Autonomous Unknown-Application Filtering and Labeling for DL-based Traffic Classifier Update

Jielun Zhang, Fuhao Li, Feng Ye, Hongyu Wu
Department of Electrical and Computer Engineering
University of Dayton, Dayton, OH, USA
{zhangj46, lif003, fye001, wuh007}@udayton.edu

***Abstract*—Network traffic classification has been widely studied to fundamentally advance network measurement and management. Machine Learning is one of the effective approaches for network traffic classification. Specifically, Deep Learning (DL) has attracted much attention from the researchers due to its effectiveness even in encrypted network traffic without compromising neither user privacy nor network security. However, most of the existing models are created from closed-world datasets, thus they can only classify those existing classes previously sampled and labeled. In this case, unknown classes cannot be correctly classified. To tackle this issue, an autonomous learning framework is proposed to effectively update DL-based traffic classification models during active operations. The core of the proposed framework consists of a DL-based classifier, a self-learned discriminator, and an autonomous self-labeling model. The discriminator and self-labeling process can generate new dataset during active operations to support classifier update. Evaluation of the proposed framework is performed on an open dataset, i.e., ISCX VPN-nonVPN, and independently collected data packets. The results demonstrate that the proposed autonomous learning framework can filter packets from unknown classes and provide accurate labels. Thus, corresponding DL-based classification models can be updated successfully with the autonomously generated dataset.**

***Index Terms*—Traffic Classifier, Deep Learning, Application Filtering, Autonomous Update**

主旨

I. Introduction

Network packet classification is fundamental for advancing future networks, e.g., in differentiated Quality-of-Service, network intrusion detection (firewall access control), traffic shaping, resource allocation, etc. [1], [2]. However, the huge amount of users and heavy network traffic between multi-source network applications increase the uncertainty of network traffic [3]–[5]. Such uncertainty further challenges network measurement and management. To combat those issues, data packet classification methods are in need of an advancement. Traditional traffic classifiers that are based on port assignments or the clear text of packet payload become less effective due to dynamic port assignments and encrypted payloads by many applications [6]–[8]. Recently, Machine Learning (ML) algorithms [9], [10] and Deep Learning (DL) algorithms [11]–[13] have been applied to network packet classification. In particular, DL-based traffic classifiers, such as those built around Convolutional Neural Network (CNN), Multilayer Perceptron (MLP), Stacked Autoencoder (SAE), can provide high accuracy (around 95%) in packet classification even for encrypted applications [11]–[13]. Nonetheless, those DL-based traffic classifiers are created with a closed-world assumption, where a classifier is assumed to only identify the classes in a static dataset. In practice, we face an open-world issue, where the number of applications fluctuates and new types of packets are unknown to the classifiers. An existing classifier could be less accurate every time a new application goes on-line.

This work has been supported in part by the University of Dayton Office for Graduate Academic Affairs through the Graduate Student Summer Fellowship Program.

To tackle the issue, we propose an autonomous model updating framework to seamlessly update DL-based packet classifiers during active operations. Specially, the proposed framework is capable of (i) filtering packets of unknown applications from active network traffic, (ii) clustering the packets of unknown classes into corresponding discovered classes and assigning labels, (iii) building a new training dataset including both the existing classes and the discovered classes, and (iv) updating the current classifier through transfer learning. Our major contributions in this paper can be concluded as follows:

- An autonomous model updating framework is proposed to update DL-based traffic classifiers by generating a new dataset through packet filtering and labeling from unknown classes.
- Four distinct DL-based traffic classifiers are implemented to perform accurate traffic classification.
- A dataset is built by capturing numerous packets of several popular Internet applications for evaluation and verification of the proposed scheme.
- The proposed framework is evaluated in three distinct scenarios where all the built classifiers are used individually.

内容

The rest of the paper is organized as follows. Section II summarizes the related work. Section III formulates the problem and introduces the preliminaries. Section IV illustrates the proposed framework. Section V demonstrates the evaluation result. Section VI concludes the work.

II. Related Work

A. Traditional Traffic Classification

Traditional network traffic classification has been widely studied, such as port-based and payload-based traffic classification approaches [6]–[8]. Port-based approach uses the port

传统算法的弊端

assignment of a packet in its TCP/UDP header to match the default port number on the file released by the Internet Assigned Numbers Authority (IANA) [6]. Payload-based method, e.g., deep packet inspection [7], inspects the header and the payload for comparing the signatures on the application level. Nevertheless, they fail to perform traffic classification due to port translation (i.e., Network Address Port Translation [8]) and encryption of network packets [7].

B. Machine Learning based Traffic Classification

To tackle the issues with the traditional methods, researchers applied both unsupervised ML algorithms (i.e., K-Means, k-nearest neighbors) and supervised ones (e.g., logistic regression, support vector machine) [14]–[17] to build traffic classifiers, which use packet features such as packet size, inter-arrival time, etc., for classification. Anderson *et al.* in [15] proposed to use logistic regression algorithm for learning flow-level features, header information and other features jointly to classify the encrypted traffic as either malware traffic or normal traffic. Saber *et al.* in [16] proposed to combine Principal Component Analysis (PCA) [18] along with the support vector machine for traffic classification relying on only time-based flow features. Their work achieved an average classification accuracy at 94% but required a relatively high overhead time to obtain the flow features. However, ML-based classifiers usually provide low classification accuracy and they require manual feature selection [17].

C. Deep Learning based Traffic Classification

DL-based traffic classifiers have been widely studied in many researching fields such as computer vision, natural language processing, etc. [19]–[22]. Nonetheless, not until recently did researchers start to build neural networks (e.g., MLP, SAE, CNN) for traffic classification. Li *et al.* in [21] proposed a byte segment neural network including an attention that extracts features of payload segments individually and outputs classification results with a Softmax-classification layer. Moreover, several end-to-end DL-based classification models are proposed [12], [13], [22], [23]. Liu *et al.* in [22] developed FS-Net based on recurrent neural networks and an autoencoder for both traffic classification and packet feature mining. Lotfollahi *et al.* in [13] and Wang *et al.* in [12] applied MLP, SAE, and CNN for traffic classification, where the average classification accuracy is above 95%.

DL算法存在的不足

Apparently, the previous network traffic classification problems were mostly defined based on a closed-world assumption, such that they can only classify packets in a static dataset. In an open-world assumption, these DL-based classifiers need to be updated promptly to provide accurate traffic classification in order to support the network measurement and management.

III. Problem Formulation

A. Open-world Packet Classification

Let $\mathcal{D} = \{(p_1, l_1), (p_2, l_2), ..., (p_n, l_n)\}$ be a set of labeled training dataset that includes a total of M existing application classes. Define p_i as the i-th instance of application labeled with $l_i \in \mathcal{C} = \{c_1, c_2, ..., c_M\}$, and c_j as the corresponding class j. A DL-based classification model $F(p_n) \rightarrow \hat{l}_n$ is created from dataset $\mathcal{D}$ to predict label $\hat{l}_i$ of p_i that matches the actual label l_i.

