IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

A Novel SDN-based Application-Awareness Mechanism by Using Deep Learning

NAN HU[1], FANGJUN LUAN[1], XIAOXI TIAN[1], AND CHENGDONG WU[2]
[1]Information & Control Engineering Faculty, Shenyang Jianzhu University, Shenyang 110168, China
[2]Robot Science & Engineering Faculty, Northeastern University, Shenyang 110819, China

Corresponding author: Nan Hu (e-mail: zerovanila@sina.com)

ABSTRACT With the rapid development of the Internet of Things (IoT) and smart cities, more and more types of applications have been emerging. In fact, different applications have different features and different requirements on services. In order to satisfy users' Quality of Service (QoS) requirements, the application-awareness technique should be leveraged to distinguish different applications for providing the differentiated services. However, the traditional Internet only can obtain the local network view, which belongs to the offline awareness mode and cannot adapt to the dynamical network environment. At the right time, Software-Defined Networking (SDN) has been accepted as a new networking paradigm thanks to its network awareness on the global status information, which can greatly facilitate the online application-awareness. At present, three ways, i.e., port number, depth packet inspection and deep learning can be used for the application-awareness. To the best of our knowledge, the deep learning based application-awareness method is the most cutting-edge technique. In spite of this, the previous related schemes fail to effectively guarantee the correctness and stability. To this end, this paper proposes a Convolutional Neural Network (CNN) based deep learning mechanism to do the application-awareness, including three phases, i.e., traffic collection, data pre-processing and application-awareness. The SDN environment is implemented based on the MiniNet and the simulation experiments are made based on the TensorFlow. The experimental results show that the proposed application-awareness mechanism outperforms three benchmarks on recall ratio, precision ratio, F value and stability.

INDEX TERMS Application-Awareness; Software-Defined Networking; Deep Learning; Convolutional Neural Network.

I. INTRODUCTION

IN the traditional Internet, the network only supports the best effort services and fails to provide the differentiated services, that is, all applications have a fair competition on the limited network resources [1]. Furthermore, with the rapid development of the Internet of Things (IoT) [2] and smart cities [3], there are more and more new types of applications emerging, such as online live and online payment. In particular, different applications have different features and different requirements on services [4]. For example, the online video belongs to the real-time application, which has high demand on delay and delay jitter; the email belongs to the non-real-time application, which has high demand on packet loss ratio. Therefore, in order to satisfy users's Quality of Service (QoS) requirements, it requires to study the application-awareness technique while guaranteeing to provide the efficiently differentiated services (like the above, the applications of online videos are given the priority to be processed).

Before doing the application-awareness, it is considerably significant to quickly obtain the traffic information of all applications (i.e., traffic collection). However, the traditional network (i.e., the current Internet) lacks of the function of automatically obtaining traffic information, and it usually needs to spend much labor and time gathering the statistical information. At the right time, Software-Defined Networking (SDN) has been accepted as a new networking paradigm due to its network awareness on the global status information, which can collect the traffic information of all applications and greatly facilitate the online application-awareness [5]-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3021185, IEEE Access

[7].

In terms of the SDN-based application-awareness, to the best of our knowledge, there are three ways for addressing it, i.e., port number, Depth Packet Inspection (DPI) and deep learning [8]. To be specific, the port number based technique usually uses the common port number (0-1023) and the universal port number (1025-49151) to recognize the applications from the transmission layer. However, it is a credible technique no longer. For example, the application program can uses another port number to grasp the control privilege of network operating systems [9]. For DPI, each application has an unique signature for its recognition [10]-[11]. However, it requires to resolve all fields of packet in case of using DPI, which consumes lots of CPU resources and has a serious influence on the scalability. With respect to this, some new techniques such as Deterministic Finite Automata (DFA), Field Programmed Gate Array (FPGA) and Graphics Processing Units (GPU) are usually employed to optimize DPI [12]. In spite of this, these techniques always have some limitations such as low efficiency, correctness and stability. Under such context, more and more researchers pay attention to the deep learning based application-awareness method which is the most cutting-edge technique.

The deep learning based application-awareness technique usually depends on the feature statistics of traffic which can be obtained by the header of packet rather than the whole packet, thus its consumed time is smaller than that consumed by DPI [13]. For the deep learning, it involves the supervised learning strategy [14], unsupervised learning strategy [15] and semi-supervised learning strategy [16]. Among them, the unsupervised learning strategy can adapt to these data without the special mark which exits in the most common datasets. In spite of this, the previous deep learning based application-awareness schemes fail to effectively guarantee the correctness and stability. To this end, this paper proposes a Convolutional Neural Network (CNN) [17]-[18] based deep learning mechanism which belongs to the unsupervised learning strategy to do the application-awareness. CNN has two distinguished features, i.e., local connection and parameters sharing, and it is a network model with the multiple hidden layers, including an input layer, several convolution layers, several pooling layers, two fully connected layers and an output layer, where each hidden layer consists of many feature patterns and each feature pattern involves many neurons. Based on such architecture mode, CNN can effectively decrease the complexity of network structure and the overhead of updating network parameters.

Given the above consideration, this paper uses the CNN-based Deep learning method to address the SDN-based Application-awareness (CDSA), and the major contributions are summarized as follows.

- The system framework of CDSA is proposed, which is composed of three phases, i.e., traffic collection, data pre-processing and application-awareness. Among them, the first two are the preproduction phases while the last one is the practical-operation phase.
- The traffic information is collected under the SDN environment based on the OpenFlow and the data normalization is realized based on the min-max method.
- The application-awareness is done by the CNN model, including four parts, i.e., activation function, pooling function, classification function and loss function which are realized by Rectified Linear Unit (ReLU), t-distributed Stochastic Neighbor Embedding (t-SNE), Softmax and gradient descent algorithm respectively.
- The proposed CDSA is implemented, and the experimental results show it has more efficient recall ratio, precision ratio, F value (including $F1$ value) and stability than three benchmarks.

The following Section 2 fully reviews and compares the related work. Section 3 presents and analyzes the proposed CDSA in details. The performance evaluation is made in Section 4. Finally, Section 5 concludes this paper.

