Deep Learning for Encrypted Traffic Classification: An Overview

Shahbaz Rezaei, *Member, IEEE,* and Xin Liu, *Senior, Member,*

***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 have been declined due to the dramatic changes in the Internet traffic, particularly the increase in encrypted traffic. With the proliferation of deep learning methods, researchers have recently investigated these methods for traffic classification task and reported high accuracy. In this article, we present a general framework for deep-learning-based traffic classification. We discuss commonly used deep learning methods and their application in traffic classification tasks. Then, we present open problems and their challenges, as well as opportunities for traffic classification.**

***Index Terms*—Traffic classification, deep learning, machine learning.**

I. 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, malware detection, and intrusion detection. Because of its importance, many different approaches have been developed over years to accommodate the diverse and changing needs of different application scenarios. In particular, new advances in communications, including encryption and port obfuscation, raise additional challenges to network classification.

Traffic classification techniques have evolved significantly over time. The first and easiest approach is to use port numbers. However, its accuracy has been decreasing because newer applications either use well-known port numbers to disguise their traffic or do not use 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 or data packet inspection (DPI), focus on finding patterns or keywords in data packets. These methods are only applicable to unencrypted traffic and has 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 the human-engineered features, which limits their generalizability.

Deep learning obviates the need to select features by a domain expert because it automatically selects features through training. This characteristics makes deep learning a highly desirable approach for traffic classification, especially when new classes constantly emerge and patterns of old classes evolve. Another important characteristics 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 non-linear relationship between the raw input and corresponding output without the need to break the problem into the small sub-problems of feature selection and classification.

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

II. Overview of Classification Problems on Computer Network

Fig. 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 deep-learning-based approaches.

A. Problem Definition

The first step to build a network traffic classifier is to clearly define the goal of classification. Typical goals include QoS provisioning, resource usage planning, billing system customization, intrusion detection, and malware detection. To serve its goal, one can categorize traffic classes based on 1) protocols (e.g. UDP, TCP, FTP or HTTP), 2) applications (e.g. Skype, WeChat or Torrent), 3) traffic-types (e.g. browsing, downloading or video chat), 4) websites, 5) user actions (e.g. posting a comment or sending voice message), 6) operating systems, 7) browsers, and so on. Hence, the goal is to label each flow with corresponding traffic classes. A flow is usually

S. Rezaei and X. Liu are with Computer Science Department, University of California, Davis, USA (e-mails: srezaei@ucdavis.edu and liu@cs.ucdavis.edu).

Manuscript received April 19, 2005; revised August 26, 2015.

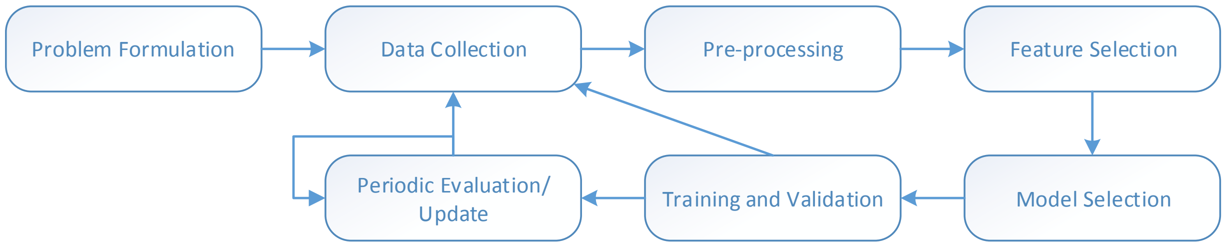


Fig. 1. General Framework to build a network classifier.

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 the cases where flows need to be classified as fast as possible, usually within the first few to tens of packets. For instance, for QoS provisioning and routing, classification needs to be online because the output of the classification 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: (a) the input data for classification is raw packet data, part of it, or information directly derived from it, (b) Similar ML algorithms are used. The focus of the article is on encrypted application/traffic-type classification. Yet, the same methodology may be used for other classification problems with minor modifications.

