obtained from the entire flow, such as average packet length and minimum inter-arrival time. Many papers used these features and demonstrated high accuracy. However, to obtain statistical features, a classifier is required to observe all or a large portion of a flow and thus is only suitable for offline classification. Moreover, in some cases like application classification, statistical features can be affected by user-specific behaviors, OS-specific patterns, or network-specific conditions. Hence, the dataset should be collected with more care.

Although time series and statistical features might be slightly different for unencrypted and encrypted versions of the same traffic, they are available regardless of encryption. Hence, models trained on unencrypted traffic may also work with encrypted traffic. On the other hand, payload data and some header information, such as layer 4 information of traffic encrypted by IPsec, might not exist in plain text. However, in these cases, there are still unencrypted fields available during handshake that can be used for classification.

Deep Learning Techniques

In this section, we briefly introduce some recent papers for each deep learning method, the summary of which is shown in Table 1. Then we complete our seven-step framework by explaining the model selection and evaluation in detail.

Multi-Layer Perceptron

For network traffic classification, pure multi-layer perceptron (MLP) has rarely been used due to its complexity and low accuracy. In [7], many deep learning methods are compared to the RF algorithm to show the performance gap. Three mobile datasets with different numbers of labels are used. Many deep learning methods outperform RF. However, the experiment settings are not completely fair because input features used for RF, MLP, and other DL methods are different. Hence, the results should not be considered as a comprehensive comparison of ML methods.

Convolutional Neural Networks

The simplest convolutional neural network (CNN) model proposed in [8] basically represents each flow or session with a one-dimensional vector to feed the CNN model. The proposed CNN model

has two convolutional, two pooling, and two fully connected layers. The model normalizes the bytes in each packet and uses only the first 784 bytes. The model is evaluated on an encrypted application dataset of 12 classes and shows a significant improvement over the C4.5 approach that uses time series and statistical features. In [9], the authors also use CNN with two convolutional, two pooling, and three fully connected layers. The approach uses reproducing kernel Hilbert space (RKHS) embedding and converts time series data into two-dimensional images. The proposed CNN model outperforms classical ML methods and MLP in protocol and application classification. A semi-supervised approach based on a simple one-dimensional CNN is used in [3] to classify five Google applications. The model is trained to predict the statistical features of the entire flow from a few sampled packets with a large unlabeled dataset. Then the weights are transferred to a new model and re-trained for application classification with only a few labeled samples. This work shows the possibility of using sampled time series features instead of the first *n* packets, which is more feasible for high bandwidth operational networks [3].

Recurrent Neural Networks

For network classification tasks, mixed models are reported to outperform pure LSTM or CNN models [5]. To capture both spatial and temporal features of a flow, both the CNN and recurrent neural network (RNN) are used in [10, 5]. Aside from minor differences, both studies take the content of the first 6 to 30 packets of a CNN model followed by an RNN or LSTM model. Although the input features, neural network architectures, and datasets are different, both studies report high accuracy.

Auto-Encoders

Auto-encoders (AEs) are usually used in an unsupervised fashion to obtain smaller representation of input data that can be used later as part of a classifier. For instance, in [11], an auto-encoder is used to reconstruct the input. Then a softmax layer is applied to the encoded internal representation of the auto-encoder and achieve a moderate accuracy. A private dataset with seven traffic types is used. Moreover, 9 statistical features of 12 intervals and both flow directions are used as input. In [6], the authors take header and payload data to train a one-dimensional CNN and a stacked AE model. Both models show high accuracy, while the CNN model marginally outperforms the stacked AE model.

Generative Adversarial Networks

Generative models can be used to handle the dataset imbalance problem in network traffic classification. The imbalance problem refers to scenarios where the number of samples for each class varies considerably. In such cases, ML algorithms usually have difficulties predicting minority classes. The most frequent and easiest approach to deal with an imbalanced dataset is oversampling by duplicating samples of minor classes, or undersampling by removing some samples from major classes. In [12], a generative adversarial network (GAN) is used to generate synthesized samples to handle the imbalance problem. A public dataset with two classes, SSH and non-SSH, and 22 statistical features for the input are used. Deep models are only used to generate synthesized data. For classification, classical ML algorithms, including SVM, RF, and decision tree, are trained.

Model Selection

Several factors affect the choice of DL models for network traffic classification. The most important one is input features. Features directly affect not only the accuracy, but input structure/dimension, which influences computational complexity and number of packets for classification (memory complexity). Next, one should choose a suitable model accordingly. Here, we do not cover header features alone; we only cover it in conjunction with other features because header features alone are often not effective enough for classification.

