University of Nebraska - Lincoln

DigitalCommons@University of Nebraska - Lincoln

Faculty Publications from the Department of Electrical and Computer Engineering

Electrical & Computer Engineering, Department of

10-19-2018

Datanet: Deep Learning Based Encrypted Network Traffic Classification in SDN Home Gateway

Pan Wang

Feng Ye

Xuejiao Chen

Yi Qian

Follow this and additional works at: https://digitalcommons.unl.edu/electricalengineeringfacpub

Part of the Computer Engineering Commons, and the Electrical and Computer Engineering Commons

This Article is brought to you for free and open access by the Electrical & Computer Engineering, Department of at DigitalCommons@University of Nebraska - Lincoln. It has been accepted for inclusion in Faculty Publications from the Department of Electrical and Computer Engineering by an authorized administrator of DigitalCommons@University of Nebraska - Lincoln.

IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Open Access CC-BY

Received September 10, 2018, accepted September 21, 2018, date of publication September 27, 2018, date of current version October 19, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2872430

Datanet: Deep Learning Based Encrypted Network Traffic Classification in SDN Home Gateway

PAN WANG[1], (Member, IEEE), FENG YE[2], (Member, IEEE), XUEJIAO CHEN[3], (Member, IEEE), AND YI QIAN[4], (Senior Member, IEEE)

[1]Department of Modern Posts, Nanjing University of Posts and Telecommunications, Nanjing 210003, China
[2]Department of Electrical and Computer Engineering, University of Dayton, Dayton, OH 45469, USA
[3]School of Communication, Nanjing College of Information Technology, Nanjing 210023, China
[4]Department of Electrical and Computer Engineering, University of Nebraska–Lincoln, Omaha, NE 68182, USA

Corresponding author: Feng Ye (fye001@udayton.edu)

This work was supported in part by the 2016 Jiangsu Provincial Government Scholarship Program, China, in part by the 2017 Jiangsu Overseas Visiting Scholar Program for University Prominent Young & Middle-Aged Teachers and Presidents, China, in part by the Top-Notch Academic Programs Project of Jiangsu Higher Education Institutions under Grant PPZY2015A092, in part by the National Natural Science Foundation of China under Grant 61373135, Grant 61672299, Grant 61702281, and Grant 61602259, and in part by the Natural Science Foundation of Jiangsu Province under Grant BK20150866 and Grant BK20160913.

ABSTRACT A smart home network will support various smart devices and applications, e.g., home automation devices, E-health devices, regular computing devices, and so on. Most devices in a smart home access the Internet through a home gateway (HGW). In this paper, we propose a software-defined-network (SDN)-HGW framework to better manage distributed smart home networks and support the SDN controller of the core network. The SDN controller enables efficient network quality-of-service management based on real-time traffic monitoring and resource allocation of the core network. However, it cannot provide network management in distributed smart homes. Our proposed SDN-HGW extends the control to the access network, i.e., a smart home network, for better end-to-end network management. Specifically, the proposed SDN-HGW can achieve distributed application awareness by classifying data traffic in a smart home network. Most existing traffic classification solutions, e.g., deep packet inspection, cannot provide real-time application awareness for encrypted data traffic. To tackle those issues, we develop encrypted data classifiers (denoted as DataNets) based on three deep learning schemes, i.e., multilayer perceptron, stacked autoencoder, and convolutional neural networks, using an open data set that has over 200 000 encrypted data samples from 15 applications. A data preprocessing scheme is proposed to process raw data packets and the tested data set so that DataNet can be created. The experimental results show that the developed DataNets can be applied to enable distributed application-aware SDN-HGW in future smart home networks.

INDEX TERMS Encrypted traffic classification, home gateway, distributed network management, deep learning, SDN.

I. INTRODUCTION

A smart home has a network that supports various types of smart devices, including home automation, health care and entertainment [1], [2]. Those smart devices are independently operated and managed by network operators, service providers and home users. Due to different network quality-of-service (QoS) requirements, it is challenging to have efficient end-to-end network management for smart home users [3], [4]. In this paper, we propose an application-aware software-defined network (SDN) inspired home gateway (HGW) framework (SDN-HGW) for distributed smart home networks.

SDN is an emerging and promising networking paradigm to dramatically simplify network management, improve network resource utilization, reduce operating costs and promote innovation and evolution [5], [6]. Current SDN controller can monitor and manage QoS in core networks [7], [8]. However, end-to-end network QoS may not be managed by the SDN controller due to different QoS requirements for applications and end users, as well as security and privacy issues.

2169-3536 © 2018 IEEE. Translations and content mining are permitted for academic research only.
Personal use is also permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

Moreover, the core network controller based solutions usually require relatively high computational resources [9], [10]. QoS management in a smart home network requires a light-weight and distributed scheme to achieve local real-time application awareness. To tackle this issue, our proposed SDN-HGW framework leverages the concept of SDN for its separated layers with advanced controls and flexibility in computing resources [11]–[16].

For user privacy, many applications have applied security protocols such as HTTPS, SSH, SSL etc [17], [18]. Most existing solutions to application awareness are based on Deep Packet Inspection (DPI) and Machine Learning (ML). However, those solutions cannot provide accurate real-time traffic classification encrypted network applications [19]–[22]. In this work, we design and develop DataNets (i.e. encrypted data packet classifiers) using three approaches: multilayer perceptron (MLP), stacked autoencoder (SAE) and convolutional neural network (CNN), based on a selected dataset from the "ISCX VPN-nonVPN" encrypted traffic dataset [23]. As a proof of concept, the selected dataset has more than 20, 000 data packets from 15 types of applications. The experimental results show that the developed DataNets can provide real-time and fine-grained application awareness to SDN-HGW in distributed home networks. The major contributions in this work are summarized as follows:

- An application-aware SDN-HGW framework is proposed for smart home networks.
- DataNets are developed using three deep learning based approaches for encrypted data classification based on an open dataset.
- Experiments are conducted to demonstrate the accuracy of the developed DataNets. Computational efficiency is also evaluated with practical HGW settings.

The remaining of the paper is organized as follows. Related work is discussed in Section II. The proposed framework for application-aware SDN-HGW is illustrated in Section III. The data preprocessing scheme is given in Section IV. The core development of DataNets is described in Section V. Evaluation and experimental results are given in Section VI. Finally, conclusion and future work are presented in Section VII.

II. RELATED WORK

SDN is a new network paradigm that enables network infrastructure virtualization by decoupling the control and data planes, creating a dynamic, flexible, automated and manageable architecture [1], [5]. SDN controller in the core network is the key component of the whole SDN architecture which mainly control SDN switches in order to manage the whole data flow [1], [24]. While the SDN controller enables efficient flow control, it cannot achieve distributed end-to-end QoS management due to no control of end users.

Traffic classification has been widely studied in the network management domain with three major approaches, port based [25]–[27], statistical approach [28], [29] and payload based [21], [22], [30], [31]. The port-based approach is one of the earliest for traffic classification. It uses the association of the ports in the TCP/UDP header with well-known TCP/UDP port numbers assigned by the IANA [27]. Port-based approach is simple and fast. Nevertheless, not all protocols can be classified by ports due to dynamic port assignment and tunnels and Network Address Port Translation (NAPT) [25], [26]. The statistical approach uses payload-independent parameters such as packet length, inter-arrival time, flow duration etc. for classification. For example, DPI is one of the mostly accepted techniques [21], [22], [31]. However, DPI techniques have limitations regarding encrypted payload encryption, user privacy and tunneling transfer. Besides some customized statistical methods, many researches were carried out using Machine Learning (ML) algorithms [29], [32], [33]. Although ML can alleviate some limitations of DPI, existing flow-based solutions cannot provide real-time, fine-grained traffic classification to support application-aware smart home networks.

Recently, researchers have tried to apply deep learning [34], [35] to traffic classification [36], [37]. Wang [36] proposed an SAE based method to identify unencrypted data traffic. However, the dataset was not open to the public and the work did not demonstrate its application for encrypted traffic. Lotfollahi *et al.* [37] introduced SAE and CNN based methods to classify encrypted traffic. However, the evaluation of computational performance was not provided.

In this work, we develop and evaluate DataNets using three deep learning based approaches, MLP, SAE and CNN, given the popularity of them in the community. Moreover, a selected dataset from an open source with all encrypted data traffic is used for DataNets development. The developed DataNet is the core to the SDN-HGW that can help to achieve distributed end-to-end network measurement and network management in the SDN paradigm. The success of this work will enable further development in networking, e.g. network resource management, new business plans, etc., without compromising security/privacy of service providers nor users.