B. 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 and recent datasets available for research purpose[1], there is no commonly agreed-upon dataset for most traffic-related classification problems. Possible reasons include 1) the number of possible traffic classes is enormous, it is difficult for one dataset to contain all traffic types, 2) there is no commonly accepted data collection and labeling methods, 3) 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 edge of the network, at the core of the network or any place in between. Collection point can dramatically affect available features, reliable labeling and generalization, which are discussed next.

1) Reliable Labeling: Correct labels are crucial to the performance of traffic classification methods. However, labeling data is not always 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 work for encrypted traffic.

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 the fully controlled environment, it is not easy to distinguish and remove background traffic completely. It has been shown that 70% of the smartphones' traffic is background traffic and only 30% is directly related to the user interactions [1]. Despite the limitations, capturing data of each class in a controlled environment has been the most commonly-adopted strategy in practice.

2) Available Features: Useful information in packets are 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 accessible. Furthermore, at some capturing points, such as a router in the center of an 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 phenomena transforms the distribution of interarrival time and heavily depends on network conditions, traffic load, and time. Packet length may also change when the traffic passes through a tunnel, proxy, etc. 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 capturing point.

3) Representative Dataset: A representative dataset should contain diverse and abundant samples from each class to avoid overfitting. It has been shown that the accuracy drops by as much as 26% when 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 dataset contains interactions of only one or few users. It is also a big limitation on studies that captured the traffic generated by a script [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 an ISP 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 different device/user configuration than the training set.

[1] https://wand.net.nz/wits/
https://projects.cs.dal.ca/projectx/Download.html
http://www.unb.ca/cic/datasets/vpn.html

C. Dataset Pre-processing

Data cleaning and pre-processing significantly affect the performance of ML algorithms. 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 the entire flow are probably immune to a few unrelated packets. On the other hand, methods that use first few packets for classification might be affected more. Note that this pre-processing step is sometimes ignored due to its computational complexity.

Another pre-processing step which is crucial to the performance of deep learning methods is data 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.

D. Features

State-of-the-art traffic classification methods use one or more categories of the 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 first 20 packets have been shown to be enough for reasonable accuracy even for encrypted traffic.

Header: This includes all useful header fields in a packet, typically layer 3 and layer 4 information, when unencrypted. In pre-deep-learning era, fields including port number, protocol, and packet length, were carefully chosen by domain experts as representative features. In some recent approaches, especially deep-learning-based ones, entire packets are taken as input. Note that server IP addresses might be used to limit the range of traffic classes for better accuracy in operational networks. For instance, one can use Google's IP addresses to limit traffic classes to Google's applications. However, IP addresses should be used judiciously due to the widespread use of CDNs and the dynamic allocation of IP addresses.

Payload Data: Even for encrypted traffic, information above 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, such as average packet length, maximum packet length, and minimum inter-arrival time[2]. A large number of paper used these features and demonstrated high accuracy. However, to obtain statistical features a classifier is required to observe the entire or 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, network-specific conditions, etc. Hence, dataset should be collected with more care.

[2] A comprehensive list of statistical features can be found at https://centauri.stat.purdue.edu:98/netsecure/Papers/flowattributes_ademontigny.pdf

Although time series and statistical features might be slightly different for unencrypted traffic and the encrypted version of the same traffic, they are available regardless of encryption. Hence, methods depending on these features for unencrypted traffic may also work with encrypted traffic as well. On the other hand, payload data and some header information, for instance layer 4 information of traffic encrypted by IPsec, might not exist in plain text for encrypted traffic. However, in these cases, there are still unencrypted fields available during handshake that can be used for classification. It is worth mentioning that in some cases privacy policies and laws prohibit accessing or storing packet content which limit the use of payload features.

III. Deep Learning Techniques

Deep learning is a branch of ML algorithms which employs multiple layers of non-linear units to enhance modeling capacity and to automatically extract features. In this section, we introduce the most common deep learning architectures that have been used for traffic classification tasks. For each architecture, we briefly introduce some recent papers that used the architecture, the summary of which is shown in Table I. Then, we complete our 7-step framework by explaining the model selection and evaluation in detail.