II. RELATED WORK

In recent years, three ways, i.e., port number, DPI and deep learning have been widely used for the SDN-based application-awareness. To the best of our knowledge, the port number based technique has been laid aside gradually [19]. Given this, we in this section only reviews the related work on the other two techniques for the SDN-based application-awareness (also called application classification or traffic classification).

In [12], DPI was introduced into the control plane of SDN seamlessly by extending the structure of flow table, in which both traffic behaviors and network status were exploited by SDN cooperatively. In particular, to improve the efficiency of application-awareness module, the packet would be sent to SDN controller for identification only when it arrived at the first time. In [20], a DPI offloading mechanism for traffic classification based on a stateful SDN data plane in network switches was proposed, which could reduce the amount of traffic volume inspected by the DPI without reducing classification accuracy.

In [21], a software-defined traffic classification framework was introduced, which dynamically selected the best suitable flow features and most effective machine learning classifiers with the help of a controller and a group of virtual functions. In [22], a simple architecture deployed in an enterprise network that gathered traffic data using the OpenFlow protocol was devised to obtain the high accuracy classification by using the supervised learning method, including random forests and two variations of gradient boosting classifiers i.e., stochastic gradient boosting and extreme gradient boosting. In [23], an efficient sampling and classification approach with the two-phase elephant flow detection was proposed. In the first phase, the sampling efficiency was improved by estimating the arrival interval of elephant flows and filtering out the redundant samples. In the second phase, the samples were classified with a new supervised classification algorithm based on correlation among data flows. In [24], a support vector machine based internet traffic identification and clas-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3021185, IEEE Access

TABLE 1. The summarized related work.

Reference	Approach	Result	Category
[12]	Introduce DPI into SDN controller, where network states and traffic behaviors are exploited	Facilitate the improvement of throughput and reduce latency time of end-to-end communication	DPI
[20]	Exploit the stateful SDN data planes to entirely delegate the filtering logic for traffic classification down to the switches	Reduce the amount of traffic volume	DPI
[21]	Propose the virtual network functions to flexibly select and apply the best suitable machine learning classifiers at run time	Improve the accuracy of classification by up to 13%	Deep learning
[22]	Employ the supervised learning method, including random forests and two variations of gradient boosting classifiers i.e., stochastic gradient boosting and extreme gradient boosting	Obtain the high accuracy classification	Deep learning
[23]	Design an efficient sampling and classification approach with the two-phase elephant flow detection	Provide the accurate detection with less sampled packets and shorter detection time	Deep learning
[24]	Introduce a support vector machine based internet traffic identification and classification	Get the 99.00% classification accuracy for YouTube traffic type and the 92.78% accuracy for YouTube streaming in length and quality	Deep learning
[25]	Propose the deep learning based method to realize the application-aware network resource allocation for SDN	Improve the network service quality and resource utilization rate	Deep learning
[26]	Use the semi-supervised traffic classification to design an application-aware SDN architecture, including core controller, access controller and data collecting controller	Optimize traffic classifiers and achieve the valid traffic classification with as few as 20% of labeled data entries in the training data-sets	Deep learning
[27]	Develop an expanded framework to prepare network data and apply the supervised machine learning techniques with application performance feedback	Enhance the network to adapt to the performance objectives of applications	Deep learning
[28]	Identify problems with the current application-layer classification in campus network and analyze the advantage of doing application-layer classification with SDN	Improve the recognition/accuracy rate and throughput	DPI and deep learning
[29]	Classify the network traffic into different classes according to the QoS requirements, providing the crucial information to enable the fine-grained and QoS-aware traffic engineering	Provide good performance in terms of classification accuracy and communication costs	DPI and deep learning
[30]	Similar to [29] but with the multiple classifiers for training	Achieve good classification accuracy	DPI and deep learning

sification was proposed to identify application traffic, which not only classified traffic type but also could classify the streaming in length and quality. In [25], the deep learning based method was proposed to realize the application-aware network resource allocation for SDN. The controller worked as the brain of the network which acquired all the network status information. The virtualized network functions carried in the deep neural networks model worked at the data plane and was responsible for classifying and reporting the traffic class information. It could greatly improve the network service quality and resource utilization rate. In [26], an application-aware SDN architecture was devised by using semi-supervised traffic classification, where three different control points were designated, i.e., core controller, access controller and data collecting controller. In [27], the application-awareness in SDN was expanded to enable any application to have a customized performance-based metric. It automatically analyzed and translated an arbitrary performance metric into a comprehensible application metric used for making decisions in the network by the deep learning.

There have been a few proposals on the combination of DPI and deep learning. For example, in [28], an application layer classifier combining both DPI and deep learning was designed to do classification in SDN. It tried to take advantage of the two classifiers to achieve a high speed while maintaining the acceptable accuracy rate. In [29], a QoS-aware traffic classification framework for SDN was proposed by jointly exploiting DPI and semi-supervised deep learning so that realizing the accurate traffic classification, while requiring minimal communications between the network controller and the SDN switches. It classified the network traffic into different classes according to the QoS requirements, which provided the crucial information to enable the fine-grained and QoS-aware traffic engineering. Similar to [29], [30] also proposed an SDN flow classification framework using DPI and semi-supervised deep learning. However, different from [29], [30] used the multiple classifiers for training.

With respect to the related researches, they are summarized in Table 1, including the simple descriptions on approach and result.

Although the above reviewed proposals could address the SDN-based application-awareness to some extent, the DPI-based methods (i.e., [12] and [20]) consumed too many CPU resources and had a serious influence on the scalability. In addition, DPI devices were introduced into the network, which increases the complexity of network and there were some privacy concerns. In particular, the current many applications adopted the encryption technique for protecting users' privacy, which caused that it is scarcely possible to exploit DPI to do the SDN-based application-awareness. Furthermore, the deep learning based methods (i.e., [21]-[30]) failed to effectively guarantee the correctness and stability. Different from them, this paper proposes a CNN-based deep learning mechanism to do the application-awareness, which not only effectively decreases the complexity of network structure and the overhead of updating network parameters from the perspective of theory, but also can guarantee the correctness and stability (i.e., recall ratio, precision ratio and F value) from the perspective of experiment.

III. THE PROPOSED CDSA

In this section, we first present the system framework of CDSA. Then, we give the detailed introduction for each major module mentioned in the system framework. Finally,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3021185, IEEE Access

we introduce the K-fold Cross Validation (K-CV) to partition the dataset.