Assume that the current DL-based traffic classifier F^t at time t has been created for M^t existing classes of applications i.e., $\mathcal{C}^t = \{c_1, c_2, ..., c_{M^t}\}$, using the training dataset $\mathcal{D}^t$. Besides existing classes, we define $\mathcal{U}^t$ as the set of unknown classes that exist in active traffic at time t. Thus, the collection of all possible classes of application is $\Omega = \{\mathcal{U}^t \cup \mathcal{C}^t\}$ in the open-world assumption. The classifier is to be updated at a future time $(t+1)$ as F^{t+1} that can discover previously unknown classes of applications $\mathcal{N}^{t+1}$ (defined as *discovered classes*) together with the existing classes. A new dataset $\mathcal{D}^{t+1}$ that comprises both $\mathcal{C}^t$ and $\mathcal{N}^{t+1}$ is thus required to update classifier F^{t+1}, such that $F^{t+1}(p_i) \rightarrow \hat{l}_i$, and $\hat{l}_i$ matches the actual label l_i for all $l_i \in \mathcal{C}^{t+1}$. To be specific, three subproblems are formulated to update a DL-based traffic classifier in an open-world classification scenario so that the updated F^{t+1} can correctly identify all M^{t+1} classes of packets in the current active network. The three subproblems are as follows:

1) Filtering unknown packets $\mathcal{N}^{t+1}$. 过滤、识别、更新
2) Identify the number of discovered classes.
3) Update the DL-based traffic classifier from F^t to F^{t+1}.

B. Preliminaries (Convolutional Neural Network)

Convolutional Neural Network (CNN) is a popular DL architecture used for image classification [24]. A typical CNN consists of convolutional layers, pooling layers, dense/fully-connected layers and a softmax layer. The convolutional layers perform the feature extraction by convolution kernels. For a data sample I in the format of 2-D matrix, it is processed by convolutional kernels in each convolutional layer as follows:

$$\begin{aligned} c[i][j][k] = & q[k] + \sum_{l} \sum_{s=1}^{W} \sum_{t=1}^{H} w[s][t][l][k] \\ & * I[(i-1)+s][(j-1)+t][l], \end{aligned} \tag{1}$$

where '$*$' is the convolution operator, k is the order of convolution kennels; l is the channel number of the input; W and H are the width and length of the convolution kernel; w and q are the weights and bias in the corresponding channel. Note that the stride in the illustrated example is set to 1.

The output of a convolutional layer is activated by Rectifier Linear Units (ReLU) for non-linearity. It provides a faster training process and help to avoid gradient vanishing problem compared with other activation functions (i.e., sigmoid and tanh functions) [25]. The activation process produced by ReLU is formulated as:

$$x[i][j][k] = \max\left(0, c[i][j][k]\right). \tag{2}$$

Pooling layers are usually attached to the output of activation functions for dimension reduction, which can speed up the training process. Nonetheless, all the raw packets used in the proposed scheme are relatively small, thus we remove the

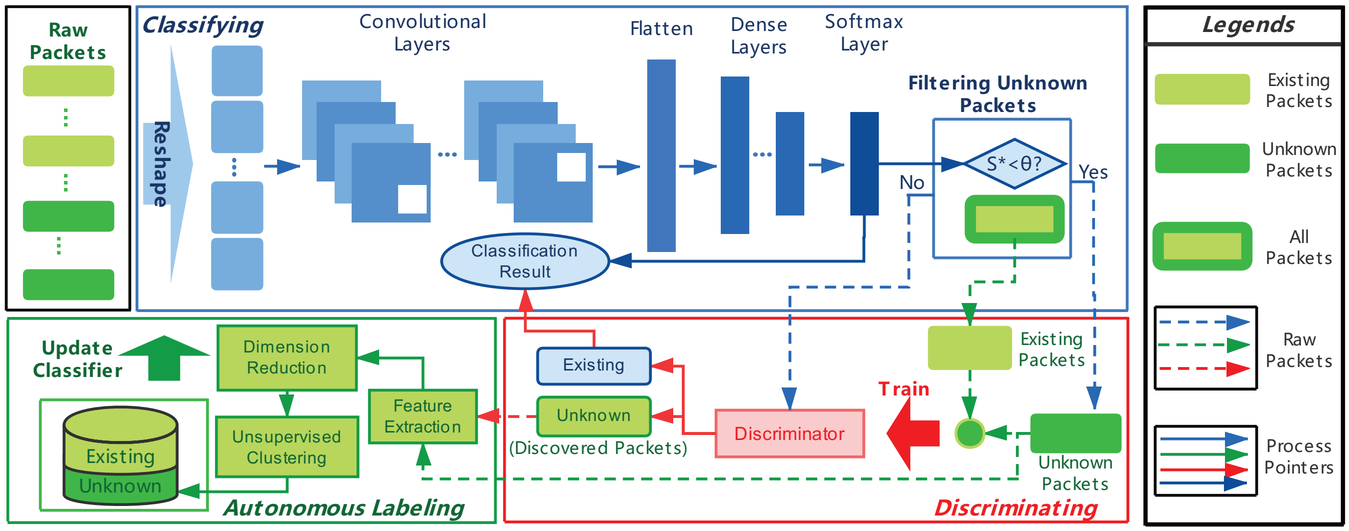

Fig. 1. Overview of the proposed autonomous model updating framework.

pooling layers in the proposed DL-based traffic classifiers in order to keep all details in the raw packets. The outputs of the last convolutional layer are flattened and passed to fully connected layers, which are also known as dense layers, where feature maps are produced. Softmax function is widely used at the end of neural networks to map non-normalized output to a probability distribution over classes of prediction [24]. A Softmax layer accepts the output from the last fully-connected layer and provides the final classification result. The fully-connected layer and the Softmax layer are computed as follows:

$$\mathbf{y} = [y_1, y_2, ..., y_N] = (W^T \cdot \mathbf{v}) + \mathbf{b}, \tag{3}$$

$$\mathbf{s} = [s_1, s_2, .., s_N] = \frac{\exp(y_n)}{\sum_{i=1}^{N} \exp(y_i)}, \tag{4}$$

where $\mathbf{v}$ is the output of the former dense layer; $\mathbf{y}$ is the output vector of the last dense layer that is connected to the Softmax layer; s_n is the categorical probability for the input to be classified into class n, where $s_n \leq 1$ and $\sum s_n = 1$.