A. Multi-Layer Perceptron (MLP)

Multi-layer perceptron (MLP) is the first neural network architecture and consists of an input layer, an output layer and several hidden layers of neurons. Each layer has several neurons which are densely connected to adjacent layers, as shown in Fig. 2(a). A neuron takes a weighted sum of its inputs and passes through a non-linear activation function to produce an output. Theoretically, a dense and deep enough MLP can estimate any arbitrary function. However, due to the huge number of parameters that a model need to learn, this model is usually very complex, inefficient and hard to train for an arbitrary complicated problem. Although the use of deep MLP alone has been declined, a few layers of fully-connected neurons, which can be considered as a MLP, is still used as small part of other models.

For network traffic classification, pure MLP has been rarely used. In [5], many deep learning methods were compared with random forest (RF) algorithm to show the performance gap. They used 3 mobile datasets with different number of labels. Many deep learning methods outperformed RF in two of the datasets. However, the experiment settings were not completely fair and the results should not be considered as a comprehensive comparison of ML methods.

B. Convolutional Neural Networks

Similar to MLP, convolutional neural networks (CNNs) also consist of several layers with learnable parameters. MLP fails to work well with high dimensional input leading to a large number of learnable parameters in hidden layers. CNN

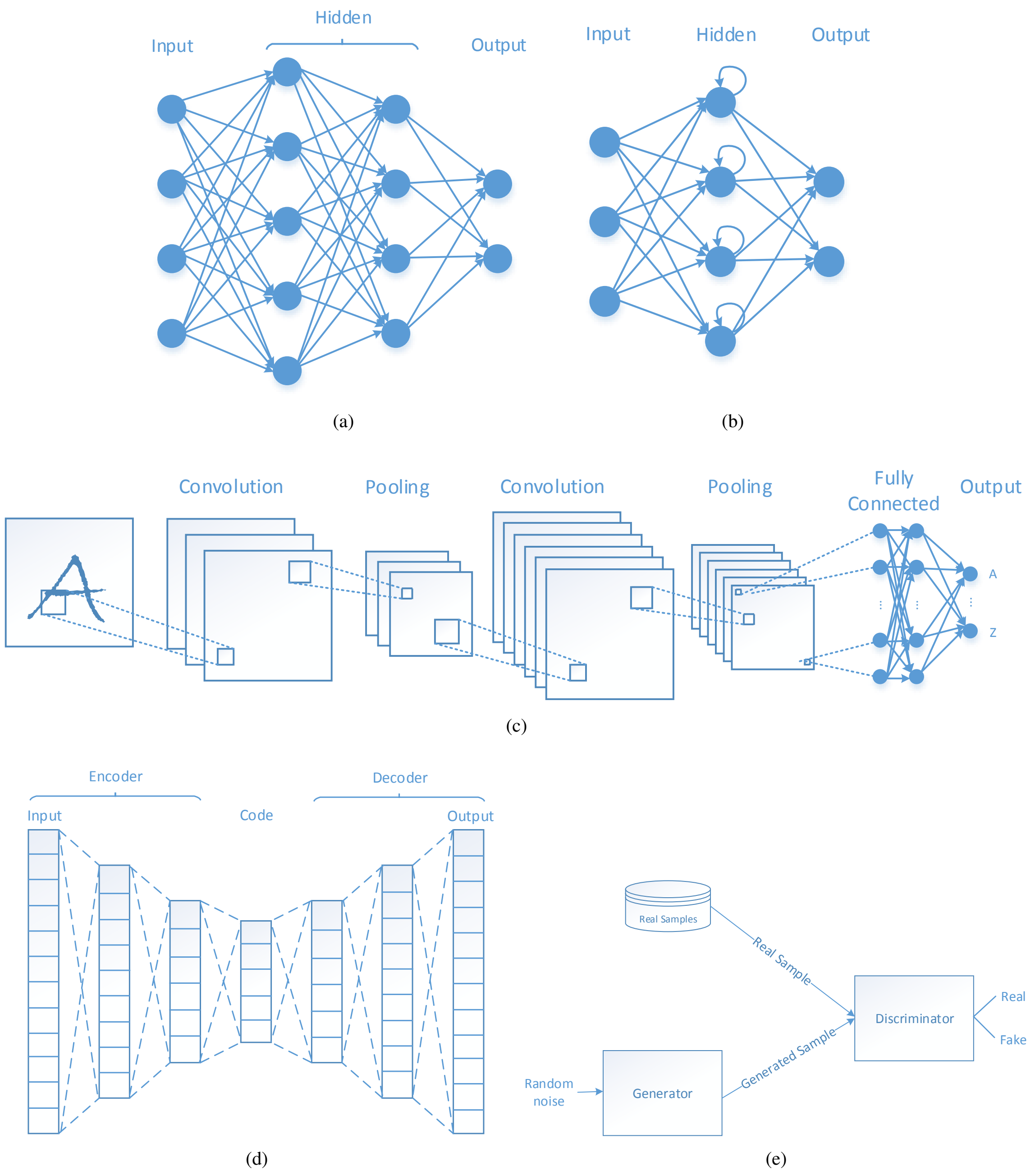


Fig. 2. Common deep learning models: a) MLP, b) CNN, c) RNN, d) AE, e) GAN.

architecture, shown in Fig. 2(c), solves this problem by using convolution layers. In a convolution layer, a set of small kernels with a small number of learnable parameters are used. The same set of kernels are used on the entire input to produce the output for the next layer. By using the same set of kernels in a layer, the number of learnable parameters are dramatically reduced. The use of these kernels on the entire input helps the model to also capture shift invariant features more easily. Pooling layer is also used after one or few convolution layer for subsampling. Moreover, fully connected layers are usually used for the last hidden layers.

CNN architecture showed significant improvement over traditional methods. In the last few years, many variations of CNN architecture have been introduced to allow deeper and more accurate models, such as GoogLeNet, ResNet and VGG.