A. SYSTEM FRAMEWORK

As depicted in Figure 1, the proposed CDSA system framework consists of four major modules, i.e., traffic collection, data pre-processing, features modelling and application-awareness. Among them, the traffic collection module is responsible for collecting traffic information under the SDN environment based on the OpenFlow. The second module is used to do data normalization for these collected traffic features. After the data pre-processing, a large proportion of data is regarded as the training sample used for features modelling based on TensorFlow, which is completed by the third module. For the remaining data, it is regarded as the test sample used for application-awareness based on CNN. In particular, the third module supports the last module; to be specific, the output of application-awareness module depends on the features modelling module. In the next sections, we will introduce traffic collection module, data pre-processing module and application-awareness module in details irrespective of the introduction of features modelling module, this is because the TensorFlow [31] is a classical training way and there is no need to over-introduce it.

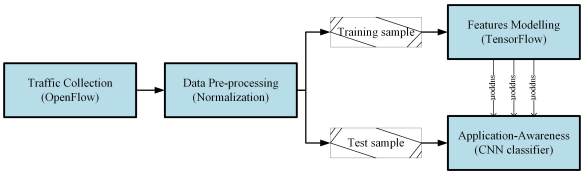


FIGURE 1. The system framework. The features modelling module is implemented based on TensorFlow, which is a classical training way and is specifically introduced no longer in this paper.

B. TRAFFIC COLLECTION

The traditional traffic collection usually adopts the offline pattern because the network fails to automatically obtain the traffic information. Under such context, it requires an online pattern to support the efficient application-awareness without taking much labor and time to gather the statistical information. To our best of knowledge, SDN has the awareness function on the global network status information, that is to say, it can collect the traffic information of all applications without the additional overhead. In fact, for the traffic collection in SDN, it requires to use some flow tables which are installed by the OpenFlow protocol. Given this, under SDN environment, we deploy the OpenFlow switch to help complete the traffic collection. On this basis, the network deployment topology of traffic collection is shown in Figure 2. The upper half of Figure 2 is the SDN network environment while the bottom half of Figure 2 is the traditional Internet, and their communication is based on two switches, i.e., general switch and OpenFlow switch. The corresponding process is simply described as follows. The general switch collects the traffic of all hosts. According to the techniques of port mirroring and redirection, the general switch submits its collected traffic information to the OpenFlow switch (i.e., traffic replication). Finally, the OpenFlow switch transmits the related traffic to the SDN controller to which it belongs. Unlike the above way, the SDN controller is connected with the general switch directly. However, such way without deploying the OpenFlow is considerably difficult to do engineering implementation due to the fact that many application program interfaces need to be written.

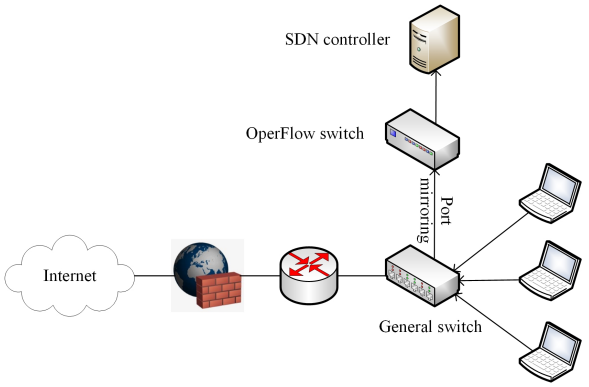


FIGURE 2. The illustration of traffic collection.

C. DATA PRE-PROCEEDING

Each application is composed of many features, such as source port number, destination port number, duration, etc. In particular, these features are not at the same order of magnitude and have no the mutual comparability among them. In order to conveniently facilitate the features modelling and application-awareness, these features should be at the same order of magnitude via the operation of normalization. Suppose that there are n applications and the arbitrarily application is denoted as app_i. Suppose that each application has m features, and the arbitrarily feature of app_i is denoted as $af_{i,j}$. Let Fe denote the set with respect to n applications and m features, and we have

$$Fe = \left(af_{i,j}\right)_{n\times m} = \begin{bmatrix} af_{1,1} & af_{1,2} & \cdots & af_{1,m} \\ af_{2,1} & af_{2,2} & \cdots & af_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ af_{n,1} & af_{n,2} & \cdots & af_{n,m} \end{bmatrix}. \tag{1}$$

In this paper, we use the min-max method to do normalization for $af_{i,j}$. Mathematically, we have equation (2).

Let $Fe^{'}$ denote the normalized set regarding n applications and m features, and the values of $n \times m$ features are between 0 and 1, i.e., $af^{'}_{i,j} \in (0, 1)$.

D. CNN-BASED APPLICATION-AWARENESS

CNN consists of many hidden layers with two distinguished advantages, i.e., local connection and parameters sharing. In

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3021185, IEEE Access

this section, we use CNN to do the SDN-based application-awareness. For the used CNN model, it includes four major parts, i.e., activation function, pooling function, classification function and loss function.

1) Activation function

In order to improve the computation efficiency, the ReLU is exploited as the activation function. In particular, ReLU-based activation function can effectively solve the vanishing gradient problem. Let $ln_{i,j}$ denote the arbitrarily neuron which belongs to the i-th layer (denoted as $layer_i$) and the j-th neuron, and we have

$$s(ln_{i,j}) = \begin{cases} 0, \ ln_{i.j} \leq 0 \\ ln_{i,j}, \ ln_{ij} > 0 \end{cases} \tag{3}$$

where $s(ln_{i,j})$ is the status of $ln_{i,j}$. When the stimulus intensity reaches some level, $ln_{i,j}$ is activated. Furthermore, the output result of $ln_{i,j}$ after using the activation function is 0, which means that $ln_{i,j}$ is not active.

2) Pooling function

The traffic features extracted from the convolution layers are the high-dimensional data, which increases the complexity of computation. Thus, it requires to find a method as the pooling function for the dimensionality reduction. In order to avoid the more complex computation, this paper exploits the t-SNE [32] rather than the Maxout function (involving too many parameters) [33] for it.