The choice of input feature and ML method are highly correlated. Furthermore, the size of the dataset also affects the choice of the model. For instance, deep methods are not suitable when the dataset is small. Assuming a large dataset, three commonly used input features and corresponding suitable models are described below:

Time Series+Header: Since time series features are barely affected by encryption, they have been widely applied to various scenarios and datasets. The first few packets, from 10 to 30 packets, are reported to be enough for classification in many datasets [9, 5]. Sampled packets from the entire flow are also shown to achieve promising accuracy [3]. Classical ML algorithms and MLP models work well when the number of packets, representing the input dimension, is small. For high-dimensional inputs, CNN and LSTM are reported to be more accurate [5]. However, even for relatively low-dimensional inputs, the CNN model is used [5, 3]. However, in general, computational complexity and training time of deep models are higher than those of classical ML algorithms.

Payload+Header: In current encrypted traffic, the first few packets that contain handshake information are typically unencrypted and have been successfully used for classification. Due to the high dimensionality of the input (large number of bytes in payload), classical ML methods and MLP do not work well. In such cases, CNN or a combination of CNN and LSTM are reported to achieve high accuracy [6–8, 10]. It is possible to also use time series features alongside payload information to slightly improve accuracy, but this barely changes the input dimension or the choice of model.

Statistical Features: The number of statistical features, and consequently the input dimension, is often small. Hence, most papers used classical ML methods or in rare cases MLP for statistical features. Although most studies obtained statistical features by observing the entire flow, it has been shown that obtaining statistical features from the first 10 to 180 packets, depending on datasets and the choice of statistical features, might be sufficient for classification. Even though statistical features allow us to build a simpler classifier based on classical ML algorithms, it may not be suitable for online and fast classification because it needs to capture a large number of packets to obtain dependable statistical features from a flow.

Table 2 summarizes features, the corresponding models, and their properties. Note that there is no guarantee that these approaches work for all data-

Several factors affect the choice of DL models for network traffic classification. The most important one is input features. Features directly affect not only the accuracy, but input structure/dimension, which influences computational complexity and number of packets for classification (memory complexity).

Feature	Time series+header	Payload+header	Statistical
Model	Classical ML/MLP/CNN/LSTM	CNN/CNN+LSTM	Classical ML/MLP
Computational complexity	Low/medium	High	Low
Number of packets needed	Medium	Low	High

Table 2. Guide for model and feature selection.

sets. That is the reason why one might need to go to the data collection step if there is not enough data, or to feature or model a selection step if the chosen features or model are not suitable.

Training and Validation

The training and validation step is similar to other DL applications where a model's hyper-parameters are tuned to obtain the best accuracy. Typically, a dataset is divided into three separate sets: training, validation, and test sets. A model is trained on the training set, and the accuracy of the validation set is observed to tune the model's hyper-parameters. Finally, unbiased accuracy is obtained by using the test set. The detailed best practices of the last two steps are outside the scope of this article. One can read a training and validation guideline on any other application and apply the same best practices here.

Periodic Evaluation/Update

The last step, periodic evaluation/update, has not been comprehensively studied yet. In most network-related applications, traffic characteristics of classes are always changing. Moreover, new traffic classes, called zero-day applications, are constantly emerging. However, only a limited number of papers studied such challenges, and they are still open problems worth more comprehensive analysis.

Open Problems and Opportunities

In traffic classification, unencrypted traffic classification has been extensively studied, and many commercial and free tools have been developed. Encrypted traffic classification is a harder task due to the lack of representative features, but a few studies have shown successful classification of TLS 1.2 and VPN traffic in UDP mode. It is still not clear whether these methods can handle the significantly larger number of classes common in operational networks. There are also many unsolved problems in traffic classification that we introduce in this section.

Stronger Encryption Protocols

Traffic classification for stronger encryption protocols, in particular QUIC and TLS 1.3, has not been well investigated. Most browsers, including Chrome and Firefox, have already implemented the TLS 1.3 draft version and in the future, most applications and websites will adopt these stronger encryption protocols. Previous studies on TLS 1.2 mainly used plain text fields during the handshake. But, with the introduction of 0-RTT connectivity in TLS 1.3 and QUIC, only a few fields in the first packet remain unencrypted, and it is unclear if they suffice for classification. In [3], QUIC is classified using sampled time series features. But the classification is limited to five Google applications, and the accuracy was high when only training and test sets are captured based on script-generated traffic.