The simplest CNN model was proposed in [6] which basically represented each flow or session with a 1-dimensional vector to feed the CNN model. Their CNN model had 2 convolutional, 2 pooling, and 2 fully connected layers. They normalized the bytes in each packet and used only the first 784 bytes. They evaluated their model on an encrypted application dataset of 12 classes and showed a significant improvement over C4.5 ML approaches that used time series and statistical features. In [7], authors also used CNN with 2 convolutional, 2 pooling, and 3 fully connected layers for protocol and application classification tasks. They used reproducing kernel Hilbert space (RKHS) embedding and converted the early time series data into 2-D images. Their CNN model outperformed classical ML methods and MLP in protocol and application classification task.

C. Recurrent Neural Networks

Recurrent neural networks (RNNs) are neural networks containing loops to store temporal information, as shown in Fig. 2(b). RNNs were designed specifically for sequential data where the output may depend not only on the last input but on the previous inputs as well. RNNs have been successfully applied to speech recognition, time series prediction, translation, and language modeling. Gradient vanishing and exploding, which makes learning long-term dependencies difficult (e.g. dependencies between inputs that are far apart), was a common obstacle in traditional RNNs. The long short-term memory (LSTM) was introduced to mitigate these problems by adding a set of gates that control when information is stored or removed.

For network classification tasks, mixed models have been reported to outperform pure LSTM or CNN models [11]. To capture both spatial and temporal features of a flow, both CNN and RNN were used in [8], [11] for different application. Aside from minor differences, both studies took the content of the first 6 to 30 packets to the CNN model followed by a RNN or LSTM model. Although the exact input features, the neural network architectures, and the datasets were different, they both reported high accuracy.

Despite their success in sequential data, LSTMs are not suitable for complex tasks which requires explicit and external memory. New architectures, such as memory networks and neural Turing machines (NTMs), have been recently introduced to embed explicit memory into the architecture, referred to as memory augmented neural networks (MANN). MANNs have been successfully applied in language modeling, question answering and one-shot learning. The performance of MANNs on the network classification task has not been studied yet.