IV. Autonomous Model Updating Framework

The proposed autonomous model updating framework consists of three stages, including DL-based packet classification, self-learned unknown application discrimination, and autonomous unknown packet self-labeling, as illustrated in Fig. 1. The classifier is to be deployed at a network gateway for identifying each incoming data packet. It also provides confidence scores (to be detailed in Section IV-A) that are further used in the discriminator. The discriminator is proposed to discriminate the data packets from unknown applications autonomously. Such a process can be triggered either manually or automatically after a period. The application classes of those filtered packets from the discriminator are denoted as *discovered classes*. The discovered classes are further clustered and self-labeled as the new generated dataset. The classification model is eventually updated based on the autonomously

以上图形构架

generated dataset. The details of each processing stage in the proposed framework are presented as follows.

A. Deep Learning based Traffic Classifier

Without loss of generality, we assume that there is a DL-based traffic classifier developed based on neural network models, e.g., CNN, MLP, RNN. The classifier first extracts features from an input data packet, and then performs classification by computing the categorical probabilities of the existing classes. The classifier is able to accurately identity the existing classes (e.g. M^t classes) based on the training dataset $\mathcal{D}^t$. In an open-world scenario, assume that there are packets from multiple applications of unknown classes in an active network traffic during discrimination. The packets of these unknown application classes are denoted as $\mathbf{p}_u$. Apparently, $\mathbf{p}_u$ cannot be properly classified by the original classifier.

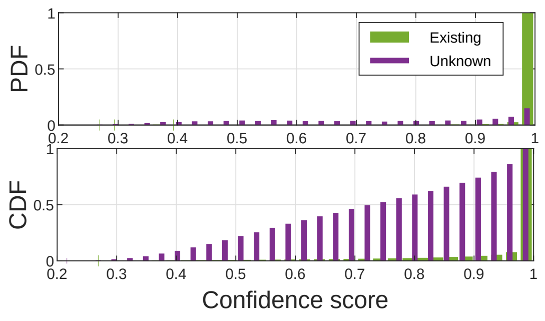

Fig. 2. Evaluation result on the classification model.

Let $\mathbf{s}$ be the output of the Softmax layer, and let $s^\star = \max(\mathbf{s})$ be the confidence score obtained every time a single packet is processed by the classifier, and $s_c^\star$ and $s_u^\star$ be the confidence score computed for the packets that belongs to the existing (also known as current) classes and unknown classes correspondingly. Let $\mathbf{P}$ be a group of packets (including $\mathbf{p}_c$ and $\mathbf{p}_u$) passed through the model during discrimination, and we define $\mathbf{S}_c$ and $\mathbf{S}_u$ as the confidence score sets that contain all the corresponding $s_c^\star$ and $s_u^\star$ computed from the packets in

$\mathbf{P}$. Examples of the probability density function (pdf) and the Cumulative Distribution Function (CDF) of a pair of $\mathbf{S}_c$ and $\mathbf{S}_u$ are illustrated in Fig. 2. By inspecting the pdf and CDF from extensive experiments, we discovered that most of $s_c^\star$ are distributed closely to 1. On the contrary, $s_u^\star$ is distributed uniformly since the classifier has been trained only by the samples of existing classes. Such a characterization of the confidence score sets $\mathbf{S}_c$ and $\mathbf{S}_u$ can be applied to filter partial packet samples of unknown classes directly. The pre-filtered packet samples are used to build a set of training data for the discriminator to further cluster and label the unknown classes.

B. Self-learning Discriminator

The discriminator is designed as a DL-based binary classifier to distinguish $\mathbf{p}_c$ and $\mathbf{p}_u$. Let ϵ be the classification accuracy of the classifier, and define θ as the boundary where CDF of $\mathbf{S}_c$ reaches $(1-\epsilon)$. For a packet whose $s^\star < \theta$, it will be treated as one the *discovered classes* (formerly unknown classes). Let $\mathbf{P_n}^L$ be a collection of these packets, where L remarks that $\mathbf{P_n}^L$ is a part of the entire set $\mathbf{P_n}$ of all discovered classes whose confidence scores locates on the left side of the boundary.

Algorithm 1: Filtering packets of unknown classes

```
Input  : $\mathbf{P}, \mathcal{D}^t$
Output: $\mathbf{P_n}$
1  $\mathbf{P_n^L}$, $\mathbf{P_n^{R'}}$, $\mathbf{P^R} \leftarrow \emptyset$    //Initialization
2  forall $\mathbf{p}_i \in \mathbf{P}$ do
3      Calculate $\mathbf{s}_i$ based on Eq. (4)
4      Calculate $s_i^\star = \max(\mathbf{s}_i)$
5      if $s_i^\star \leq \theta$ then
6          $\mathbf{P_n^L} \leftarrow \mathbf{P_n^L} || p_i$
7      else
8          $\mathbf{P^R} \leftarrow \mathbf{P^R} || p_i$
9      end
10 end
11 Train discriminator $D(p_i)$ by $\{\{\mathcal{D}^t,\mathbf{0}\},\{\mathbf{P_n^L},\mathbf{1}\}\}$
12 forall $p_i \in \mathbf{P^R}$ do
13     if $D(p_i) == 1$ then
14         $\mathbf{P_n^R} \leftarrow \mathbf{P_n^R} || p_i$
15     end
16 end
17 $\mathbf{P_n} \leftarrow \{\mathbf{P_n^L}, \mathbf{P_n^R}\}$
```

算法解释

Once both training samples of $\mathbf{p}_c$ (stored in the database) and the set $\mathbf{P_n}$ (filtered by the boundary θ) are obtained, the discriminator can be trained and classify the remaining packets in set $\mathbf{P^R}$ whose confidence scores are above the boundary. The packets identified as ones in the unknown classes will be inserted to the set $\mathbf{P_n}^R$. $\mathbf{P_n}^L$ and $\mathbf{P_n}^R$ are merged as the entire set $\mathbf{P_n}$ of all possible discovered classes. Details of the discriminating process are summarized in Alg. 1.

C. Autonomous Unknown Packet Self-Labeling

Although packets of the discovered classes can be filtered by the discriminator, the actual labels of these packets still need to be assigned in order to build a new training set for model update. The proposed an autonomous label assigner in an open-world traffic classification scenario performs autonomous labeling in two steps. Firstly, it extracts the feature of packets from the discovered classes in a low dimension. Then, it clusters the extracted features into different groups and label them accordingly. Note that the actual application remains unknown unless the network provider and/or end user are willing to share the information.

1) Feature Dimension Reduction: Feature maps are obtained from a dense layer by passing filtered packets in $\mathbf{P_n}$ through the classifier. They contain the hidden correlations between packets of the unknown classes and the learned packets used to train the classifier previously. Such hidden correlations can thus be used for clustering as the prior knowledge.