Let $af^{'}_{i,j}(1)$ and $af^{'}_{i,j}(2)$ denote arbitrary two points in terms of the high-dimensional space, and the correspondingly mapped points to the low-dimensional space are denoted as $af^{''}_{i,j}(1)$ and $af^{''}_{i,j}(2)$, and we have the conditional probability with respect to $af^{'}_{i,j}(1)$ and $af^{'}_{i,j}(2)$ in the high-dimensional space, as follows.

$$hpc_{2|1} = \frac{exp(-hd^2_{1,2}/2\sigma^2)}{\sum_{k\neq 1} exp(-hd^2_{1,k}/2\sigma^2)} \tag{4}$$

$$hd_{1,2} = \| \ af^{'}_{i,j}(1) - af^{'}_{i,j}(2) \ \| \tag{5}$$

Furthermore, we can obtain the joint probability distribution function on $af^{'}_{i,j}(1)$ and $af^{'}_{i,j}(2)$ in the high-dimensional space, as follows.

$$hjpd_{1,2} = \frac{hpc_{1|2} + hpc_{2|1}}{n \times m} \tag{6}$$

By using the law of t-distribution, and we can obtain the joint probability distribution function on $af^{''}_{i,j}(1)$ and $af^{''}_{i,j}(2)$ in the low-dimensional space, as follows.

$$ljpd_{1,2} = \frac{(1 + ld^2_{1,2})^{-1}}{\sum_{k\neq 1}(1 + ld^2_{1,k})^{-1}} \tag{7}$$

where $ld_{1,2}$ is the Euclidean distance between $af^{''}_{i,j}(1)$ and $af^{''}_{i,j}(2)$. Let $Cost$ denote the difference between $hjpd_{1,2}$ and $ljpd_{1,2}$, and we have

$$Cost = \sum\sum hjpd_{1,2} log \frac{hjpd_{1,2}}{ljpd_{1,2}} \tag{8}$$

where the involved gradient training is defined as follows.

$$\frac{\delta Cost}{\delta af^{''}_{i,j}(1)} = 4 \sum_{1\neq 2} ld_{1,2}(hjpd_{1,2} - ljpd_{1,2})(1 + ld^2_{1,2})^{-1} \tag{9}$$

Let $RFe^{'}$ denote the reduced result of dimensionality from $Fe^{'}$, and we have the iteration equation (10). Among them, I, η and α are the number of iterations, learning rate and momentum factor respectively. In particular, the three parameters can guarantee the convergence speed and avoid to fall into the local optimum.

3) Classification function

The classifier is used for the result output and it is very significant in terms of the design of CNN. In this paper, we use the Softmax [34] as the function of output layer, because it can effectively handle the multi-classification problem by the modelling for the multinomial distribution. In addition, it has a distinguished advantage, that is, the correct classification has higher probability while the error classification has lower probability. The whole process on using the Softmax function is depicted in Figure 3.

Among them, $w_{i,j}$ denote the weight of $af_{i,j}$, and p_i denotes the prediction probability for app_i, and it is defined as follows.

$$p_i = \frac{e^{z_i}}{\sum_{j=1}^{nm} e^{z_j}} \tag{11}$$

Furthermore, we have $p_i \in [0, 1]$ and the following equation is satisfied.

$$\sum_{i=1}^{nm} p_i = 1 \tag{12}$$

4) Loss function

What the loss function exits means that it has the absolute difference between the real value and the prediction value via the Softmax function. Meanwhile, the smaller loss function value means the more accurate application-awareness. Given this consideration, it requires to use the gradient descent algorithm [35] to update and optimize these involved parameters in order to minimize the loss function value. Let $loss$ denote the general loss function, and we have

$$af^{'}_{i,j} = \frac{af_{i,j} - min\{af_{i,1}, af_{i,2}, \cdots, af_{i,m}\}}{max\{af_{i,1}, af_{i,2}, \cdots, af_{i,m}\} - min\{af_{i,1}, af_{i,2}, \cdots, af_{i,m}\}} \tag{2}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3021185, IEEE Access

$$loss = \sum_{k=1}^{K} \Big(\lambda_k ln p_k + (1 - \lambda_k) ln(1 - p_k) \Big) \qquad (13)$$

where K is the total number of samples, and $\lambda_k \in \{0, 1\}$ is used to mark the status of the k-th sample.

E. K-FOLD CROSS VALIDATION

In order to take maximum advantage of CNN, it is necessary to explore more useful information from the limited dateset (i.e., the number of samples). In this paper, we use the K-CV method [36] for it, which can effectively decrease the generalization error and prevent the over-fitting. To be specific, the dataset is divided into K sets randomly, and each set is regarded as a sample. For the K sets, $K - 1$ ones are regarded as the training samples and the remaining one is regarded as the test sample. The cross validation is performed for K times until each set has been regarded as the test sample. On this basis, we can obtain the K accuracy degrees on CNN-based application-awareness, and the corresponding average value is considered as the accuracy degree of CNN.

IV. EVALUATION PERFORMANCE

A. ENVIRONMENT INSTALL

The proposed CDSA consists of two parts: one is the SDN-based traffic collection and the other one is CNN-based application-awareness. For the first one, the SDN environment is implemented based on the MiniNet; for the second one, the simulation experiments are made based on the TensorFlow. The hardware environment is Intel(R), Core(TM)i7, CPU870 2.93GHz, 4.00RAM. This paper uses the GPU provided by Google to do the training for features modelling. To be specific, the dataset is submitted to the cloud storage of Google firstly; then, the successfully debugged application programs are inserted into the development tool, i.e., Colaboratory; finally, the TensorFlow is started for training.

In addition, the proposed CDSA is compared with three advanced schemes, i.e., [22], [23] and [27] which have been published in International Conference on Network Protocols (ICNP), on Advanced information networking and Applications (ICAA), and on Communication software and Networks (ICCN) respectively. Meanwhile, three classical factors, i.e., recall ratio, precision ratio and F value as well as a novel factor, i.e., stability are considered the metrics of performance evaluation. Furthermore, for the involved parameters, many groups of simulations under different settings are made and the most suitable combination is determined as follows. $K = 10$, $\alpha = 0.5$, $I = 80$ and $\eta = 0.42$.

TABLE 2. The details of 12 applications.