III. OVERVIEW OF THE APPLICATION-AWARE SDN HOME GATEWAY FRAMEWORK

An overview of the proposed SDN-HGW framework is shown in Fig. 1. The proposed framework is to provide fine-grained and application-level traffic classification over encrypted traffic from a SDN-HGW. The framework consists of *smart home infrastructure*, *data plane*, *control plane* and *application plane*, described in the following.

The *smart home infrastructure* is composed of smart devices in a home network. Smart devices can be generally divided into three types: home automation, healthcare and entertainment [2]. All smart devices access the external network (e.g., the Internet) through the SDN-HGW.

The *data plane* is at the SDN-HGW to provide distributed network QoS management for the last hop to meet different requirements of applications. As shown in Fig. 2, the proposed SDN-HGW not only forwards packets through the

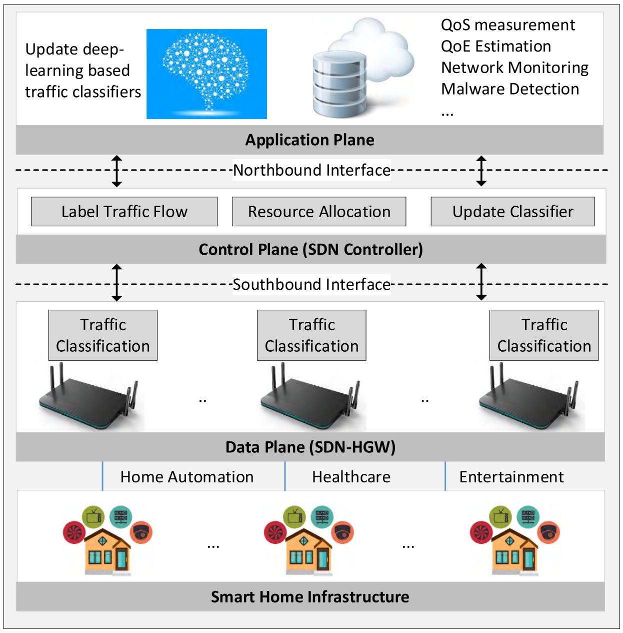


FIGURE 1. Overview of the SDN-HGW framework.

southbound interface, but also acts as a probe for traffic identification in smart home networks. First, the probe will receive the packets from the raw packet collector, which could be an eBPF program focusing on packet capturing from the eBPF Datapath [38]. Secondly, the probe uses a traffic classifier, which is defined as **DataNet** in the proposed framework. Different from existing SDN controller design, DataNet would require a local artificial intelligence (or neural network) co-processor that is embedded in the next-generation network gateway. The results will be tagged as an AppID for the SDN controller for further networking management. Note that the proposed SDN-HGW can collect edge traffic flows that may hardly be collected by the SDN controller or core network. For example, some packets such as ARP, DNS, DHCP, etc. may only be visible at the edge network gateway [24].

The *control plane* is supported by the SDN controller for three tasks. First, it labels each data flow with the probing results from each SDN-HGW. Second, the SDN controller periodically updates the DataNet and downloads the latest classifiers to each SDN-HGW. Third, the SDN controller manages network resources based on accurately labeled traffic flows in the core network.

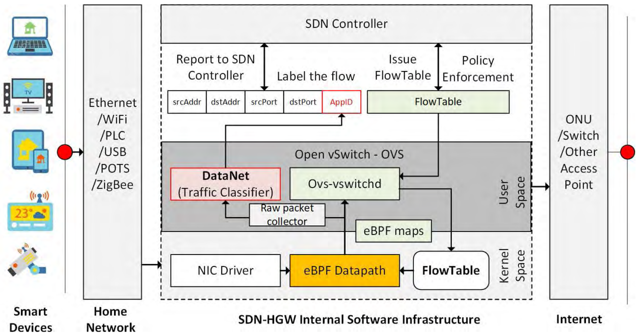


FIGURE 2. The internal software architecture of SDN-HGW.

The *application plane* supports applications through the northbound interface with the SDN controller. In particular, a deep learning based training platform is deployed in the application plane. The platform is responsible for creating and updating DataNets from collected data samples from various network applications. Note that the training platform has much more powerful computing resources compared to a SDN-HGW for fast and accurate DataNet development.

IV. DATA PACKET PRE-PROCESSING

DataNet is the core of the proposed SDN-HGW framework. However, a raw data packet captured is not in ideal form DataNet processing (and developing). For example, the data packet used in this work is in PCAP or PCAPNG format [39]. A raw packet includes information that are unnecessary for classification, e.g., total numbers of TCP/UDP/ICMP, etc. In this section, we show the procedures for pre-processing raw data packet and raw dataset to create DataNets at the training platform, and to probe data packets at each SDN-HGW.

A. PRE-PROCESSING PACKET BYTE VECTOR

Pre-processing a raw data packet has three steps, *parsing*, *truncating/padding* and *normalization*. An overview of the pre-processing procedure is shown in Fig. 3. A raw data packet is processed byte by byte, similar to a pixel of an image which can be easily imported to a deep learning based classifier. *Parsing* is to remove the Ethernet header of a raw data packet. Data-link layer information such as MAC address, type of frame, etc., is not useful in packet classification. The parsing process reduces the input size of a packet. Moreover, some noise is filtered during the process for better classification accuracy.

Truncating and zero-padding is to fix the size of each data packet input to the classifier. An equal size of all inputs is required for the proposed deep learning based packet classifiers. In particular, define n as the targeted input size for DataNet, where $0 < n \leq 1500$. The maximum transmission unit has a size of 1500 bytes. An input packet is truncated or zero-padded, depending on its length compared to n.

An input after truncating and zero-padding is defined as a packet byte vector (PBV). For example, the i-th PBV is described as follows:

$$X_i = \{x_{i1}, x_{i2}, x_{i3}, \dots x_{in}\} \tag{1}$$

where x_{ij} denotes the j-th byte X_i. Each PBV is then *normalized* to $[0, 1]$ for faster classification. For simplicity, we assume X_i is the normalized result of the i-th PBV. Classification of a data packet is processed using the normalized PBV.

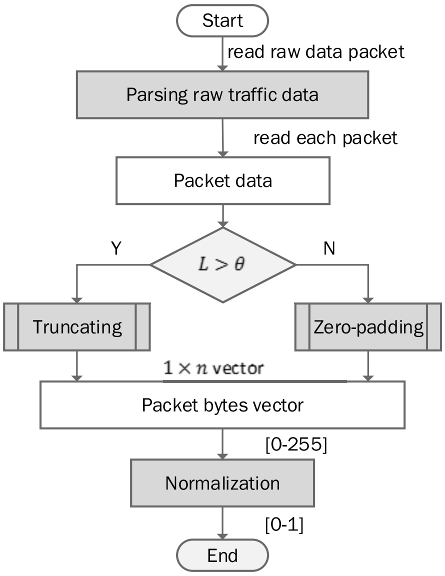


FIGURE 3. Raw data packet pre-processing.

B. PREPROCESSING PACKET BYTE MATRIX

To create DataNet, a dataset with labeled raw data packets is processed in the steps as shown in Fig. 4.

The raw dataset is processed into a packet byte matrix (PBM) $\mathbb{X}$ as follows:

$$\mathbb{X} = \{X_1^T, X_2^T, \ldots, X_m^T\}^T, \tag{2}$$

where $(\cdot)^T$ is the transposition function, and m is the number of raw data packet in the dataset. Fig. 5 shows an illustration of a PBM.