特征提取来降维

Based on the design of classification models, the adoption of feature extraction schemes may be considered to reduce the dimension of the feature for efficient clustering. PCA is adopted in this work due to its computational efficiency. Other dimension reduction schemes can also be used, e.g., Stacked Autoencoder and Reconstruction Independent Component Analysis. Let $\mathbb{Y} \in \mathbb{R}^{W\times V}$ be the combination of the feature maps obtained from V packets in $\mathbf{P_n}$, such that:

$$\mathbb{Y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_V] = \begin{bmatrix} y_{1,1} & y_{1,2} & \cdots & y_{1,V} \\ y_{2,1} & y_{2,2} & \cdots & y_{2,V} \\ \vdots & \vdots & \ddots & \vdots \\ y_{W,1} & y_{W,2} & \cdots & y_{W,V} \end{bmatrix}, \quad (5)$$

where $\mathbf{y}_v$ is the feature vector of v-th packet in $\mathbf{P_n}$ that contains a total of W feature values, and W is the number of neurons in the dense layer. Mapminmax Normalization is performed as follows to normalize $y_{w,v}$, s.t.,

$$y_{w,v} = \frac{y_{w,v} - \min(\mathbf{y_v})}{\max(\mathbf{y_v}) - \min(\mathbf{y_v})}. \quad (6)$$

The result of the normalization would be 0 if $\mathbf{y_v}$ is a zero vector. The covariance matrix is defined as follows:

$$G = \frac{1}{W-1}\sum_{i=1}^{W} (\mathbf{y_v} - \mu_v)(\mathbf{y_v} - \mu_v)^{\mathrm{T}}, \quad (7)$$

where u_v is the mean of $\mathbf{y_v}$. We then compute the eigenvector $U = [U_1, U_2, \ldots, U_W]$ of G s.t. $(\lambda I - G)U = 0$, where $\lambda = [\lambda_1, \ldots, \lambda_W]$ are eigenvalues. Note that λ_W are rearranged in descending order, i.e., $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_W$. Let H denote the extracted principal components as follows:

$$H = U^{\mathrm{T}}\mathbb{Y}. \quad (8)$$

The first q columns of H can be chosen as the representation of the principal information for the feature set $\mathbb{Y}$, where q defines the dimension of the extracted principal components.

2) Autonomous Clustering: The extracted features are then clustered into a few groups that represent high similarity among the other packets in the same group. For simplicity, K-mean [26] is applied in this work for demonstration. In theory, other clustering algorithms can be applied in the process. Assume that the centroids (centers of the clusters) of the i-th discovered class $\mathbf{N}_i$ after the clustering is marked as $\mathbf{O}^i$, which is calculated as follows:

$$\mathbf{O}^i = \frac{1}{|\mathbf{N}_i|} \cdot \sum_{\mathbf{o}_j \in \mathbf{N}_i} \mathbf{o}_j, \tag{9}$$

where $|\mathbf{N}_i|$ is the total number of samples in class $\mathbf{N}_i$. The similarity between principal components of the network traffic from two applications can be measured by their Euclidean Distance, calculated as follows:

$$d(\mathbf{o}_a, \mathbf{o}_b) = \sqrt{(\mathbf{o}_a - \mathbf{o}_b)^2}. \tag{10}$$

A smaller d represents a closer relationship, in contrast, a larger d value represents a lower similarity between them. The converged centroids will be used to cluster packets in $\mathbf{P_n}$. The objective function for the clustering model is defined as follows:

$$\min \sum_{\mathbf{o}_j \in \mathbf{N}_i} |\mathbf{o}_j - \mathbf{O}^i|^2. \tag{11}$$

For autonomous clustering, Bayesian information criterion (BIC) [27] is applied to find the optimal number of group, denoted as K. The BIC in our proposed clustering problem is calculated as follows:

$$\text{BIC} = V \cdot \ln\left(\frac{R}{V}\right) + k \cdot \ln(V), \tag{12}$$

$$R = \sum_{i=1}^{k} \sum_{\mathbf{o}_j \in \mathbf{N}_i} \sqrt{(\mathbf{o}_j - \mathbf{O}^i)^2}, \tag{13}$$

聚类算法

where V is the number of samples in $\mathbf{P_n}$ to be clustered; k is the index of clusters for enumeration; and R is the sum of root squared errors. Let $K_{\max}$ be the upper bound of the groups, such that $K_{\max} \leq V$. $K_{\max}$ can be defined based on the network environment. In each test of index k, a BIC value is calculated for clustering model evaluation. After finishing all enumeration of k from 1 to $K_{\max}$, the clustering model provides the most BIC decreasing is considered as the optimal one and the number of clusters included in it is the optimal cluster number. The overall autonomous clustering algorithm is summarized in Alg. 2, where $\mathbf{o}$ is the set of all extracted features; $\mathcal{M}$ is the clustering model with converged cluster centroids; $\mathcal{M}^*$ is the optimal clustering model; $l_k \in \mathbf{l_n}$ are the new assigned labels to the packets of discovered classes in $\mathbf{P_n}$.

D. New Dataset Generation and Model Update

The new dataset $\mathcal{D}^{t+1}$ is composed of both the old dataset $\mathcal{D}^t$ and the newly discovered ones (denoted as $\{\mathbf{P_n}, \mathbf{l_n}\}$), such that $\mathcal{D}^{t+1} \leftarrow \mathcal{D}^t \parallel \{\mathbf{P_n}, \mathbf{l_n}\}$. To update the classifier from current state F^t to F^{t+1}, transfer learning scheme is adopted since it migrates parameters of the current model to boost the training process. The detailed transfer learning process is presented in Alg. 3. 循环后参数更新

Algorithm 2: Autonomous application clustering

```
Data: $\mathbf{o}$, $K_{\max}$
Result: $\mathcal{M}^*$,$\mathbf{l_n}$
1  initialization;
2  For $k = 1 : K_{\max}$
3  Randomly choose $k$ objects from $\mathbf{O}$ as the initial centers
     of clusters of the new classes;
4  Repeat:
5  1) Assign or reassign each $\mathbf{o}_i$ to the cluster to which the
     $\mathbf{o}_i$ is the most similar, based on the mean value of all
     $\mathbf{o}_i$ in the cluster;
6  2) Update the cluster centroids by calculating the mean
     value of the $\mathbf{o}_i$ for each cluster;
7  Until Cluster centroids convergence, or reach the
     assigned maximum iteration time;
8  Save the current model $\mathcal{M}_k$;
9  Compute BIC$_k$ based on Eq. (13), (12) for $\mathcal{M}_k$;
10 Calculate $\Delta$BIC $=$ BIC$_k$ $-$ BIC$_{k-1}$ for $k > 1$;
11 EndFor
12 Find $\mathcal{M}_K$ obtains max($\Delta$BIC);
13 $\mathcal{M}^* \leftarrow \mathcal{M}_K$;
14 Assign label $l_k$ to all packets clustered in the $k$-th
     unknown class based on $\mathcal{M}^*$;
15 Store $l_k$ in the label set $\mathbf{l_n}$ with respect to the packets in
     $\mathbf{P_n}$;
```

最优聚类即为质心分布?