No.	Application Type	The Number of Items
1	WWW	328091
2	MAIL	28567
3	FTP-CONTROL	3054
4	FTP-PASV	2688
5	ATTACK	1793
6	P2P	2094
7	DATABASE	2648
8	FTP-DATA	5797
9	MULTIMEDIA	576
10	SERVICES	2099
11	INTERACTIVE	110
12	GAMES	8

B. DATASET INTRODUCTION

This paper adopts an open Moore dataset [37]-[38] which has 24 hours duration of traffic collection from the University of Cambridge to verify the proposed CDSA. The dataset includes 12 different applications and 370000 items, and each application type owns 248 features. For these applications, their details are shown in Table 2.

As can be seen from Table 2, the number of items regarding GAMES or INTERACTIVE application type (i.e., the last two applications) is very small, thus this paper only considers to use the first ten application types as these samples for verification. In addition, in order to conveniently facilitate the comparability between the real value and the prediction value, this paper uses the one-hot coding method

$$RFe^{'}(I) = RFe^{'}(I-1) + \eta \frac{Cost}{RFe^{'}} + \alpha\big(RFe^{'}(I-1) - RFe^{'}(I-2)\big) \qquad (10)$$

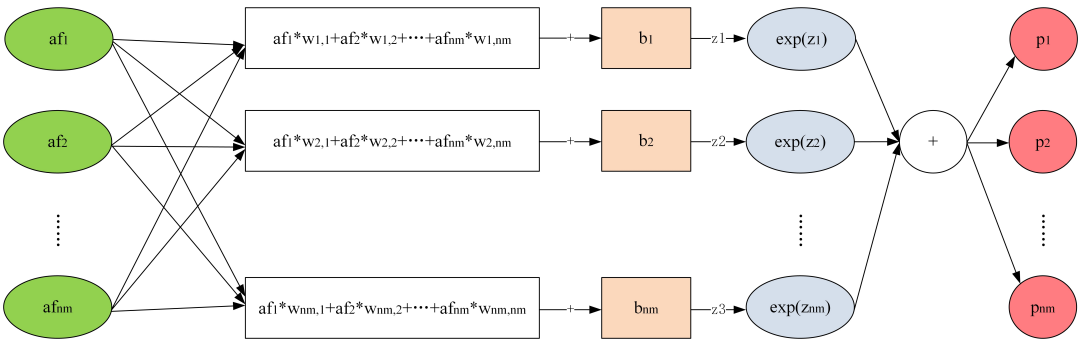


FIGURE 3. The whole process on using the Softmax function.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3021185, IEEE Access

TABLE 3. The marked results for all applications based on one-hot coding.

Application Type	Coding
ATTACK	(1,0,0,0,0,0,0,0,0,0)
DATABASE	(0,1,0,0,0,0,0,0,0,0)
FTP-CONTROL	(0,0,1,0,0,0,0,0,0,0)
FTP-DATA	(0,0,0,1,0,0,0,0,0,0)
FTP-PASV	(0,0,0,0,1,0,0,0,0,0)
MAIL	(0,0,0,0,0,1,0,0,0,0)
MULTIMEDIA	(0,0,0,0,0,0,1,0,0,0)
P2P	(0,0,0,0,0,0,0,1,0,0)
SERVICES	(0,0,0,0,0,0,0,0,1,0)
WWW	(0,0,0,0,0,0,0,0,0,1)

to quantify the application, that is to say, for some application, its location is marked as 1 while the others are marked as 0. On this basis, the marked results for all applications are shown in Table 3.

C. CNN STRUCTURE DETERMINATION

This section determines the structure of CNN. In order to decrease the number of parameters and the training time, the convolution kernel is set as 3×3, and the length of input matrix is 16, where the moving step lengths of convolution layer and pooling layer are set as 1 and 2 respectively. Given this, we provide 10 different CNN structures, and the detailed structure information is shown in Table 4.

For the 10 different CNN structures, we compute the corresponding whole accuracies, and the computation results are shown in Table 5, where the whole accuracy is defined as the ratio between the number of applications that have been perceived correctly and the total number of applications.

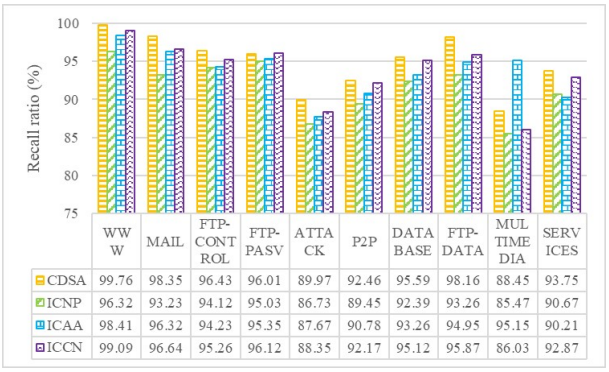
We can observe that the fifth CNN structure (i.e., S5) can contribute to the best whole accuracy. By combining with Table 4, we can future observe that, with the increasing of convolution kernels and weight parameters, the corresponding whole accuracy becomes to ascend. However, when reaching a certain level, the corresponding whole accuracy begins to descend, which is caused by the increased computation complexity. Furthermore, it indicates that the modest increasing on the number of convolution kernels for CNN model can be conducive to the whole accuracy in terms of the SDN-based application-awareness. Here, we emphasize that the following experimental results on the proposed CDSA are reported according to the S5-based CNN structure.

D. COMPARISON ANALYSIS

1) Recall ratio

Four different methods (i.e., CDSA, ICNP, ICAA and ICCN) under different application types for doing the SDN-based application-awareness, the corresponding results on recall ratio are reported in Figure 4.