Each PBV X_i is associated with a label L_i, e.g., AIM, Email, Netflix, etc. After preprocessing, the raw dataset is composed of a PBM and a label vector, described as follows:

$$\mathbb{X} = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_m \end{bmatrix} \longleftarrow \begin{bmatrix} L_1 \\ L_2 \\ \vdots \\ L_m \end{bmatrix} \tag{3}$$

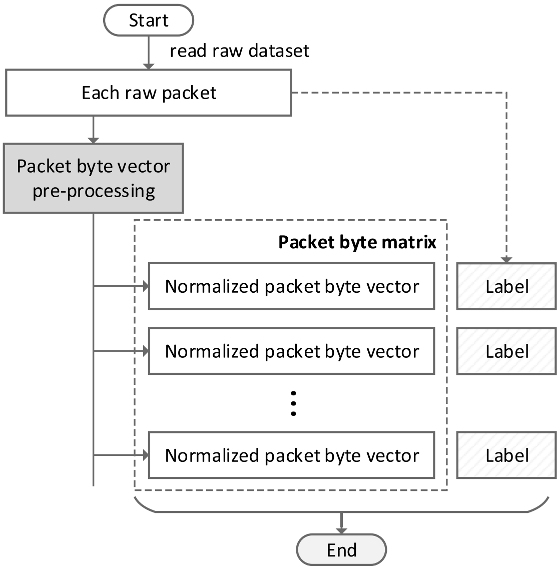
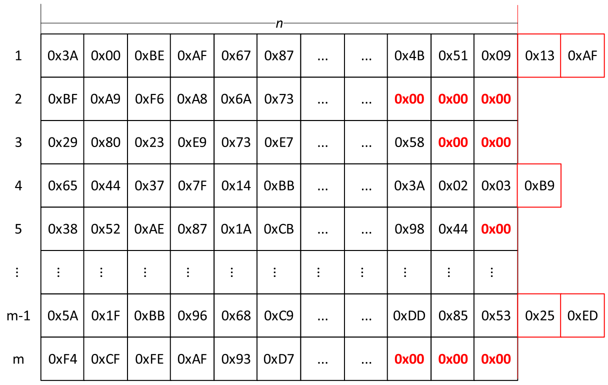

Start
read raw dataset
Each raw packet
Packet byte vector pre-processing
Packet byte matrix
Normalized packet byte vector
Label
Normalized packet byte vector
Label
⋮
Normalized packet byte vector
Label
End

FIGURE 4. Preprocessing the training dataset.



						n							
1	0x3A	0x00	0xBE	0xAF	0x67	0x87	...	...	0x4B	0x51	0x09	0x13	0xAF
2	0xBF	0xA9	0xF6	0xA8	0x6A	0x73	...	...	0x00	0x00	0x00		
3	0x29	0x80	0x23	0xE9	0x73	0xE7	...	...	0x58	0x00	0x00		
4	0x65	0x44	0x37	0x7F	0x14	0xBB	...	...	0x3A	0x02	0x03	0xB9	
5	0x38	0x52	0xAE	0x87	0x1A	0xCB	...	...	0x98	0x44	0x00		
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮		
m-1	0x5A	0x1F	0xBB	0x96	0x68	0xC9	...	...	0xDD	0x85	0x53	0x25	0xED
m	0xF4	0xCF	0xFE	0xAF	0x93	0xD7	...	...	0x00	0x00	0x00		

FIGURE 5. Illustration of a Packet Byte Matrix.

An example of a labeled PBM is shown in Table 1. Since dimension of data varies even for the same class of application, a fixed size will ease the maintenance and update with more future applications added to the system. Without loss of generality, 1480 is chosen for the rest of this work.

TABLE 1. Examples of the packet byte matrix with labels.

No	Label	Packet Byte Vector	Input size n	Original size
1	AIM		1480	42
2	Email		1480	108
⋮	⋮	⋮	⋮	⋮
m	Netflix		1480	1480

V. DEEP-LEARNING BASED ENCRYPTED DATA CLASSIFIER (DATANET) DESIGN

After preprocessing, $\mathbb{X}$ is applied to create DataNet, i.e. encrypted data packet classifiers. In particular, we develop three types of DataNet based on *Multi-Layer Perceptron (MLP)*, *Stacked AutoEncoder (SAE)* and *Convolutional Neural Network (CNN)* respectively.

A. MLP BASED DATANET

MLP is a class of feedforward artificial neural network (ANN) as shown in Fig. 6. A MLP consists of three or more layers. The first layer is for input data, i.e., a PBV X_i. One or more hidden layers extract features from the input. The last layer outputs a classification result. Each hidden layer, e.g., the i-th layer, is composed of multiple neurons that is mainly a nonlinear activation function as follows:

$$f(x) = \sigma\left(W^{(i)} \cdot x + b^{(i)}\right), \tag{4}$$

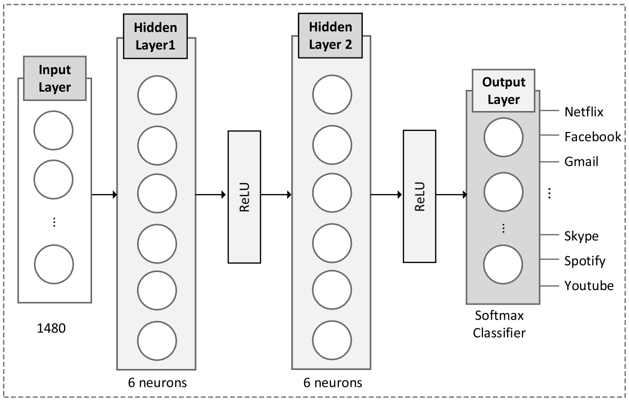


FIGURE 6. Overview of the MLP based DataNet.

where $\sigma(\cdot)$ is an activation function, e.g., $\sigma(x) = \tanh(x)$. The important characteristic of the activation function is that it provides a smooth transition as input values change. $W^{(i)}$ is a weight matrix and $b^{(i)}$ is a bias vector. There may be more than one hidden layer. Each layer passes through the same function with a different weight matrix and a bias vector. The final layer outputs the results of the last hidden layer, e.g., layer j, as follows:

$$o(x) = g\left(W^{(j)} \cdot x + b^{(j)}\right). \tag{5}$$

The MLP structure specified for the proposed packet classifier is shown in Fig. 6. It consists of one input layer, two hidden layers and one output layer. Using the full size of the data packet as an example, the input layer has 1480 inputs. The two hidden layers are composed of 6 and 6 neurons respectively. The output layer is composed of 15 neurons with Softmax as classifier. The classification process is defined as follows:

1) Input PBV $X_i = \{x_{i1}, x_{i2}, \ldots, x_{in}\}$ to the first hidden layer and compute the output as follows:

$$z^{(1)} = W^{(1)}\sigma\left(X_i\right) + b^{(1)}, \tag{6}$$

where $\sigma(\cdot)$ is the activation function, i.e. a rectified linear unit (ReLU). ReLU is a non-linear operation as follows:

$$ReLU(x) = \max[0, x]. \tag{7}$$

2) For hidden layer 2 compute the output as follows:

$$z^{(2)} = W^{(2)}\sigma\left(z^{(1)}\right) + b^{(2)}, \tag{8}$$

3) A fully-connected layer with a Softmax classifier output the final results as follows:

$$\hat{y} = \frac{\exp z^j}{\sum \exp z^i}, \tag{9}$$

where z^j is the output of the j-th neuron. $\hat{Y} = \{\hat{y}_1, \hat{y}_2, \hat{y}_3, ...\hat{y}_N\}$ is the complete set of classes, and N denotes number of classes. The output with the highest probability indicates the class of the input value.

In our MLP model, we use *cross entropy* as loss function and the calculation of gradient and updates of weights and bias are defined as follows:

1) Calculate the loss function of cross entropy between the output value and the label value as follows:

$$L = -\sum_{i=1}^{n} y_i \ln f(x_i, \theta), \tag{10}$$

2) Update weights and bias using gradient descent as follows:

$$\begin{aligned} w &\leftarrow w - \eta\frac{\partial L}{\partial w}, \\ b &\leftarrow b - \eta\frac{\partial L}{\partial b}. \end{aligned} \tag{11}$$

To start the training process, training parameters are set as $\{N_e, M, \eta\}$, where N_e is the maximum number of Epoch, M is the size of mini_batch used in the stochastic gradient method, η is the learning rate. The complete process for the training process is summarized in Alg. 1. Without loss of generality, the algorithm only summarizes the basic structure of the process. Stopping criteria such as validation is not given in the description.

Algorithm 1 MLP Based DataNet Training

```
Require: Training data, training parameters
Ensure: MLP based DataNet.
for $t = 1$ to $N_e$ do
    for each batch of $M$ input data do
        For each training samples $X_i \in \mathbb{X}$:
        Compute the output using Eq. (6);
        Process with activation function Eq. (7);
        Compute the output using Eq. (8);
        Process with activation function Eq. (7);
        Output classification results according to Eq. (9);
        Compute the training error according to Eq. (10);
        Update weights and bias according to Eq. (11);
    end for
end for
```

B. SAE BASED DATANET

We propose to design a classifier based on stacked autoencoders (SAE) [40]. An autoencoder is usually used for dimensionality reduction or feature extraction. In general, autoencoders are used for automatic features extraction. As shown in Fig. 7, the proposed SAE architecture consists of five layers, input, three encoders and the output layer. The input layer has a size of 1480, and the three encoders are stacked with sizes of 740, 92 and 32 neurons respectively.