Algorithm 3: Classifier Update

```
Input  : $F^t$, $\mathcal{D}^{t+1}$
Output: $F^{t+1}$
1 while the model updating interval lapses do
2     Load new dataset $\mathcal{D}^{t+1}$ in the database
3     Load layers and weights in $F^t$
4     Resize the last dense layer from $M^t$ to $M^{t+1}$
5     Resize Softmax layer from length $M^t$ to $M^{t+1}$
6     Save the modified layers for $F^{t+1}$
7     Train $F^{t+1}$ with the new dataset $\mathcal{D}^{t+1}$
8 end
```

Mt为应用程序类？

V. Evaluation

A. Dataset for Evaluation

Part of the evaulation dataset is selected from “ISCX VPN-nonVPN dataset” (ISCXVPN2016) [28]. A total of $206,688$ packets, including Skype, Youtube, Vimeo, etc. [12], [28] are extracted from the dataset. Those applications are encrypted by different security protocols, e.g., HTTPS, SSL, SSH, etc. To further evaluate the proposed model updating scheme, we also collected a dataset from real-life network applications with encrypted packets. To ensure rich diversity and quantity, we capture a total of $492,721$ packets from 5 distinct applications, including Google Map, Speedtest by Ookla, Tencent QQ, Discord and DOTA2. Details of the dataset is summarized in Table I. Note that the first 24 bytes of each packet are removed to focus on encrypted payload only.

TABLE I
SUMMARY OF THE DATASET USED FOR EVALUATION.

Application	Total # samples	Application	Total # samples
*Google Map**	54, 114	Netflix	51, 932
*Speedtest (upload)**	112, 354	SCP (download)	15, 390
*Speedtest (download)**	39, 302	SFTP (download)	4, 729
*Discord**	20, 032	Skype file	4, 607
*Tencent QQ (voice)**	143, 370	TorTwitter	14, 654
*DOTA2**	123, 549	Vimeo	18, 755
Email clients	4, 417	VOIPbuster	35, 469
Facebook chat	5, 527	Youtube	12, 738

Note: Applications marked with $*$ in italics are new source applications for packet collection. The others are sampled from ISCX VPN-nonVPN dataset.

B. Design of Classification Model

The input of our proposed 1-D classifier is a packet vector with a dimension of 1×1456 bytes. In our designed 2-D classifier, packet vectors are reshaped to 39×39 and can be visualized as gray images (see examples on the left-hand side of Fig. 3). In our designed 3-D classifier, packet vectors are converted in to a 3D tensor with a size of $22 \times 22 \times 3$, which can be visualized as 24-bit RGB images (see examples on the right-hand side of Fig. 3). Detailed specifications of converted packets and the built classification models are summarized in the Table II.

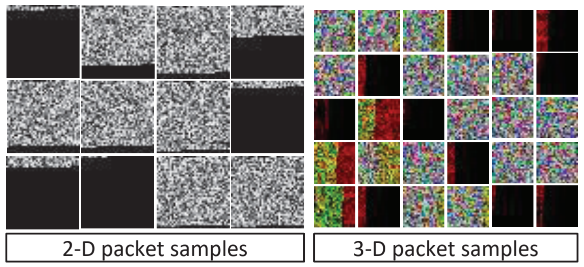

Fig. 3. Examples of input packets to 2-D CNN and 3-D CNN classifiers.

TABLE II
SPECIFICATIONS OF THE PROPOSED DEEP LEARNING BASED NETWORK TRAFFIC CLASSIFICATION MODELS.

Classifier type	*MLP*	*1-D CNN*	*2-D CNN*	*3-D CNN*
Input size	1×1456	1×1456	39×39	22×22×3
Input length	1456	1456	1521	1452
Convolutional kernel size	-	(1×9)×16 (1×9)×16	(3×3)×64 (3×3)×32	(3×3×3)×64 (3×3×3)×32
Activation function	ReLU	ReLU	ReLU	ReLU
Sizes of dense layers	768 128 9	128 9	128 9	128 9
Softmax layer	Yes	Yes	Yes	Yes

C. Experiment Settings

1) Experiment Environment: The evaluation and simulation of the proposed schemes are conducted on a workstation with an Intel® Xeon® CPU E5-2630 v3 @ 2.40GHz, 32.0 GB RAM @ 2133 MHz, a 480 GB SSD and an NIVIDIA GeForce GTX 1080 Ti. Matlab 2019a running in Windows 10 Enterprise

评估方法

is used for the scheme implementation. We also use another graphic processing unit, i.e., a docked NIVIDIA GeForce GTX 1080 connected to an ultrabook through Thunderbolt 3, to evaluate the processing speed of the classifiers.

2) Evaluation Metrics: Recall, Precision and F measurement score are applied to evaluate both the proposed clustering scheme and the updated classifiers. We define a true positive (TP) decision assigns two similar packets to the same cluster, a true negative (TN) decision assigns two dissimilar packets to different clusters. A classifier may have two types of erroneous outputs. One is the false positive (FP) decision, which assigns two dissimilar packets to the same cluster. The other one is false negative (FN) decision, which assigns two similar packets to different clusters. The evaluation metrics are formulated as follows:

$$R = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}, \quad P = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}, \quad F_\beta = \frac{(\beta^2+1)PR}{\beta^2 P + R}, \quad (14)$$

where $\beta > 1$ can be used as the penalty factor to provide more weight to recall, and we choose $\beta = 1$ in this paper.

To evaluate the clustering performance, let TP be the number of true positive instances properly classified as **X**; TN be the number of true negative instances properly classified as **not X**; FP be the number of false positive instances classified as **X** incorrectly; and FN be the number of false negative instances classified as **not X** incorrectly. The Rand Index (RI) [29] is used, which is defined as follows:

$$\mathrm{RI} = \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{FP}+\mathrm{FN}+\mathrm{TN}}. \quad (15)$$

The RI provides equal weight to FP and FN instances. To penalize FN instances over FP instances for focusing on clustering similar packets to the same cluster as much as possible, F measure can be applied instead.

D. Evaluation Results

To demonstrate the effectiveness and robustness of our proposed autonomous classifier updating scheme, evaluations are conducted in three distinct scenarios, as detailed in Table III.

TABLE III
EVALUATION SCENARIOS.

	Scenario		
	A	**B**	**C**
Existing classes	Email clients, Youtube, Vimeo, Skype file, SFTP (down), TorTwitter, Facebook chat, VOIPbuster, SCP (down).	Skype file, Facebook chat, VOIPbuster, Youtube, DOTA2*, Email clients, Vimeo, SFTP (down), STest (up)*.	Youtube, Facebook chat, Email clients, Skype file, Vimeo, SFTP (down), TorTwitter, VOIPbuster, SCP (down).
Unknown classes	Discord*, Google Map*.	TorTwitter, SCP (down).	Netflix, STest (down)*, QQ (voice)*.