We find that the proposed CDSA has the highest recall ratio, this is because CDSA uses the CNN model. To be specific, CNN has the multiple hidden layers, which can explore the implying information. Especially, the explored information can fully express the application feature which contributes to the whole accuracy and at the same time



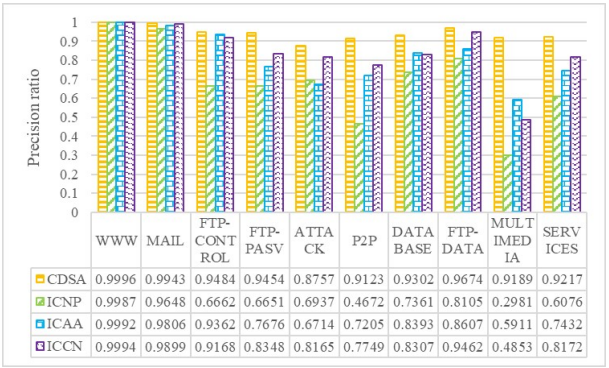
	WWW	MAIL	FTP-CONTROL	FTP-PASV	ATTACK	P2P	DATABASE	FTP-DATA	MULTIMEDIA	SERVICES
CDSA	99.76	98.35	96.43	96.01	89.97	92.46	95.59	98.16	88.45	93.75
ICNP	96.32	93.23	94.12	95.03	86.73	89.45	92.39	93.26	85.47	90.67
ICAA	98.41	96.32	94.23	95.35	87.67	90.78	93.26	94.95	95.15	90.21
ICCN	99.09	96.64	95.26	96.12	88.35	92.17	95.12	95.87	86.03	92.87

FIGURE 4. The recall ratio among CDSA, ICNP, ICAA and ICCN.

guarantees the recall ratio. Besides, we also find that ATTACK and MULTIMEDIA have lower recall ratio than the other application types, because their inclusive number of applications is smaller than that of the others. For these samples with the insufficient data items, it is very difficult to train the unique and correct application feature. In terms of the other three methods, ICNP has the lowest recall ratio because it is only deployed at an enterprise network and it adopts the stochastic gradient boosting and extreme gradient boosting, which cannot guarantee the universal recall ratio. Compared to ICAA, ICCN has higher recall ratio because it can automatically analyze and translate an arbitrary performance metric into a comprehensible application feature.

2) Precision ratio

This section reports the experimental results on precision ratio with respect to CDSA, ICNP, ICAA and ICCN under different application types for doing the SDN-based application-awareness, as shown in Figure 5.



	WWW	MAIL	FTP-CONTROL	FTP-PASV	ATTACK	P2P	DATABASE	FTP-DATA	MULTIMEDIA	SERVICES
CDSA	0.9996	0.9943	0.9484	0.9454	0.8757	0.9123	0.9302	0.9674	0.9189	0.9217
ICNP	0.9987	0.9648	0.6662	0.6651	0.6937	0.4672	0.7361	0.8105	0.2981	0.6076
ICAA	0.9992	0.9806	0.9362	0.7676	0.6714	0.7205	0.8393	0.8607	0.5911	0.7432
ICCN	0.9994	0.9899	0.9168	0.8348	0.8165	0.7749	0.8307	0.9462	0.4853	0.8172

FIGURE 5. The precision ratio among CDSA, ICNP, ICAA and ICCN.

It is obvious that the proposed CDSA can obtain the highest precision ratio, and two related reasons are analyzed as follows. On one hand, during the process of CNN computation, the t-SNE is considered as the pooling function for the dimensionality reduction, greatly avoiding the complex and excessive computation on the redundant application features. On the other hand, the classifier is increased a deviator to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.3021185, IEEE Access

TABLE 4. The detailed CNN structure information.

No.	Input Layer	Convolution Layer (Kernel)	Pooling Layer (Filter window)	Convolution Layer (Kernel)	Pooling Layer (Filter window)	Fully Connected Layer (Weights)	Output Layer
S1	16×16	$5 \times 3 \times 3$	2×2	$10 \times 3 \times 3$	2×2	$256 \times 10 \times 4 \times 4$	1×10
S2	16×16	$10 \times 3 \times 3$	2×2	$15 \times 3 \times 3$	2×2	$256 \times 15 \times 4 \times 4$	1×10
S3	16×16	$15 \times 3 \times 3$	2×2	$20 \times 3 \times 3$	2×2	$256 \times 20 \times 4 \times 4$	1×10
S4	16×16	$20 \times 3 \times 3$	2×2	$25 \times 3 \times 3$	2×2	$256 \times 25 \times 4 \times 4$	1×10
S5	16×16	$25 \times 3 \times 3$	2×2	$30 \times 3 \times 3$	2×2	$256 \times 30 \times 4 \times 4$	1×10
S6	16×16	$30 \times 3 \times 3$	2×2	$35 \times 3 \times 3$	2×2	$256 \times 35 \times 4 \times 4$	1×10
S7	16×16	$35 \times 3 \times 3$	2×2	$40 \times 3 \times 3$	2×2	$256 \times 40 \times 4 \times 4$	1×10
S8	16×16	$40 \times 3 \times 3$	2×2	$45 \times 3 \times 3$	2×2	$256 \times 45 \times 4 \times 4$	1×10
S9	16×16	$45 \times 3 \times 3$	2×2	$50 \times 3 \times 3$	2×2	$256 \times 50 \times 4 \times 4$	1×10
S10	16×16	$50 \times 3 \times 3$	2×2	$55 \times 3 \times 3$	2×2	$256 \times 55 \times 4 \times 4$	1×10

TABLE 5. The whole accuracies of different CNN structures.

No.	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
The whole accuracy	0.7831	0.8024	0.8316	0.8632	0.9426	0.9185	0.9023	0.8714	0.8506	0.8213

adjust the difference between the real value and the prediction value during the process of engineering implementation; under such condition, the adopted gradient descent algorithm can update and optimize these involved parameters easily. In addition, similar to the recall ratio, the two special application types, i.e., ATTACK and MULTIMEDIA also have lower precision ratio than the others. Furthermore, for the other three application-awareness methods, ICCN has the highest precision ratio because it expands the SDN to enable each application to have a customized performance-based metric, which can greatly improve the feature recognition and obtain the relatively high precision ratio. Moreover, different from ICNP, ICAA establishes the arrival interval of elephant flows and filters out the redundant samples, which can guarantee the precision ratio. Thus, ICAA has higher precision ratio than ICNP.

3) F value

In this section, we test F value which is computed by

$$F_\beta = \frac{(\beta^2 + 1) * ReR_i * PreR_i}{\beta^2 * (ReR_i + PreR_i)} \tag{14}$$

where ReR_i and $PreR_i$ are the recall ratio and the precision ratio with respect to app_i respectively. In addition, β is a parameter which is set as 1, 2, 3 and 4 in this section. The experimental results on $F1$, $F2$, $F3$ and $F4$ are reported in Figure 6, Figure 7, Figure 8 and Figure 9 respectively.