The classification process is defined as follows:

1) Input PBV $X_i = \{x_{i1}, x_{i2}, \ldots, x_{in}\}$ with 1480 neurons to the first hidden layer, Encoder 1 with 740 neurons and compute the output using Eq. (6).

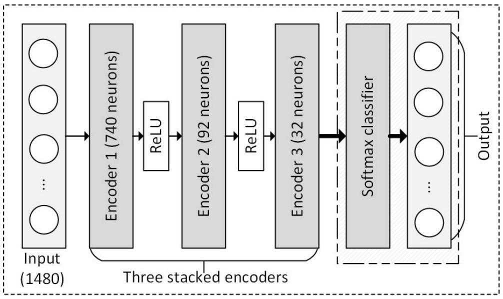


FIGURE 7. Overview of the SAE based DataNet.

2) After activation of ReLU according to Eq. (7), the results are as input to the second hidden layer, Encoder 2 with 92 neurons and using Eq. (8).
3) The same with above mentioned, after activation of ReLU, the results are as input to the third hidden layer, Encoder 3 with 32 neurons.
4) A fully-connected layer with a Softmax classifier output the final results as Eq. (9).

The training process of the classifier has two steps: *training encoders*; and *training the classifier*. As shown in Fig. 8, three encoders are trained in a greedy layer-wise fashion [41]. The process to train Encoder 1 is described as follows:

1) Input PBV $X_i = \{x_{i1}, x_{i2}, \ldots, x_{in}\}$, where $n = 1480$ in this design. Compute the output using Eq. (6). The process the results with an activation function, i.e. Eq. (7).
2) After activation with ReLU, the results are input to the output layer (i.e. Decoder 1 with 1480 neurons) and compute the output using Eq. (8) and the activation of Eq. (7).
3) Compute the reconstruction errors between input and output using mean squared error as follows:
$$\varepsilon(k) = \frac{1}{m}\sum_{i=1}^{m} e_j^2(k) \tag{12}$$
where $e_j(k) = \hat{y}_j(k) - y_j(k)$ is the error between the output and the targeted value and m is the number of samples.
4) Update the weight matrices based on the least mean squares algorithm through back-propagation. The change in each weight is as follows:
$$\Delta w_j(i) = -\eta \frac{\partial \varepsilon(k)}{\partial z_j(k)} z_i(k) \tag{13}$$
where z_i is the output of the previous neuron and η is the learning rate.

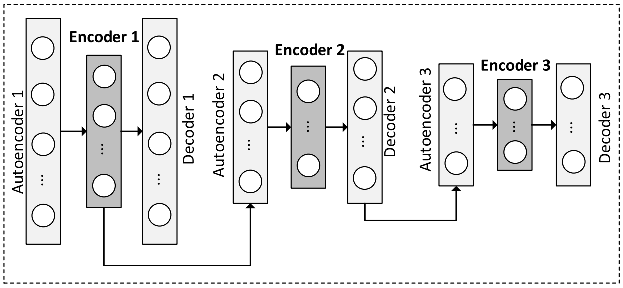


FIGURE 8. Encoder training for SAE based DataNet.

Encoder 1 is the input of successive layer to train Encoder 2 in a similar way, where the size of each layer is updated to targeted size. Encoder 3 is trained with Encoder 2 as the input.

Algorithm 2 SAE Based DataNet Training

```
Require: Training data, training parameters
Ensure: SAE based DataNet.
Initial training dataset $\mathbb{X}$:
for Three encoders do
    for $t = 1$ to $N_e$ do
        for each batch of $M$ input data do
            For each training samples $X_i \in \mathbb{X}$:
            Compute the output using Eq. (6);
            Process with activation function Eq. (7);
            Compute the reconstruction error, i.e. Eq. (12);
            Update the weights and bias according to Eq. (13);
        end for
    end for
    Use the current Encoder as the input of the successive layer.
end for
Train the DataNet based on Alg. 1.
```

In the second step of SAE based DataNet training, the final classifier can be trained using Alg. 1, given all three encoders. Note that the input has a size of 32 instead of 1480. The complete process for training the SAE based DataNet is summarized in Alg. 2. The training parameters are set as $\{N_e, M, \eta\}$, where N_e is the maximum number of Epoch, M is the size of *mini_batch* used in the stochastic gradient method, η is the learning rate.

C. CNN BASED DATANET

We develop encrypted packet classifiers based on CNN [42]. CNN is a typical deep learning network applied for classification. Different from a typical deep neural network, e.g., the artificial neural network, CNN applies function convolutions, as follows:
$$y(t) = x(t) * \omega(t), \tag{14}$$
where $x(t)$ is the input function and $\omega(t)$ is the kernel function. A CNN structure consists of three types of layer: input layer, hidden layer and output layer. Computational features include local receptive field, shared weights and bias, and pooling, as described below.

1) LOCAL RECEPTIVE FIELD

In an ordinary neural network, the inputs are depicted as a vertical line of neurons. In CNN, the layers have neurons arranged in three dimensions: width, height, depth. Each neuron in the l-th layer is connected to a small region of the neurons from the previous layer. The small region is called the *local receptive field*. We slide the local receptive field across the entire $(l-1)$-th layer. And there will be a different neuron in the l-th layer for each local receptive field.

2) SHARED WEIGHTS AND BIASES

transition from a layer to the next layer is defined by a weight matrix and a bias, e.g.,

$$f(x_i, w_i, b) = \sum_i w_i x_i + b. \tag{15}$$

In CNN, the weight matrix and the bias are usually shared for different transitions. The shared weights and bias are often called as *kernel* or *filter*. The advantage of applying shared weights and biases is that it greatly reduces the number of parameters.

3) POOLING

Pooling is an important component of a CNN. A pooling layer is usually used after a convolutional layer to reduce the dimensionality of the results from convolution. In the mean time, pooling would retain most information with the reduced dimensionality. One common procedure for pooling is *max-pooling*. In max-pooling, a pooling unit outputs the largest element within a rectangular subregion. Other popular procedures for pooling include *average pooling*, *L2 pooling*, etc.

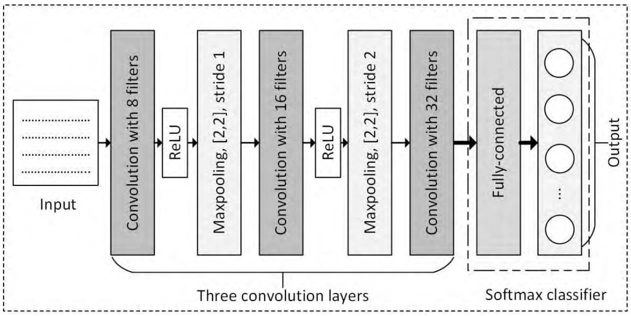

Input
Convolution with 8 filters
ReLU
Maxpooling, [2,2], stride 1
Convolution with 16 filters
ReLU
Maxpooling, [2,2], stride 2
Convolution with 32 filters
Fully-connected
...
Output
Three convolution layers
Softmax classifier

FIGURE 9. Overview of the CNN based DataNet.

The CNN structure specified for our proposed packet classifier is shown in Fig. 9. It consists of three convolution layers, 2 Maxpooling layers and a fully-connected layer with Softmax as classifier. Since the input data packet is converted into a two-dimensional (2D) matrix, we will discard depth and focus on processing the 2D data. The classification process is defined as follows:

1) The first convolutional layer processes the input data with 8 filters, where each filter has a size of [3, 3]. Each filter moves 1 step after one convolution operation.
2) Results of the convolution layer are input to an activation function, i.e. Eq. (7).
3) After activation of ReLU, the results are then processed through max pooling. In each step, the max pooling processes a [2, 2] input as follows:

$$\text{max pooling}\begin{bmatrix} x_1 & x_2 \\ x_3 & x_4 \end{bmatrix} = \max(x_1, x_2, x_3, x_4). \tag{16}$$

   The max pooling has a step size of 1.
4) The outputs of the max pooling are processed by the second convolutional layer with 16 [3, 3] filters. The step size in this layer is 2.
5) An activation of ReLU (i.e. Eq. (7)) follows to process the outputs.
6) The max pooling in the second layer has a size of [2,2] and step size 2.
7) The third convolutional layer has 32 [3, 3] filters.
8) A fully-connected layer with a Softmax classifier (i.e. Eq. (9)) outputs the final results.