Note: packets of the application marked with $*$ are selected from our captured dataset; STest stands for Speedtest; ‘up’ and ‘down’ represent upload and download respectively.

TABLE IV
PERFORMANCE EVALUATION RESULTS OF THE CLASSIFIERS.

		Recall			**Precision**			**F1 score**			**Speed (packets/ms)**			**Bandwidth (Mbps)**		
		Max.	Avg.	Min.	Max.	Avg.	Min.	Max.	Avg.	Min.	GPU1	GPU2	CPU	GPU1	GPU2	CPU
Scenario A																
MLP	Original	95.0	94.5	94.1	95.0	94.5	94.1	95.0	94.5	94.1	19.7	21.3	12.4	229.3	236.8	144.6
	Updated	91.1	90.7	90.0	91.5	91.1	90.4	91.3	90.9	90.2	21.2	20.4	11.6	247.3	238.1	136.2
1D-CNN	Original	97.7	97.1	96.7	97.6	97.1	96.7	97.6	97.1	96.8	6.1	8.7	0.4	70.7	101.8	4.7
	Updated	94.5	93.9	93.2	95.2	94.7	94.2	94.8	94.3	93.7	5.9	8.5	0.4	68.9	99.9	4.6
2D-CNN	Original	97.7	97.0	96.6	97.7	97.1	96.6	97.7	97.0	96.6	7.3	10.3	0.4	89.2	125.6	5.3
	Updated	96.7	96.3	95.7	96.9	96.5	95.9	96.8	96.4	95.8	8.1	10.4	0.4	98.4	126.7	5.3
3D-CNN	Original	97.4	97.1	96.7	97.4	97.1	96.7	97.4	97.1	96.7	13.6	15.1	0.9	158.1	175.7	10.9
	Updated	96.5	96.0	95.6	96.6	96.1	95.6	96.5	96.1	95.5	14.5	15.6	0.9	168.1	181.4	10.9
Scenario B																
MLP	Original	93.2	92.3	91.7	93.2	92.3	91.7	93.2	92.3	91.6	20.3	21.3	12.5	236.6	249.1	145.2
	Updated	92.0	91.3	90.6	92.1	91.4	90.7	92.0	91.3	90.6	21.7	20.4	12.3	252.6	237.9	142.8
1D-CNN	Original	97.7	97.4	97.1	97.7	97.4	97.0	97.8	97.4	97.0	5.3	8.7	0.4	61.2	101.8	4.8
	Updated	96.5	96.0	95.6	96.5	96.1	95.7	96.5	96.0	95.7	6.5	8.7	0.4	75.4	101.3	4.7
2D-CNN	Original	98.2	97.9	97.4	98.2	97.9	97.4	98.2	97.9	97.4	8.7	10.2	0.4	105.9	125.1	5.3
	Updated	96.6	96.2	95.7	96.6	96.2	95.7	96.6	96.2	95.7	8.9	10.0	0.4	108.1	121.5	5.0
3D-CNN	Original	97.6	97.3	96.8	97.6	97.3	96.8	97.6	97.3	96.8	12.2	15.2	1.0	142.1	176.1	11.0
	Updated	96.4	96.0	95.6	96.5	96.0	95.6	96.4	96.0	95.6	14.1	15.3	0.9	163.9	178.8	10.5
Scenario C																
MLP	Original	92.4	91.8	91.1	92.4	91.8	91.1	92.4	91.8	91.1	20.8	20.6	12.5	242.1	240.1	146.1
	Updated	92.5	91.8	91.1	92.6	92.0	91.1	92.5	91.9	91.2	20.6	20.9	11.7	239.4	243.0	136.6
1D-CNN	Original	98.0	97.6	97.2	98.0	97.6	97.2	98.0	97.6	97.2	6.2	8.7	0.4	72.1	101.6	4.7
	Updated	97.8	97.5	97.0	97.8	97.5	97.0	97.8	97.5	97.0	6.3	8.6	0.4	72.9	100.0	4.5
2D-CNN	Original	98.2	97.9	97.5	98.2	97.9	97.6	98.2	97.9	97.5	7.7	10.3	0.4	93.5	125.9	5.3
	Updated	98.1	97.8	97.3	98.1	97.8	97.3	98.1	97.8	97.3	7.6	10.0	0.4	93.0	122.1	5.2
3D-CNN	Original	97.8	97.4	97.0	97.8	97.4	97.0	97.8	97.4	97.0	13.7	15.0	0.9	159.5	174.4	10.9
	Updated	97.4	97.1	96.7	97.4	97.1	96.7	97.4	97.1	96.7	13.2	14.9	0.9	153.8	173.4	10.8

In each scenario, we compose five data portions $\mathcal{P}_1$, $\mathcal{P}_2$, $\mathcal{P}_3$, $\mathcal{P}_4$, and $\mathcal{P}_5$ by selecting packets randomly in the combined dataset which including both ISCX dataset and the newly collected dataset. Data portion $\mathcal{P}_1$ is the training dataset that consists of $2,500$ packets randomly chosen from each existing application class. Data portion $\mathcal{P}_3$ is the validation dataset that consists of $1,200$ packets from each existing application. Data portions $\mathcal{P}_2$ and $\mathcal{P}_4$ consist of 500 packets from each existing application and $4,500$ packets from each unknown application. Data portion $\mathcal{P}_4$ represents the active unknown packets. It is used along with data portion $\mathcal{P}_2$ as well as partial data in portion $\mathcal{P}_1$ (500 packets from each class) to simulate the active network traffic. The combination of the data portions allow the discriminator to learn the relationship between existing classes and unknown classes autonomously. Portion $\mathcal{P}_5$ is comprised of $1,200$ packets from unknown classes. It is combined with portion $\mathcal{P}_4$ and portion $\mathcal{P}_3$ to test the updated classifier.

For better illustration, we choose to compare the updating performance of the proposed scheme among MLP, 1D-CNN, 2D-CNN and 3D-CNN based classifiers in all proposed scenarios. The summary of classification performance is given in Table IV. We use two different GPUs for the evaluation, namely, GPU1 (NVIDIA GeForce GTX 1080) and GPU2 (NVIDIA GeForce GTX 1080 Ti), as well as the CPU (Intel® Xeon® CPU E5-2630 v3 @ 2.40GHz) to evaluate the computational efficiency. 50 rounds of tests are conducted for both original and updated classifiers. 500 packets of each application are randomly chosen in each round from the testing dataset. It can be observed that all CNN based classifiers outperform the MLP one. Nonetheless, the MLP leads in computational efficiency, followed by 3D-CNN. The lightweight network structure of the MLP-based classifier allows it to support a higher bandwidth. It can be concluded from Table IV that the 3D-CNN based classifier has the best overall performance because of its high classification accuracy and computational efficiency.