We can see that the corresponding F value becomes smaller and smaller with the increasing of β. In general, the testing on $F1$ value is the most valuable reference, and the larger $F1$ value means that the involved method is more efficient. On this basis, we can find that the proposed CDSA has the largest $F1$ value, which indicates that CDSA is the best mechanism for the SDN-based application-awareness. The related reasons can be found in the above two sections.

4) Stability

The stability is a metric which reflects the effectiveness and robustness of an application-awareness method. In this

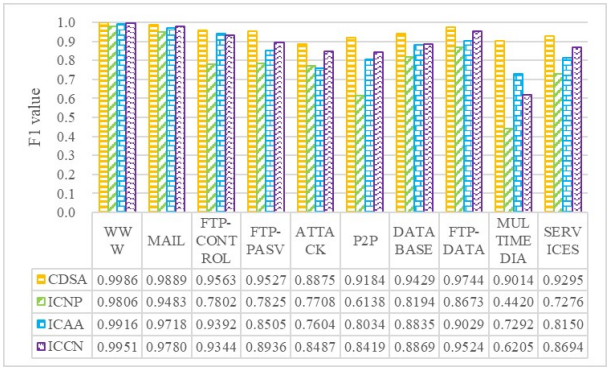


FIGURE 6. The $F1$ value among CDSA, ICNP, ICAA and ICCN.

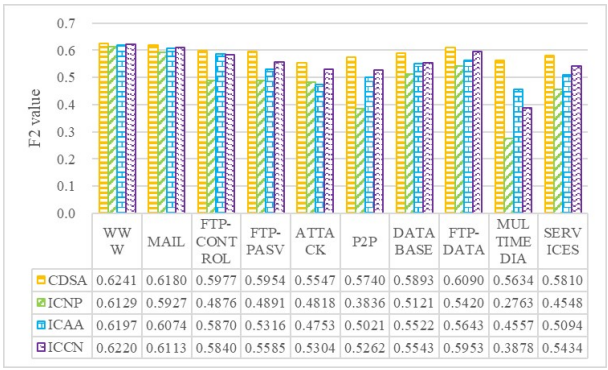
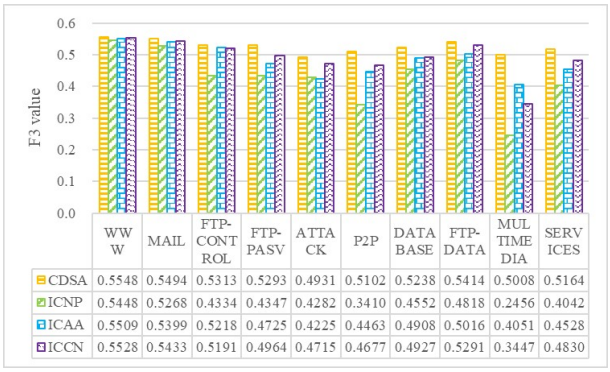


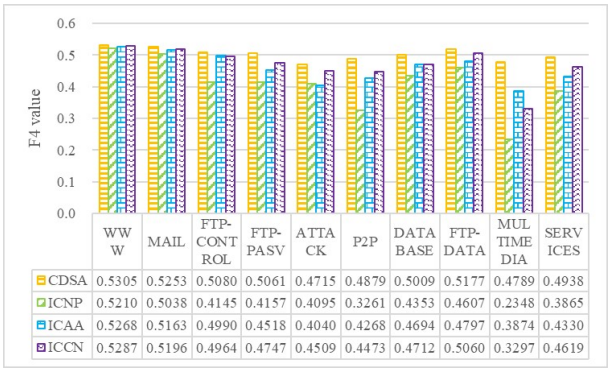
FIGURE 7. The $F2$ value among CDSA, ICNP, ICAA and ICCN.

paper, it is quantified as the fluctuation coefficient [39] among CDSA, ICNP, ICAA and ICCN with respect to recall ratio, precision ratio and $F1$ value. Furthermore, the small fluctuation coefficient means that the application-awareness mechanism has good performance. As illustrated in Table 6, the stability with respect to CDSA, ICNP, ICAA and ICCN is reported. We can observe that the proposed CDSA has the smallest fluctuation coefficient, which further suggests that it can be accepted as the best mechanism to do the SDN-based application-awareness.



	WWW	MAIL	FTP-CONTROL	FTP-PASV	ATTACK	P2P	DATABASE	FTP-DATA	MULTIMEDIA	SERVICES
CDSA	0.5548	0.5494	0.5313	0.5293	0.4931	0.5102	0.5238	0.5414	0.5008	0.5164
ICNP	0.5448	0.5268	0.4334	0.4347	0.4282	0.3410	0.4552	0.4818	0.2456	0.4042
ICAA	0.5509	0.5399	0.5218	0.4725	0.4225	0.4463	0.4908	0.5016	0.4051	0.4528
ICCN	0.5528	0.5433	0.5191	0.4964	0.4715	0.4677	0.4927	0.5291	0.3447	0.4830

FIGURE 8. The $F3$ value among CDSA, ICNP, ICAA and ICCN.



	WWW	MAIL	FTP-CONTROL	FTP-PASV	ATTACK	P2P	DATABASE	FTP-DATA	MULTIMEDIA	SERVICES
CDSA	0.5305	0.5253	0.5080	0.5061	0.4715	0.4879	0.5009	0.5177	0.4789	0.4938
ICNP	0.5210	0.5038	0.4145	0.4157	0.4095	0.3261	0.4353	0.4607	0.2348	0.3865
ICAA	0.5268	0.5163	0.4990	0.4518	0.4040	0.4268	0.4694	0.4797	0.3874	0.4330
ICCN	0.5287	0.5196	0.4964	0.4747	0.4509	0.4473	0.4712	0.5060	0.3297	0.4619

FIGURE 9. The $F4$ value among CDSA, ICNP, ICAA and ICCN.

V. CONCLUSIONS

This paper proposes a CNN-based mechanism to do the SDN-based application-awareness because CNN has local connection and parameters sharing advantages, which includes three major modules, i.e., traffic collection, data preprocessing and application-awareness. Among them, for the first part, the general switch submits its collected traffic information to the OpenFlow switch according to the techniques of port mirroring and redirection. For the second part, the min-max method is exploited to do normalization for facilitating the features modelling. For the last part, it consists of ReLU-based activation function, t-SNE-based pooling function, Softmax-based classification function and loss function. Additionally, in order to take maximum advantage of CNN, the K-CV method is used to explore more useful information from the limited dateset. Based on the real Moore dataset, the experimental results demonstrate that the proposed mechanism has more efficient recall ratio, precision ratio, F value and stability compared to three baselines.