In the model training process, a loss function is defined based on cross entropy, i.e., Eq. (10). Stochastic gradient method is applied to find weights and bias that computes the minimum loss.

To start the training process, training parameters are set as $\{N_e, M, \eta, K, S\}$, where N_e is the maximum number of Epoch, M is the size of mini_batch used in the stochastic gradient method, η is the learning rate, K is the number of filters, S is the step length. The complete training process is summarized in Alg. 3.

Algorithm 3 CNN based DataNet training

```
Require: Training data, training parameters
Ensure: CNN based DataNet.

for $t = 1$ to $N_e$ do
    for each batch of $M$ input data do
        For each training samples $x \in X$:
        Compute the convolutional results;
        Compute according to Eq. (7);
        Max pooling according to Eq. (16);
        Output classification results according to Eq. (9);
        Compute the training error according to Eq. (10);
        $(W, b) \leftarrow \arg\min L$;
    end for
end for
```

VI. EVALUATION AND EXPERIMENTAL RESULTS

In this section, we present the experimental results to evaluate the accuracy of the proposed DataNets. Moreover, we also compare the computational resource requirements for the three developed DataNets.

A. EXPERIMENT SETTINGS

1) DATASET FOR EVALUATION

The dataset for evaluation is selected from the "ISCX VPN-nonVPN traffic dataset" [23]. As shown in Table 2,

TABLE 2. Description of the chosen datasets.

Application	Security	Full dataset		Balanced dataset	
	Protocol	Quantity	Percentage	Quantity	Percentage
AIM	HTTPS	4869	2.356%	4869	6.634%
Email-Client	SSL	4417	2.137%	4417	6.018%
Facebook	HTTPS	5527	2.674%	5527	7.531%
Gmail	HTTPS	7329	3.546%	5000	6.813%
Hangout	HTTPS	7587	3.671%	5000	6.813%
ICQ	HTTPS	4243	2.053%	4243	5.781%
Netflix	HTTPS	51932	25.126%	5000	6.813%
SCP	SSH	15390	7.446%	5000	6.813%
SFTP	SSH	4729	2.287%	4729	6.443%
Skype	proprietary	4607	2.229%	4607	6.277%
Spotify	proprietary	14442	6.987%	5000	6.813%
torTwitter	proprietary	14654	7.089%	5000	6.813%
Vimeo	HTTPS	18755	9.074%	5000	6.813%
voipbuster	proprietary	35469	17.161%	5000	6.813%
Youtube	HTTPS	12738	6.163%	5000	6.813%
TOTAL		**206688**	**100%**	**73392**	**100%**

the total dataset for evaluation is composed of 15 applications, e.g., Facebook, Youtube, Netflix, etc. The chosen applications are encrypted with various security protocols, including HTTPS, SSL, SSH, and proprietary protocols. A total of 206, 688 data packets are included in the selected dataset. To reduce impacts from imbalanced dataset [43], e.g., Netflix accounts for 25.126% of the total dataset, we further create a subset with more balanced data samples for each application. The balanced subset has a total of 73, 392 data packets. It is composed of the same 15 applications, where each class accounts for around 6.18% of the total subset.

2) CONFIGURATIONS OF THE COMPUTING PLATFORM

The performance evaluations are conducted using a Thinkpad laptop computer with an Intel I7-7600U CPU at 2.8 GHz, 8 GB RAM and an external GPU (Nvidia GeForce GTX 1080) connected through Thunderbolt 3. The data preprocessing is based on Python package *Scapy* [44] to parse PCAP file. The software platform for deep learning is built on Keras library [45] with Tensorflow [46] (GPU-based version 1.4.0) as the back-end support.

3) PERFORMANCE METRICS

The performance metrics used for evaluations are *Precision*, *Recall* and F_1 *score* [47].

- ***Precision***: precision r_p is the ratio of *true positives* n_T^P over the sum of n_T^P and *false positives* n_F^P. In the proposed classification methods, precision is the percentage of packets that are properly attributed to the targeted application.
$$r_p = \frac{n_T^P}{n_T^P + n_F^P}. \tag{17}$$
- ***Recall***: recall r_c is the ratio of n_T^P over the sum of n_T^P and *false negatives* n_F^N or the percentage of packets in an application class that are correctly identified.
$$r_c = \frac{n_T^P}{n_T^P + n_F^N}. \tag{18}$$
- ***F_1-score***: the F_1 score r_f is a widely-used metric in information retrieval and classification that considers both precision and recall as follows:
$$r_f = \frac{2r_p \cdot r_c}{r_p + r_c}. \tag{19}$$

For simplicity, the evaluations in this work are conducted with full-size data packets, i.e., 1480 bytes per packet. Different settings for truncating/padding will be evaluated in our future work.

B. CREATING DATANETS

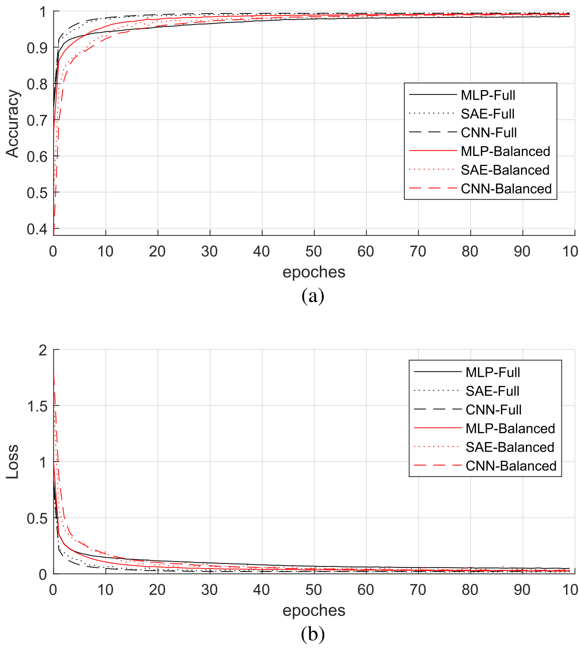
We first create two sets of *DataNets* trained from the full dataset and the balanced dataset respectively. Each set of evaluation consists of three DataNets trained from the three proposed methods, i.e., MLP, SAE and CNN based ones. Each training process uses 60% for training, 20% for validation and 20% for test. In total, the accuracy of 6 DataNets are evaluated for the packet classifier of an SDH-HGW. Settings for the three methods are given in Table 3.

TABLE 3. Settings for DataNet training.

	Max N_e	M	η	**Optimizer**
MLP based DataNets	100	128	0.01	SGDM
SAE Encoders	100	256	0.01	ADADELTA [50]
SAE based DataNets	100	256	0.01	ADAM [51]
CNN based DataNets	100	256	0.01	SGDM

C. CLASSIFICATION ACCURACY OF THE DATANETS

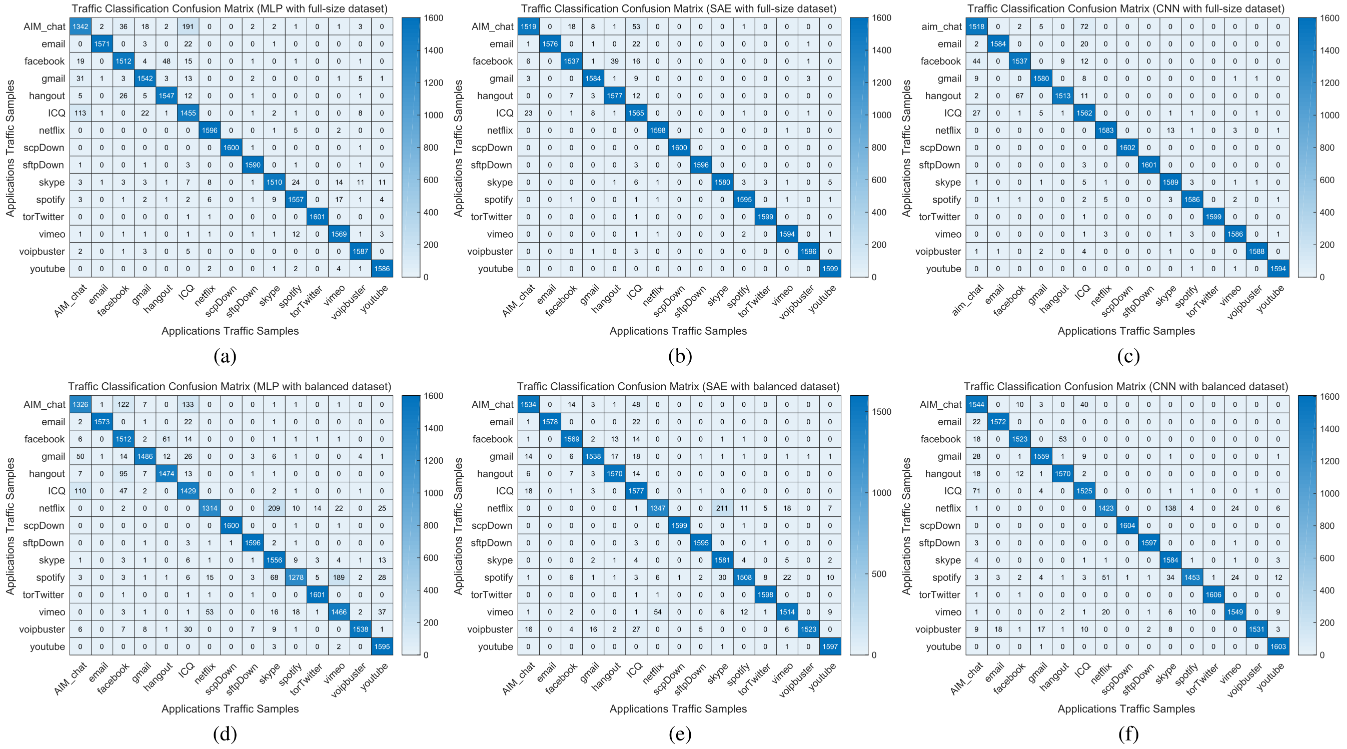
The accuracy of each DataNet model is evaluated with 50 tests. Each test is conducted using a randomly chosen