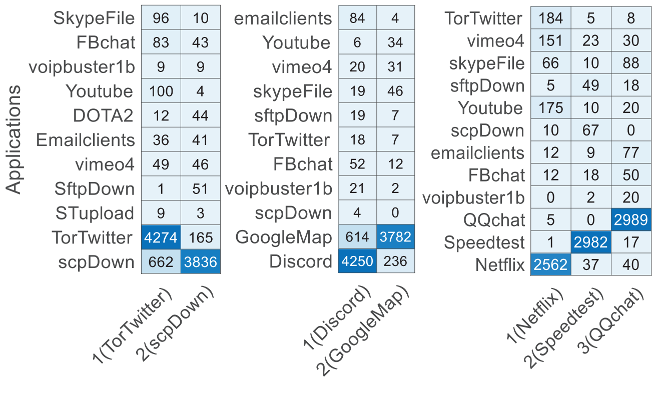

Fig. 4. Clustering results in the testing scenarios (3D-CNN).

TABLE V
AUTONOMOUS LABELING PERFORMANCE IN 3D-CNN CLASSIFIER.

3D-CNN	**Recall**	**Precision**	**F1 score**	**Rand Index**
Scenario A	83.1	75.6	79.2	79.9
Scenario B	83.7	72.0	77.4	78.7
Scenario C	94.4	83.4	88.5	92.8

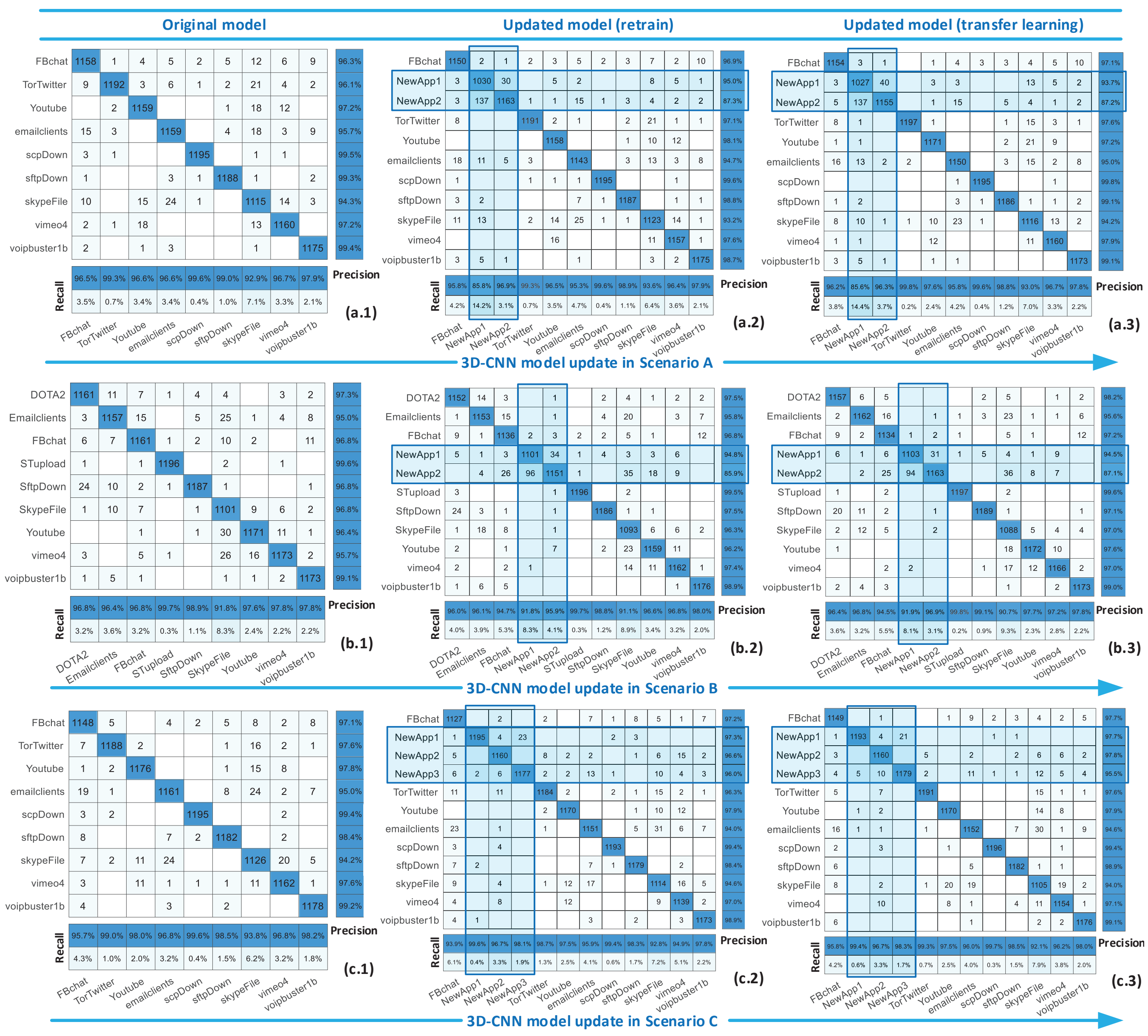
Fig. 5. Performance confusion matrices of original, updated (retain), and updated (transfer) 3-D CNN classifiers.

We further analyze the 3D-CNN based classifier in the aspects of 1) the clustering performance when assigning labels to the unknown classes, and 2) the classification performance of both original and updated classifiers in all proposed scenarios. In the autonomous clustering process of the model update for 3D-CNN classifiers, we performed feature extraction by choosing the first 7 most significant components in PCA to preserve 95% content of the original feature maps. K-means clustering algorithm is then applied to assign the labels for the corresponding unknown classes. The number of clusters is determined autonomously according to Alg. 2. The clustering results for 3D-CNN based classification model in all scenarios are presented below in Fig. 4. The calculated Recall, Precision and F score are given in Table V.

The clustering performance summarized in Fig. 4 and Table V indicates that the proposed scheme can cluster multiple unknown classes with the corresponding packets filtered by the discriminator.

To further evaluate the performance of 3D-CNN classifiers, we compose a testing dataset with data portion $\mathcal{P}_3$ (1200 packets from each existing application) and data portion $\mathcal{P}_5$ (1200 packets from each unknown application). Fig. 5 demonstrates the confusion matrices of classification results performed by the original classifier and the updated classifier with or without transfer learning. Each confusion matrix provides classification Recall and Precision at the bottom and to the right side respectively, which are accuracy metrics that indicate the classification performance. The diagonal of each confusion matrix presents the amount of packets of each application that are correctly classified. In all Scenarios, the observation shows that all classifiers have a high overall classification accuracy. However, due to the involvement of new classes, the accuracy of both updated classifiers for each existing class is slightly lower than the original classifier. In either Scenario A or Scenario B, almost all new classes are classified properly. Note that the accuracy of classifying Google Map is relatively lower than others in updated classifiers. It is because of the similarity of packets between Google Map and another unknown application appeared in Scenario A. Moreover, in Scenario C, the classification performance of new classes is superb, which reaches an average of 97%.