D. Auto-encoders (AE)

Auto-encoders (AEs) are neural networks with significantly smaller hidden layers compared to the input and output that aims to reconstruct the input at the output, as shown in Fig. 2(d). The internal encoded representation can be used for data compression or dimensionality reduction. MLPs, CNNs and RNNs can all be used as a part of the AE architecture. AE is extensively used to initialize the weights of deep architectures. There are some variations of AE, such as denoizing auto-encoders (DAEs) that are trained to output intact input samples by taking a corrupted samples forcing the model to learn more robust features, and variational auto-encoders (VAEs) that aim to generate virtual examples from a target distribution. More complex architectures, called stacked auto-encoders (SAEs), stack up several AEs where the output of each one is the input of the next AE and the whole model is trained in a greedy layer-wise fashion. It is also possible to train a model, called hybrid learning framework, which combines AE with MLP, or other models, with labeled data from the beginning. Hence, the model learns both the input and output distribution at the same time. Such hybrid models are trained with multi-objective loss functions including standard output losses as well as the input reconstruction terms.

AEs are usually used in an unsupervised fashion to obtain smaller representation of input data which can be later used as a part of a classifier. For instance, in [12], an auto-encoder was used to reconstruct the input. Then, a softmax layer was applied to the encoded internal representation of the auto-encoder and they achieved a moderate accuracy. They used their own private dataset with 7 traffic types. Moreover, they used nine statistical features of 12 intervals and both flow directions as an input. In [10], authors took header and payload data to train a 1-D CNN and a SAE model on ISCX VPN non-VPN dataset. Both models showed high accuracy, but the CNN model marginally outperformed the SAE model.

E. Generative Adversarial Networks

Generative adversarial network (GAN) is an unsupervised technique that trains a generative and a discriminative model simultaneously. As shown in Fig. 2(e), the generator aims to generate (fake) examples of the target distribution and the discriminator model aims to distinguish between real and generated data. Both models are usually neural networks. The generator is first trained to maximize the error probability by discriminator. Then, the generator is fixed and discriminator is trained to minimize the error probability while real and generated data is fed in. The procedure is continued until it converges. Despite the difficulty of training and converging GANs, it has been used in many applications, such as creating realistic images, reconstructing 3D models from images, improving image quality, creating synthesized data for applications with scarce data.

Generative models can also be used to handle 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 to predict the minority classes correctly. The most frequent and easiest approach to deal with imbalance dataset is oversampling minority classes or undersampling majority classes. In [9], auxiliary classifier GANs (AC-GAN) was used to generate synthesized samples for supervised network classification task. The main difference between AC-GAN and GAN is that AC-GAN takes both a random noise and a class label as input so as to generate the sample of the input class label. They used a public dataset with two classes, SSH and non-SSH, and 22 statistical features for the classifier input. They used deep models only to generate synthesized data. For classification part, they used classical ML algorithms, including SVM, RF, and decision tree.

F. Model Selection

Several factors affect the choice of deep learning models for network traffic classification. The most important one is the choice of features. Features directly affects input structure and dimension which influences computational complexity and number of packets for classification (memory complexity). Next, one should choose a suitable model based on the chosen

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

TABLE I
OVERVIEW OF DEEP LEARNING METHODS USED FOR TRAFFIC CLASSIFICATION.

feature. Here, we only cover header features in conjunction with other features because header features are not always effective enough for classification. In headers, only port number, window size, and in some rare cases type of service (ToS) or fragmentation-related fields provides information useful for classification. Commonly used features are:

Time Series+Header: Since time series features are barely affected by encryption, it has been widely applied to various applications and datasets. The first few packets, from 10 to 30 packets, have been reported to be enough for classification in many datasets. Classical ML algorithms and MLP models work well when the number of packets, representing the input dimension, is small. For a larger number of packets, CNN and LSTM has been reported to be more accurate [11].

Payload+Header: In current encrypted traffics, the first few packets that contain handshake information are typically unencrypted and they have been successfully used for classification. Due to the high dimensionality of the input, classical ML methods and MLP do not work well. In such cases, CNN or combination of CNN and LSTM are reported to have high accuracy [6],[5],[10],[8]. 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 limited. Hence, most papers used classical ML methods or in rare cases MLP for these 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 the datasets and the choice of statistical features, might be sufficient for classification.

Table II summarized features, the corresponding models, and their properties. Note that there is no guarantee that all these approaches work for a particular dataset. That is the reason why one might need to go to data collection step if data is not enough, or to feature or model selection step if chosen features or model are not representative[3]. Moreover, these approaches have only studied on certain traffics, not including ever-increasing QUIC, TLS 1.3 traffic and other upcoming traffics.