(a)

(b)

FIGURE 10. Training accuracy and loss. (a) Training accuracy. (b) Loss function.



(a) (b) (c)

(d) (e) (f)

FIGURE 11. Classification results of the six DataNets (average values from 50 tests). (a) MLP based DataNet (full dataset). (b) SAE based DataNet (full dataset). (c) CNN based DataNet (full dataset). (d) MLP based DataNet (balanced dataset). (e) SAE based DataNet (balanced dataset). (f) CNN based DataNet (balanced dataset).

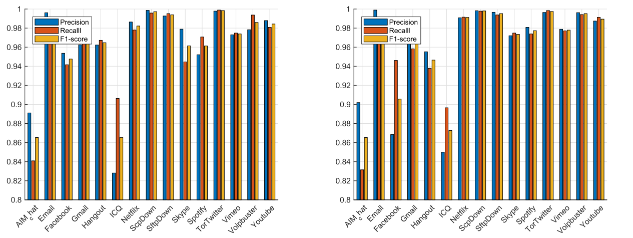
balanced subset from the full dataset. In each chosen subset, 40% are randomly chosen for classification test. For example, to test Netflix, 5000 samples are randomly chosen from the total pool of 51, 932. Then 2, 000 samples are further selected for classification test. The average of results of Precision, Recall and F1-Score are presented for each DataNet. Training Accuracy and Loss of 6 DataNets are shown in Fig. 10(a) and Fig. 10(b). All three approaches show promising convergence in training DataNets. The evaluation results of the six DataNets are summarized in Table 4.

TABLE 4. Classification accuracy of the DataNets.

	MLP		SAE		CNN	
	Full	Balanced	Full	Balanced	Full	Balanced
Precision						
Maximum	0.9714	0.9420	0.9914	0.9741	0.9930	0.9754
Average	0.9657	0.9342	0.9883	0.9692	0.9847	0.9696
Minimum	0.9595	0.9254	0.9851	0.9642	0.9628	0.9633
Recall						
Maximum	0.9717	0.9403	0.9915	0.9737	0.9920	0.9746
Average	0.9653	0.9309	0.9881	0.9678	0.9842	0.9685
Minimum	0.9582	0.9226	0.9847	0.9619	0.9613	0.9623
F1-Score						
Maximum	0.9694	0.9375	0.9905	0.9723	0.9891	0.9732
Average	0.9653	0.9308	0.9882	0.9678	0.9843	0.9685
Minimum	0.9603	0.9235	0.9855	0.9641	0.9656	0.9634

For better illustration, Fig. 11 shows the average results of the detailed classification for each DataNet. The elements on the diagonal of confusion matrix present accurately classified results. The inaccuracy of ICQ classification using MLP DataNets can be verified from Fig. 11(a) and Fig. 11(d). As we can see, many ICQ packets are wrongly classified as AIM_Chat and vice versa. Similar results for SAE based DataNets (Figs. 11(a) and 11(d)) and CNN based DataNets (Figs. 11(c) and 11(f)) are also given. We can see that both SAE and CNN DataNets have better performance in classifying the two on-line chatting applications. More discussion on each type of DataNet is given below.

The evaluation results of MLP based DataNets are shown in Fig. 12. The MLP DataNet trained from the full dataset achieves high *precision* for most applications, as shown in Fig. 12(a). Specifically, Email, Netflix, ScpDown, SftpDown and TorTwitter are close to 100%. Precision of AIM_Chat and ICQ has a relatively low precision at 88% and 83% respectively. Similarly, *recall* and *F1-score* are high

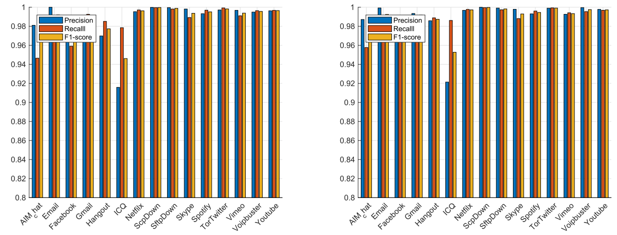


(a) (b)

FIGURE 12. Testing results of MLP based DataNets. (a) Trained from full dataset. (b) Trained from balanced dataset.

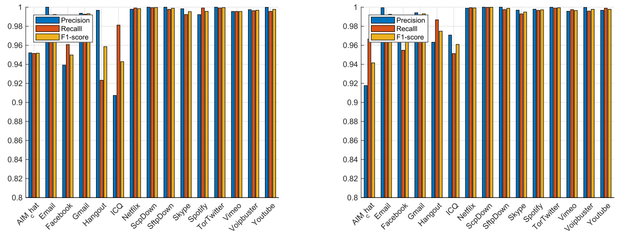
for most applications except AIM_Chat and ICQ. Three metrics of most applications achieve an average result of 96%. In comparison, the MLP DataNet trained from the balanced dataset has relatively low performance. In fact, the recall of Spotify classification is even lower than 80% with the MLP DataNet trained from the balanced dataset.

The evaluation results of SAE based DataNets are shown in Fig. 13. We can see that all precision, recall and F1-Score are much higher than MLP based DataNets. All three metrics achieve an average result of 98% for SAE based DataNets.



(a) (b)

FIGURE 13. Testing results of SAE based DataNets. (a) Trained from full dataset. (b) Trained from balanced dataset.



(a) (b)

FIGURE 14. Testing results of CNN based DataNets. (a) Trained from full dataset. (b) Trained from balanced dataset.

The evaluation results of CNN based DataNets are shown in Fig. 14. As we can see, the CNN based DataNets achieve even better results of precision, recall and F1-Score. All three metrics achieve an average result of 98%.

D. COMPUTATIONAL PERFORMANCE

We then evaluate the computational performance of the three types of DataNets. Without loss of generality, the three DataNets trained from the balanced dataset are used for evaluation. 50 tests are performed for each DataNet. Each test records the computational performance of classifying 150, 000 data packets. With a relatively powerful configuration, e.g. the configurations set for training, the classification processes can be performed in real-time for a home network with a bandwidth over 100 Mbps, as shown in Table 5.

In order to achieve real-time network management for a local user (e.g., a home user), the proposed DataNets shall be processed at the SDN-HGW instead of the SDN core network controller. Nonetheless, an SDN-HGW has limited computational on CPU, memory and flash. We conduct a case study with an SDN-HGW configured with a 4-core CPU (with more than 1500 DMIPS), 512 MB RAM and 128 MB Flash. As shown in Table 6, the MLP DataNet has the most computational efficiency. It can process over 10, 900 packets per second, or roughly 16.48 MB. In other words, real-time processing capability of MLP DataNet is over 131 Mbps.

TABLE 5. Computing performance with GPU settings.