VI. Conclusion

Network traffic classification is the fundamental for accurate network measurement and efficient network management. To solidify those classifiers in an open-world assumption, we proposed an autonomous model updating framework that can filter the packets of unknown classes and cluster them to the corresponding classes. The filtered packets with their assigned labels as well as the packets of existing classes are combined to produce a new dataset to update the classifier. To evaluate the proposed framework, we used the packets captured in real life and the packets in an open dataset. Moreover, three scenarios were designed by mixing different classes in the dataset. The evaluation results demonstrated that our proposed autonomous model updating framework can update DL-based traffic classifiers with the capability of classification of the packets from unknown classes in active network.

Reference

[1] H. Doroud, G. Aceto, W. de Donato, E. A. Jarchlo, A. M. Lopez, C. D. Guerrero, and A. Pescape, "Speeding-up dpi traffic classification with chaining," in *2018 IEEE Global Communications Conference (GLOBECOM)*, Dec 2018, pp. 1–6.

[2] J. Van Lunteren and A. Engbersen, "Packet classification," Mar. 20 2007, uS Patent 7,193,997.

[3] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, 2018.

[4] P. Li, Z. Chen, L. T. Yang, J. Gao, Q. Zhang, and M. J. Deen, "An improved stacked auto-encoder for network traffic flow classification," *IEEE Network*, vol. 32, no. 6, pp. 22–27, November 2018.

[5] J. Zhang, F. Ye, and Y. Qian, "A distributed network qoe measurement framework for smart networks in smart cities," in *2018 IEEE International Smart Cities Conference (ISC2)*, Sep. 2018, pp. 1–7.

[6] A. W. Moore and K. Papagiannaki, "Toward the accurate identification of network applications," in *International Workshop on Passive and Active Network Measurement*. Springer, 2005, pp. 41–54.

[7] M. Finsterbusch, C. Richter, E. Rocha, J.-A. Muller, and K. Hanssgen, "A survey of payload-based traffic classification approaches," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 2, pp. 1135–1156, 2013.

[8] D. C. Sicker, P. Ohm, and D. Grunwald, "Legal issues surrounding monitoring during network research," in *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*. ACM, 2007, pp. 141–148.

[9] Z. Wang, Y. Dong, S. Mao, and X. Wang, "Internet multimedia traffic classification from qos perspective using semi-supervised dictionary learning models," *China Communications*, vol. 14, no. 10, pp. 202–218, Oct 2017.

[10] J. Kornycky, O. Abdul-Hameed, A. Kondoz, and B. C. Barber, "Radio frequency traffic classification over wlan," *IEEE/ACM Transactions on Networking*, vol. 25, no. 1, pp. 56–68, Feb 2017.

[11] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware traffic classification using convolutional neural network for representation learning," in *2017 International Conference on Information Networking (ICOIN)*. IEEE, 2017, pp. 712–717.

[12] P. Wang, F. Ye, X. Chen, and Y. Qian, "Datanet: Deep learning based encrypted network traffic classification in sdn home gateway," *IEEE Access*, vol. 6, pp. 55 380–55 391, 2018.

[13] M. Lotfollahi, M. Jafari Siavoshani, R. Shirali Hossein Zade, and M. Saberian, "Deep packet: a novel approach for encrypted traffic classification using deep learning," *Soft Computing*, May 2019. [Online]. Available: https://doi.org/10.1007/s00500-019-04030-2

[14] H. Shi, H. Li, D. Zhang, C. Cheng, and X. Cao, "An efficient feature generation approach based on deep learning and feature selection techniques for traffic classification," *Computer Networks*, vol. 132, pp. 81–98, 2018.

[15] B. Anderson, S. Paul, and D. McGrew, "Deciphering malwares use of tls (without decryption)," *Journal of Computer Virology and Hacking Techniques*, vol. 14, no. 3, pp. 195–211, 2018.

[16] A. Saber, B. Fergani, and M. Abbas, "Encrypted traffic classification: Combining over-and under-sampling through a pca-svm," in *2018 3rd International Conference on Pattern Analysis and Intelligent Systems (PAIS)*, Oct 2018, pp. 1–5.

[17] P. Wang, X. Chen, F. Ye, and Z. Sun, "A survey of techniques for mobile service encrypted traffic classification using deep learning," *IEEE Access*, vol. 7, pp. 54 024–54 033, 2019.

[18] S. Loisel and Y. Takane, "Comparisons among several methods for handling missing data in principal component analysis (pca)," *Advances in Data Analysis and Classification*, vol. 13, no. 2, pp. 495–518, 2019.

[19] I. Masi, F. Chang, J. Choi, S. Harel, J. Kim, K. Kim, J. Leksut, S. Rawls, Y. Wu, T. Hassner, W. AbdAlmageed, G. Medioni, L. Morency, P. Natarajan, and R. Nevatia, "Learning pose-aware models for pose-invariant face recognition in the wild," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 2, pp. 379–393, Feb 2019.

[20] S. Deena, M. Hasan, M. Doulaty, O. Saz, and T. Hain, "Recurrent neural network language model adaptation for multi-genre broadcast speech recognition and alignment," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 3, pp. 572–582, March 2019.

[21] R. Li, X. Xiao, S. Ni, H. Zheng, and S. Xia, "Byte segment neural network for network traffic classification," in *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*. IEEE, 2018, pp. 1–10.

[22] C. Liu, L. He, G. Xiong, Z. Cao, and Z. Li, "Fs-net: A flow sequence network for encrypted traffic classification," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 1171–1179.

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

[24] B. Hua, M. Tran, and S. Yeung, "Pointwise convolutional neural networks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, June 2018, pp. 984–993.

[25] G. Patrini, A. Rozza, A. K. Menon, R. Nock, and L. Qu, "Making deep neural networks robust to label noise: A loss correction approach," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017, pp. 2233–2241.

[26] V. Cohen-Addad, P. N. Klein, and C. Mathieu, "Local search yields approximation schemes for k-means and k-median in euclidean and minor-free metrics," *SIAM Journal on Computing*, vol. 48, no. 2, pp. 644–667, 2019.

[27] F. K. Teklehaymanot, M. Muma, and A. M. Zoubir, "Novel bayesian cluster enumeration criterion for cluster analysis with finite sample penalty term," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4274–4278.

[28] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of encrypted and vpn traffic using time-related," in *Proceedings of the 2nd international conference on information systems security and privacy (ICISSP)*, 2016, pp. 407–414.

[29] D. Steinley, "Properties of the hubert-arable adjusted rand index." *Psychological methods*, vol. 9, no. 3, p. 386, 2004.