G. Training and Validation

Training and validation step is similar to any other deep learning applications where a model's hyper-parameters are tuned to obtain the best accuracy. Typically, dataset is divided into three separate sets: train, validation and test set. The model is trained on the train set and the accuracy of validation set is observed to tune model's hyper-parameters. Finally, the unbiased accuracy is obtained by using the test set.[4].

[3] We haven't shown direct arrow from training and validation step to feature and model selection steps since one can simply skip the unnecessary steps.

[4]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.

H. Periodic Evaluation/Update

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

IV. Open Problems and Opportunities

In traffic classification, unencrypted traffic classification has been extensively studied and many commercial and free tools have been developed for. Classification of encrypted traffic 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 significantly larger number of classes common in operational network. There are also many unsolved problems in traffic classification that we will introduce in this section.

A. 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 TLS 1.3 draft version. However, most applications and websites have not adopted TLS 1.3 yet.

Previous studies on TLS 1.2 mainly used plain text fields during the handshake. But, by introduction of 0-RTT connectivity in TLS 1.3 and QUIC, only a few fields in the first packet remain unencrypted which is not clear if they suffice for classification.

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 II
OVERVIEW OF MODELS AND FEATURES OF TRAFFIC CLASSIFICATION

B. Multi-label Classification

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

C. Middle Flow Classification

Around 90% of the flows are short-lived ones. In certain applications, such as traffic engineering, one may want to focus on long flows. However, if the classification method relies on 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 classify the elephant flows 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 conducted to use 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. 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 problem rather than user action.

D. Zero-day Applications

Zero-day applications refer to the traffic classes that are new and their samples do not exist in the training set. It has been shown that in some cases zero-day applications can make up to 60% of flows and 30% of bytes in a network traffic [13]. Despite the importance, it is in a nascent stage and only a few recent studies [13] proposed solutions which usually rely on detecting unlabeled clusters and then labeling them. In the ML community, active learning, where a model selects which datapoints should be labeled, has been studied for many years. In a recent study on classifying images of characters [14], 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.

E. 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 similar task. Transfer learning and domain adaptation are the two widely used techniques in ML 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 distribution 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 on the target task needs significantly less labeled data and training time.

In classic ML tasks, such as image classification, transfer learning has shown to be highly effective. It is usually done in two ways: (a) One common strategy is to train a model on a similar but previously labeled dataset and then tune the model on the target task, hopefully with less labeled data and faster convergence. This approach has also been used in k-shot learning methods where only k labeled samples are available for the target task. (b) The other strategy is to train a model on an unlabeled dataset, by constructing the same input with a generative model, such as auto-encoder, and then retrain the model on the target task.

In the case of network traffic classification, a publicly available dataset can be used to pre-train a model which will be later tuned for another traffic classification task with fewer labeled samples. Our preliminary experiment shows a pre-trained model can achieve a good accuracy with only a few labeled data samples for retraining. In our experiment, we trained a model to predict statistical features of a flow based on sampled packets. In other words, we used the entire flow to obtain statistical features as a label, but only the sampled packets are fed to the model as an input. Then, we retrained the model with a few flows with application-specific labels. This method allows us to pre-train a model with a large and representative unlabeled dataset which is easier to obtain.

Unlike transfer learning where the source and target tasks (i.e. their class labels) are different, domain adaptation deals with the 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 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 in which one can re-train a model periodically based on domain adaptation techniques to capture new patterns for classes whose features are constantly changing, which are not uncommon in current Internet. Despite their usefulness, these strategies have not been extensively adopted for network classification task yet.

F. Multi-task Learning

This approach refers to any model in which more than one loss function is being 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 it reduces the risk of overfitting and helps the model find relevant features faster. This works when the input data is generated from a similar probability distribution or can be generated using a set of transformations from one another. As a result, it may be possible to use additional available datasets and define a single task for each if they are similar to your target task dataset. This can easily augment the dataset and improve the generalization. Many variations of multi-task learning have been used successfully for natural language processing and computer vision.