	MLP DataNet	SAE DataNet	CNN DataNet
Speed	76 μs/step	146 μs/step	104 μs/step
Bandwidth	158 Mbps	82 Mbps	115 Mbps
GPU usage	$3-5\%$	$7-9\%$	$5-7\%$

TABLE 6. Computing performance with HGW settings.

	MLP DataNet	SAE DataNet	CNN DataNet
File Name	MLP.h5	SAE.h5	CNN.h5
File Size	178KB	12681KB	1467KB
# Parameters	12943	1359463	182927
F1-Score	0.9653	0.9882	0.9843
Per-step speed	91 μs/step	339 μs/step	2.61 ms/step
Bandwidth	131 Mbps	35.4 Mbps	4.6 Mbps
CPU usage	$7-8\%$	$19-21\%$	$59-63\%$
Memory usage	31MB	53.81MB	137.35MB

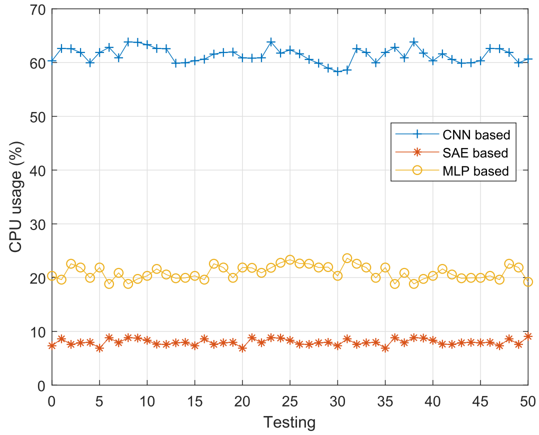


FIGURE 15. Analysis of CPU usage on an SDN-HGW.

Due to different complexity of the schemes, CPU and memory usages may vary. As shown in Fig. 15, the MLP based DataNet consumes the least computing resources amongst the three schemes, thus providing the largest bandwidth for real-time processing. We will further study this matter for better performance in the future work.

VII. CONCLUSION

In this paper, we proposed an SDN-HGW framework to support distributed end-to-end network QoS management. The core of the proposed SDN-HGW framework is DataNet, a deep learning based encrypted data packet classifier. The proposed DataNets are developed with three approaches, including MLP, SAE and CNN. An open dataset with more than 20, 000 packets from 15 applications were used to develop and test the proposed DataNets. The experimental results showed that the developed DataNets can be applied to the proposed SDN-HGW framework with accurate packet classification and high computational efficiency for real-time processing in a smart home network. In the future work, we will continue to improve the performance of DataNets and apply them to enhance network resource management, to enable new business plans, etc., without compromising security/privacy of service providers nor users.

REFERENCES

[1] P. Gallo, K. Kosek-Szott, S. Szott, and I. Tinnirello, "Sdn@home: A method for controlling future wireless home networks," *IEEE Commun. Mag.*, vol. 54, no. 5, pp. 123–131, May 2016.

[2] S. Wu *et al.*, "Survey on prediction algorithms in smart homes," *IEEE Internet Things J.*, vol. 4, no. 3, pp. 636–644, Jun. 2017.

[3] A. Kortebi, P. Le Dain, and F. Duré, "Home network assistant: Towards better diagnostics and increased customer satisfaction," in *Proc. Global Inf. Infrastruct. Symp. (GIIS)*, Oct. 2013, pp. 1–6.

[4] Y. C. Tung, W. J. Hwang, and C. H. Ho, "A novel QoS mapping algorithm for heterogeneous home networks using general regression neural networks," in *Proc. IEEE 17th Int. Conf. Comput. Sci. Eng.*, Dec. 2014, pp. 519–526.

[5] D. Kreutz, F. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.

[6] S. Bera, S. Misra, and A. V. Vasilakos, "Software-defined networking for Internet of Things: A survey," *IEEE Internet Things J.*, vol. 4, no. 6, pp. 1994–2008, Dec. 2017.

[7] S. V. Morzhov and M. A. Nikitinskiy, "Development and research of the prefirewall network application for floodlight SDN controller," in *Proc. Moscow Workshop Electron. Netw. Technol. (MWENT)*, Mar. 2018, pp. 1–4.

[8] F. Hadi, M. Imran, M. H. Durad, and M. Waris, "A simple security policy enforcement system for an institution using SDN controller," in *Proc. 15th Int. Bhurban Conf. Appl. Sci. Technol. (IBCAST)*, Jan. 2018, pp. 489–494.

[9] R. Trivisonno, R. Guerzoni, I. Vaishnavi, and A. Frimpong, "Network resource management and QoS in SDN-enabled 5G systems," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2015, pp. 1–7.

[10] S. O. Aliyu, F. Chen, and Y. He, "QoS-aware resource management in SDN-based interclouds: A software cybernetics perspective," in *Proc. IEEE Int. Conf. Softw. Qual., Rel. Secur. Companion (QRS-C)*, Jul. 2017, pp. 429–434.

[11] J. Rückert, R. Bifulco, M. Rizwan-Ul-Haq, H.-J. Kolbe, and D. Hausheer, "Flexible traffic management in broadband access networks using software defined networking," in *Proc. IEEE Netw. Oper. Manage. Symp. (NOMS)*, May 2014, pp. 1–8.

[12] S. Yi, Z. Hao, Z. Qin, and Q. Li, "Fog computing: Platform and applications," in *Proc. 3rd IEEE Workshop Hot Topics Web Syst. Technol. (HotWeb)*, Nov. 2015, pp. 73–78.

[13] M. Amiri, H. Al Osman, and S. Shirmohammadi, "SDN-enabled game-aware network management for residential gateways," in *Proc. IEEE Int. Symp. Multimedia (ISM)*, Dec. 2017, pp. 330–333.

[14] J. Proença *et al.*, "Building an NFV-based vRGW: Lessons learned," in *Proc. 14th IEEE Annu. Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2017, pp. 653–658.

[15] M. Uddin and T. Nadeem, "TrafficVision: A case for pushing software defined networks to wireless edges," in *Proc. IEEE 13th Int. Conf. Mobile Ad Hoc Sensor Syst. (MASS)*, Oct. 2016, pp. 37–46.

[16] A. C. Baktir, A. Ozgovde, and C. Ersoy, "How can edge computing benefit from software-defined networking: A survey, use cases, and future directions," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2359–2391, 4th Quart., 2017.

[17] J. Clark and P. C. van Oorschot, "SoK: SSL and HTTPS: Revisiting past challenges and evaluating certificate trust model enhancements," in *Proc. IEEE Symp. Security Privacy*, May 2013, pp. 511–525

[18] M. Y. Rhee, *Network Layer Security*. Hoboken, NJ, USA: Wiley, 2013. [Online]. Available: https://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=8043830

[19] Y. Li and J. Li, "MultiClassifier: A combination of DPI and ML for application-layer classification in SDN," in *Proc. 2nd Int. Conf. Syst. Inform. (ICSAI)*, Nov. 2014, pp. 682–686.

[20] S. Jeong, D. Lee, J. Hyun, J. Li, and J. W.-K. Hong, "Application-aware traffic engineering in software-defined network," in *Proc. 19th Asia–Pacific Netw. Oper. Manage. Symp. (APNOMS)*, Sep. 2017, pp. 315–318.

[21] D. Sanvito, D. Moro, and A. Capone, "Towards traffic classification offloading to stateful SDN data planes," in *Proc. IEEE Conf. Netw. Softw. (NetSoft)*, Jul. 2017, pp. 1–4.

[22] P. Wang, F. Ye, and X. Chen, "Smart devices information extraction in home Wi-Fi networks," *Internet Technol. Lett.*, vol. 1, no. 3, p. e42, 2018.

[23] A. H. Lashkari, G. Draper-Gil, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of encrypted and VPN traffic using time-related features," in *Proc. Int. Conf. Inf. Syst. Secur. Privacy (ICISSP)*, Feb. 2016, pp. 407–414.

[24] M. Hayes, B. Ng, A. Pekar, and W. K. G. Seah, "Scalable architecture for SDN traffic classification," *IEEE Syst. J.*, to be published, doi: 10.1109/JSYST.2017.2690259.

[25] A. W. Moore and K. Papagiannaki, "Toward the accurate identification of network applications," in *Passive and Active Network Measurement*, C. Dovrolis, Ed. Berlin, Germany: Springer, 2005, pp. 41–54.

[26] A. Madhukar and C. Williamson, "A longitudinal study of P2P traffic classification," in *Proc. 14th IEEE Int. Symp. Modeling, Anal., Simulation*, Sep. 2006, pp. 179–188.