Multi-Label Classification

A single flow may contain more than one class label, referred to as a multiplexed stream. For instance, traffic that passes through a tunnel may contain several applications that share the same 5-tuple. QUIC also may contain several classes of traffic. There is no method in traffic classification or related literature to deal with these cases. The first and most difficult challenge is how to collect and label such traffic appropriately.

Middle Flow Classification

Around 90 percent of all flows are short-lived. In certain applications, such as traffic engineering, one may want to focus on long flows. However, if the classification method relies on the first few packets, an ISP should store the first few packets of all flows, which is a huge burden. On the other hand, if the classification method works with packets in the middle of the flows, ISPs can wait and detect elephant flows, and then perform classification by capturing a few packets from the middle of the flow. This will dramatically reduce the memory and computational overhead. A few studies have shown that the accuracy is higher when the first few packets are involved in classification, but no comprehensive study has been conducted on using a set of packets from an arbitrary point in the middle of the flow. Note that some studies divide the entire flow into several bursts and then classify each burst to detect different user actions [13]. This means that the beginning of the burst should also be detected, and the capturing process must be started at this specific point. Moreover, it is not clear whether this method also works for other classification problems rather than user actions. In [3], the authors used a fixed number of sampled packets from different parts of a flow for classification and achieved moderate accuracy. However, high accuracy from the middle of a flow is still an open problem.

Zero-Day Applications

Zero-day applications refer to traffic classes that are unknown and the classifier has not trained on. It has been shown that in some cases zero-day applications can constitute up to 60 percent of flows and 30 percent of bytes of network traffic [14]. Despite the importance, it is in a nascent stage, and only a few recent studies [14] proposed solutions, which often rely on detecting unlabeled clusters and then labeling them. In the ML community, active learning, where a model selects which data points should be labeled, has been studied for many years. In a recent study on classifying images of characters [15], a combination of reinforcement learning and LSTM is used to perform one of the two possible actions: predicting the class or asking for a new label. There are many useful ideas in the ML community that can be adopted to solve the zero-day application problem.

Transfer Learning and Domain Adaptation

It is not always possible to collect a large enough representative dataset. It is often easier to obtain large datasets captured for other tasks, which

may help the model to extract common features. Moreover, training a deep model usually takes from a few hours to a few days or weeks, depending on the model size and dataset. Since retraining a model often converges faster, it is preferable to retrain a model that has already been trained for a similar task. Transfer learning and domain adaptation are the two widely used approaches to achieve such a goal.

Transfer learning allows a model trained on a source task to be used on a different target task. The assumption is that the input distributions of the source and destination tasks are similar. This process only works when the features learned by the model are not specific to the source task. Since the model is already trained to capture useful features, the retraining process needs significantly less labeled data and training time. In the case of network traffic classification, a publicly available dataset can be used to pre-train a model that can be further tuned for another traffic classification task with fewer labeled samples.

A recent paper [3] used this approach by transferring the weights of a pre-trained CNN model to a new model that was later trained to classify Google applications. The model was first trained to predict statistical features of the entire flow from sampled packets, which does not need human effort for labeling. Then it was transferred and re-trained with a small labeled dataset containing only 20 labeled samples for each class. The paper also showed that the model pre-trained on an unrelated public dataset can still be used for transfer learning. However, the accuracy was limited to below 85 percent.

Unlike transfer learning where source and target tasks (i.e., their class labels) are different, domain adaptation deals with cases where the task is the same, but the input distribution of the source and target is different. Although it is different from transfer learning, similar techniques have been used to solve both problems. An example in the context of network traffic classification would be to train a traffic classifier model with a dataset captured at the client side of the communication and then adopt the model to classify traffic at the core of the network where the data distribution is different. Another example is the case where a model is re-trained periodically based on domain adaptation techniques to capture new patterns for classes whose features are constantly changing, which are common in the current Internet. Despite their usefulness, these strategies have not been extensively adopted for network classification task yet.

Multi-Task Learning

This approach refers to scenarios where more than one loss function are optimized. One typical approach is to share the hidden layers among all tasks, while each task has its own output layer. It has been shown that this reduces the risk of overfitting and helps the model find relevant features faster. This works when the input data of different tasks is generated from a similar distribution or can be generated using a set of transformations from one another. As a result, it may be possible to use additional datasets and define new (auxiliary) tasks, which helps train the whole model. This dramatically augments the dataset and improves generalization.

There are potentially many ways to define auxiliary tasks for traffic classification without the need for additional labeling. For instance, detecting TCP/UDP classes, predicting the average packet length of flows, and detecting mice/elephant flows do not need manual human labeling. Many variations of multi-task learning were used successfully for natural language processing and computer vision. However, it has not been studied for network traffic classification.