It is shown that even for single-task problems, adding some auxiliary task will improve generalization and performance. However, it has not been studied for network traffic classification task. There are potentially many ways to define auxiliary task without the need for additional labeling. For instance, assume a typical model which takes time series information of the first 20 packets as an input. One can define many auxiliary tasks that do not need human labeling, such as detecting TCP/UDP class, predicting the average packet length of the entire flow, detecting mice/elephant flow, etc. The efficacy of multi-task learning has not been studied for network traffic classification yet.

REFERENCES

[1] T. Stöber, M. Frank, J. Schmitt, and I. Martinovic, "Who do you sync you are?: smartphone fingerprinting via application behaviour," in *Proceedings of the sixth ACM conference on Security and privacy in wireless and mobile networks*. ACM, 2013, pp. 7–12.

[2] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Traffic classification of mobile apps through multi-classification," in *IEEE Global Communications Conference (GLOBECOM)*, 2017.

[3] S. E. Coull and K. P. Dyer, "Traffic analysis of encrypted messaging services: Apple imessage and beyond," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 5, pp. 5–11, 2014.

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

[5] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Mobile encrypted traffic classification using deep learning."

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

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

[8] W. Wang, Y. Sheng, J. Wang, X. Zeng, X. Ye, Y. Huang, and M. Zhu, "Hast-ids: learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection," *IEEE Access*, vol. 6, pp. 1792–1806, 2018.

[9] L. Vu, C. T. Bui, and Q. U. Nguyen, "A deep learning based method for handling imbalanced problem in network traffic classification," in *Proceedings of the Eighth International Symposium on Information and Communication Technology*. ACM, 2017, pp. 333–339.

[10] M. Lotfollahi, R. Shirali, M. J. Siavoshani, and M. Saberian, "Deep packet: A novel approach for encrypted traffic classification using deep learning," *arXiv preprint arXiv:1709.02656*, 2017.

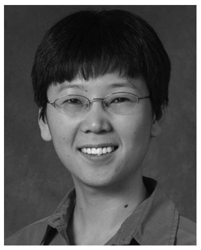
[11] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Network traffic classifier with convolutional and recurrent neural networks for internet of things," *IEEE Access*, vol. 5, pp. 18 042–18 050, 2017.

[12] J. Hochst, L. Baumgartner, M. Hollick, and B. Freisleben, "Unsupervised traffic flow classification using a neural autoencoder," in *2017 IEEE 42nd Conference on Local Computer Networks (LCN)*. IEEE, 2017, pp. 523–526.

[13] J. Zhang, X. Chen, Y. Xiang, W. Zhou, and J. Wu, "Robust network traffic classification," *IEEE/ACM Transactions on Networking (TON)*, vol. 23, no. 4, pp. 1257–1270, 2015.

[14] M. Woodward and C. Finn, "Active one-shot learning," *arXiv preprint arXiv:1702.06559*, 2017.

Shahbaz Rezaei received his B.S. degree in Computer Engineering from University of Science and Culture, Tehran, Iran, in 2011 and M.S. degree from the Sharif University of Technology, Tehran, Iran, in 2013. His research interests include mobile ad hoc networks, performance modeling, machine learning and deep reinforcement learning. He is currently a Ph.D. student at UC Davis working on solving computer network problems using deep learning approach.



Xin Liu received the Ph.D. degree in electrical engineering from Purdue University, West Lafayette, IN, USA, in 2002. She is currently a professor in the Computer Science Department, University of California, Davis, CA, USA. Before joining UC Davis, she was a postdoctoral research associate in the Coordinated Science Laboratory at UIUC. From March 2012 to July 2014, she was on leave from UC Davis and with Microsoft Research Asia. Her research interests are in the area of wireless communication networks, with a current focus on data-driven approach in networking. Dr. Liu received the Best Paper of year award of the Computer Networks Journal in 2003 for her work on opportunistic scheduling. She received the NSF CAREER award in 2005 for her research on Smart-Radio-Technology-Enabled Opportunistic Spectrum Utilization, and the Outstanding Engineering Junior Faculty Award from the College of Engineering, University of California, Davis, in 2005. She became a Chancellor's Fellow in 2011.