[27] *Service Name and Transport Protocol Port Number Registry*. Accessed: Sep. 13, 2018. [Online]. Available: https://www.iana.org/assignments/service-names-port-numbers

[28] D. C. Sicker, P. Ohm, and D. Grunwald, "Legal issues surrounding monitoring during network research," in *Proc. 7th ACM SIGCOMM Conf. Internet Meas. (IMC)*. New York, NY, USA: ACM, 2007, pp. 141–148, doi: 10.1145/1298306.1298307.

[29] T. T. T. Nguyen and G. Armitage, "A survey of techniques for Internet traffic classification using machine learning," *IEEE Commun. Surveys Tuts.*, vol. 10, no. 4, pp. 56–76, 4th Quart., 2008.

[30] M. Finsterbusch, C. Richter, E. Rocha, J.-A. Muller, and K. Hanssgen, "A survey of payload-based traffic classification approaches," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 2, pp. 1135–1156, 2nd Quart., 2014.

[31] G. Li, M. Dong, K. Ota, J. Wu, J. Li, and T. Ye, "Deep packet inspection based application-aware traffic control for software defined networks," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2016, pp. 1–6.

[32] P. Wang, S.-C. Lin, and M. Luo, "A framework for QoS-aware traffic classification using semi-supervised machine learning in SDNs," in *Proc. IEEE Int. Conf. Services Comput. (SCC)*, Jun./Jul. 2016, pp. 760–765.

[33] Z. Fan and R. Liu, "Investigation of machine learning based network traffic classification," in *Proc. Int. Symp. Wireless Commun. Syst. (ISWCS)*, Aug. 2017, pp. 1–6.

[34] Y. Bengio, A. Courville, and P. Vincent. (Jun. 2012). "Representation learning: A review and new perspectives." [Online]. Available: https://arxiv.org/abs/1206.5538

[35] J. Schmidhuber. (Apr. 2014). "Deep learning in neural networks: An overview." [Online]. Available: https://arxiv.org/abs/1404.7828

[36] Z. Wang. (2015). *The Application of Deep Learning on Traffic Identification*. [Online]. Available: http://www.blackhat.com

[37] M. Lotfollahi, R. S. H. Zade, M. J. Siavoshani, and M. Saberian. (Sep. 2017). "Deep packet: A novel approach for encrypted traffic classification using deep learning." [Online]. Available: https://arxiv.org/abs/1709.02656

[38] *Using eBPF to Accelerate OVS Datapath*. Accessed: Nov. 23, 2016. [Online]. Available: https://bit.ly/2JeocK7

[39] *PCAP Next Generation Dump File Format*. Accessed: Aug. 30, 2004. [Online]. Available: http://www.tcpdump.org/pcap/pcap.html

[40] S. Yadav and S. Subramanian, "Detection of application layer DDoS attack by feature learning using stacked AutoEncoder," in *Proc. Int. Conf. Comput. Techn. Inf. Commun. Technol. (ICCTICT)*, Mar. 2016, pp. 361–366.

[41] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Proc. 19th Int. Conf. Neural Inf. Process. Syst. (NIPS)*. Cambridge, MA, USA: MIT Press, 2006, pp. 153–160. [Online]. Available: http://dl.acm.org/citation.cfm?id=2976456.2976476

[42] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org

[43] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009.

[44] *Scapy*. Accessed: Sep. 28, 2018. [Online]. Available: https://scapy.net/

[45] F. Chollet *et al.* (2015). *Keras*. [Online]. Available: https://github.com/keras-team/keras

[46] M. Abadi *et al.* (Mar. 2016). "TensorFlow: Large-scale machine learning on heterogeneous distributed systems." [Online]. Available: https://arxiv.org/abs/1603.04467

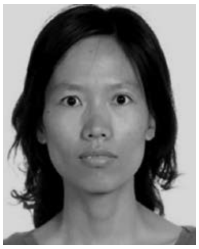
[47] H. Kim, K. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee, "Internet traffic classification demystified: Myths, caveats, and the best practices," in *Proc. ACM CoNEXT Conf. (CoNEXT)*. New York, NY, USA: ACM, 2008, pp. 11-1–11-12, doi: 10.1145/1544012.1544023.

[48] M. D. Zeiler, "ADADELTA: An adaptive learning rate method." *CoRR*, vol. abs/1212.5701, pp. 1–6, Dec. 2012.

[49] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, pp. 1–15, Dec. 2014. [Online]. Available: http://dblp.uni-trier.de/db/journals/corr/corr1412.html#KingmaB14

PAN WANG (M'18) received the B.S. degree from the Department of Communication Engineering, Nanjing University of Posts and Telecommunications, Nanjing, China, in 2001, and the Ph.D. degree in electrical and computer engineering from the Nanjing University of Posts and Telecommunications in 2013. From 2017 to 2018, he was a Visiting Scholar with the Department of Electrical and Computer Engineering, University of Dayton. He is currently an Associate Professor with the School of Modern Posts, Nanjing University of Posts and Telecommunications. His research interests include cyber security and communication network security, network measurements, quality of service, deep packet inspection, SDN, and big data analytics and its applications.

FENG YE (S'12–M'15) received the B.S. degree from the Department of Electronics Engineering, Shanghai Jiao Tong University, Shanghai, China, in 2011, and the Ph.D. degree in electrical and computer engineering from the University of Nebraska–Lincoln (UNL), NE, USA, in 2015. He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, University of Dayton (UD), Dayton, OH, USA. Prior to joining UD, he was with the Department ofElectrical and Computer Engineering, UNL, as an Instructor and a Researcher from 2015 to 2016. His research interests include cyber security and communication network security, wireless communications and networks, green ICT, smart grid communications and energy optimization, and big data analytics and its applications. He serves as the Secretary of the IEEE Technical Committee on Green Communications and Computing. He also serves as a TPC Member for numerous international conferences, including INFOCOM, GLOBECOM, VTC, and ICC. He was a recipient of the 2015 Top Reviewer Award from the IEEE Vehicular Technology Society. He served as the Co-Chair IEEE ICC 2018 and the Cognitive Radio and Networking Symposium. He served as the Publicity Co-Chair of the IEEE CBDCom 2018. He also served as the Co-Chair of the Signal Processing for Communications Symposium, ICNC 2019. He is also a reviewer for several IEEE journals, including the IEEE Transactions on Big Data, the IEEE Transactions on Green Communications and Networking, the IEEE Transactions on Smart grid, the IEEE Transactions on Vehicular Technology, and the IEEE Transactions on Wireless Communications. He is currently an Associate Editor of *Security and Privacy* (Wiley) and *China Communications*.



XUEJIAO CHEN (M'18) received the B.S. degree from the Department of Communication Engineering, Nanjing University of Posts and Telecommunications, Nanjing, China, in 2001, and the master's degree in electrical and computer engineering from the Nanjing University of Posts and Telecommunications, in 2006. Since 2017, she has been a Visiting Scholar with the Department of Electrical and Computer Engineering, University of Dayton. She is currently an Assistant Professor with the Department of Communication Engineering, Nanjing College of Information Technology, Nanjing. Her research interests include wireless communications and networks, cyber security and communication network security, network measurements, quality of service, and deep packet inspection.

YI QIAN (M'95–SM'07) received the Ph.D. degree in electrical engineering from Clemson University. He is currently a Professor with the Department of Electrical and Computer Engineering, University of Nebraska–Lincoln (UNL). Prior to joining UNL, he was with the telecommunications industry, academia, and the government. Some of his previous professional positions include serving as a Senior Member of Scientific Staff and a Technical Advisor with Nortel Networks, a Senior Systems Engineer and a technical advisor at several start-up companies, an Assistant Professor with the University of Puerto Rico at Mayagüez, Mayagüez, and a Senior Researcher with the National Institute of Standards and Technology. His research interests include information assurance and network security, network design, network modeling, simulation and performance analysis for next generation wireless networks, wireless ad hoc and sensor networks, vehicular networks, smart grid communication networks, broadband satellite networks, optical networks, high-speed networks, and the Internet. He was the Chair of the Technical Committee for Communications and Information Security, IEEE Communications Society, from 2014 to 2015. He was the Technical Program Chair of the IEEE International Conference on Communications 2018. He is serving on the editorial boards for several international journals and magazines. He is also serving as the Editor-in-Chief for the *IEEE Wireless Communications Magazine*. He is currently a Distinguished Lecturer with the IEEE Vehicular Technology Society and the IEEE Communications Society.