Training a deep model usually takes from a few hours to a few days or weeks, depending on the model size and dataset. Since retraining a model often converges faster, it is preferable to retrain a model that has already been trained for a similar task. Transfer learning and domain adaptation are the two widely used approaches to achieve such a goal.

Acknowledgment

The work was partially supported by NSF through grants CNS-1547461, CNS-1718901, and IIS-1838207.

References

[1] T. Stober *et al.*, "Who do You Sync You Are? Smartphone Fingerprinting via Application Behaviour," *Proc. Sixth ACM Conf. Security and Privacy in Wireless and Mobile Networks*, Apr. 2013, pp. 7–12.
[2] G. Aceto *et al.*, "Traffic Classification of Mobile Apps through Multi-classification," *Proc. IEEE GLOBECOM*, Dec. 2017, pp. 1–6.
[3] S. Rezaei and X. Liu, "How to Achieve High Classification Accuracy with Just a Few Labels: A Semisupervised Approach Using Sampled Packets," 2018; arxiv.org/abs/1812.09761, accessed Apr. 2019.
[4] R. Dubin *et al.*, "I Know What You Saw Last Minute — Encrypted HTTP Adaptive Video Streaming Title Classification," *IEEE Trans. Info. Forensics and Security*, vol. 12, Dec. 2017, pp. 3039–49.
[5] M. Lopez-Martin *et al.*, "Network Traffic Classifier with Convolutional and Recurrent Neural Networks for Internet of Things," *IEEE Access*, vol. 5, 2017, pp. 18,042–50.
[6] M. Lotfollahi *et al.*, "Deep Packet: A Novel Approach For Encrypted Traffic Classification Using Deep Learning," 2017; arxiv.org/abs/1709.02656, accessed Apr. 2019.
[7] G. Aceto *et al.*, "Mobile Encrypted Traffic Classification Using Deep Learning," *Proc. Network Traffic Measurement and Analysis Conf.*, June 2018, pp. 1–8.
[8] W. Wang *et al.*, "End-to-End Encrypted Traffic Classification with One-Dimensional Convolution Neural Networks," *Proc. IEEE Int'l Conf. Intelligence and Security Informatics*, July 2017, pp. 43–48.
[9] Z. Chen *et al.*, "Seq2Img: A Sequence-to-Image Based Approach Towards IP Traffic Classification Using Convolutional Neural Networks," *Proc. IEEE Int'l Conf. Big Data*, Dec. 2017, pp. 1271–76.
[10] W. Wang *et al.*, "HAST-IDS: Learning Hierarchical Spatial-Temporal Features Using Deep Neural Networks to Improve Intrusion Detection," *IEEE Access*, vol. 6, 2018, pp. 1792–1806.
[11] J. Hochst *et al.*, "Unsupervised Traffic Flow Classification Using a Neural Autoencoder," *Proc. IEEE 42nd Conf. Local Computer Networks*, Oct. 2017, pp. 523–26.
[12] L. Vu, C. T. Bui, and Q. U. Nguyen, "A Deep Learning Based Method for Handling Imbalanced Problem in Network Traffic Classification," *Proc. Eighth Int'l Symp. Info. and Commun. Tech.*, Dec. 2017, pp. 333–39.
[13] V. F. Taylor *et al.*, "Robust Smartphone App Identification via Encrypted Network Traffic Analysis," *IEEE Trans. Info. Forensics and Security*, vol. 13, no. 1, Jan. 2018, pp. 63–78.
[14] J. Zhang *et al.*, "Robust Network Traffic Classification," *IEEE/ACM Trans. Networking*, vol. 23, no. 4, 2015, pp. 1257–70.
[15] M. Woodward and C. Finn, "Active One-Shot Learning," Wksp. Deep Reinforcement Learning, NIPS, Dec. 2016.

Biographies

Shahbaz Rezaei [S'17] (srezaei@udavis.edu)} received his M.S. degree in information technology from the Sharif University of Technology, Tehran, Iran, in 2013. His research interests include computer networks, performance modeling, machine learning, and deep reinforcement learning. He is currently a Ph.D. student at the University of California, Davis.

Xin Liu [M'09, F'19] (liu@cs.ucdavis.edu) received her Ph.D. degree from Purdue University in 2002. She is currently a professor in the Computer Science Department, University of California, Davis. Her current research focuses on data-driven approaches in networking (i.e., using and developing machine learning and optimization techniques for network control and management).