However, as a novel CCN-based application-awareness for SDN, the proposed CDSA also has two limitations that cannot be ignored. On one hand, the t-SNE is regarded as the pooling function for dimensionality reduction, which has the relatively high computation complexity and decreases the application-awareness speed to some extent. On the other hand, the adopted CCN structure model contains too many hidden layers, which needs to consume too much computation time and puts off the convergence speed. In the future, we plan to enhance this paper from the following three aspects. At first, we improve the t-SNE and try to decrease its computation complexity. Secondly, we study the CCN structure model and find a relatively optimal one for supporting the SDN-based application-awareness. At last, we will test more applications (except the current 10 kinds of applications in this paper) and more datasets to further prove the application value of the proposed CDSA.

TABLE 6. The stability (the small fluctuation coefficient means that the application-awareness mechanism has good performance).

Mechanism	CDSA	ICNP	ICAA	ICCN
Fluctuation coefficient	0.3158	0.6315	0.6054	0.4226

REFERENCES

[1] Karakus M, Durresi A, Quality of service (QoS) in software defined networking (SDN): a survey, Journal of Network and Computer Applications, 2017, 80: 200-218.

[2] El-hajj M, Fadlallah A, Chamoun M, et al, A survey of Internet of Things (IoT) authentication schemes, Sensors, 2019, 19(5): 1-43.

[3] Du R, Santi P, Xiao M, et al, The sensable city: a survey on the deployment and management for smart city monitoring, IEEE Communications Surveys & Tutorials, 2019, 21(2): 1533-1560.

[4] Deng S, Wu H, Hu D, et al, Service selection for composition with QoS correlations, IEEE Transactions on Services Computing, 2016, 9(2): 291-303.

[5] Benzekki K, Fergougui AE, Elalaoui AE, Software-defined networking (SDN): a survey, Security and Communication Networks, 2016, 9(18): 5803-5833.

[6] Zhang C, Wang X, Dong A, et al, Energy efficient network service deployment across multiple SDN domains, Computer Communications, 2020, 151, 449-462.

[7] Barakabitze AA, Ahmad A, Mijumbi R, et al, 5G network slicing using SDN and NFV: a survey of taxonomy, architectures and future challenges, Computer Networks, 2020, 167, 1-40.

[8] Kim H, Claffy KC, Fomenkov M, et al, Internet traffic classification demystified: myths, caveats, and the best practices, in Proc. of ACM CoNEXT, 2008: 1-12.

[9] Touch J, Lear E, Mankin A, Service name and transport protocol port number registry, www.iana.org, 2020.

[10] Haffner P, Sen S, Spatscheck O, et al, ACAS: automated construction of application signatures, in Proc. of ACM SIGCOMM Workshop on Mining Network Data, 2005: 197-202.

[11] Smith R, Estan C, Jha S, et al, Deflating the big bang: fast and scalable deep packet inspection with extended finite automata, ACM SIGCOMM Computer Communication Review, 2008, 38(4): 207-218.

[12] Li G, Dong M, Ota K, et al, Deep packet inspection based application-aware traffic control for software defined networks, in Proc. of IEEE International Conference on Global Communications, 2017: 1-6.

[13] Nguyen TTT, Armitage G, A survey of techniques for internet traffic classification using machine learning, IEEE Communications Surveys & Tutorials, 2009, 10(4): 56-76.

[14] Archanaa R, Athulya V, Rajasundari T, et al, A comparative performance analysis on network traffic classification using supervised learning algorithms, in Proc. of IEEE International Conference on Advanced Computing and Communication Systems, 2017: 1-5.

[15] Xiao P, Liu N, Li Y, et al, A traffic classification method with spectral clustering in SDN, in Proc. of International Conference on Parallel & Distributed Computing, 2016: 391-394.

[16] Bian X, PSO optimized semi-supervised network traffic classification strategy, in Proc. of IEEE International Conference on Intelligent Transportation, Big Data & Smart City, 2018: 179-182.

[17] Fadlullah Z, Tang F, Mao B, et al, State-of-the-art deep learning: evolving machine intelligence toward tomorrow's intelligent network traffic control

systems, IEEE Communications Surveys & Tutorials, 2017, 19(4): 2432-2455.

[18] Krizhevsky A, Sutskever I, Hinton G, Imagenet classification with deep convolutional neural networks, in Proc. of International Conference on Neural Information Processing Systems, 2012: 1097-1105.

[19] Mekky H, Hao F, Mukherjee S, et al, Application-aware data plane processing in SDN, in Proc. of HotSDN, 2014: 13-18.

[20] Sanvito D, Moro D, Capone A, Towards traffic classification offloading to stateful SDN data planes, in Proc. of IEEE Conference on Network Softwarization, 2017: 1-4.

[21] Hu L, Xu C, Luo Y, vTC: machine learning based traffic classification as a virtual network function, in Proc. of SDN-NFVSec, 2016: 53-56.

[22] Amaral P, Dinis J, Pinto P, et al, Machine learning in software defined networks: Data collection and traffic classification, in Proc. of IEEE International Conference on Network Protocols, 2016: 1-5.

[23] Tang F, Li L, Barolli L, et al, An efficient sampling and classification approach for flow detection in SDN-Based big data centers, in Proc. of IEEE International Conference on Advanced Information Networking and Applications, 2017: 1106-1115.

[24] Liu C, Chang Y, Tseng C, et al, SVM-based classification mechanism and its application in SDN networks, in Proc. of IEEE International Conference on Communication Software and Networks, 2018: 45-49.

[25] Xu J, Wang J, Qi Q, et al, Deep neural networks for application awareness in SDN-based network, in Proc. of IEEE International Workshop on Machine Learning for Signal Processing, 2018: 1-6.

[26] Amaral P, Pinto PF, Bernardo L, et al, Application aware SDN architecture using semi-supervised traffic classification, in Proc. of IEEE Conference on Network Function Virtualization and Software Defined Networks, 2018: 1-6.

[27] Jahromi HZ, Delaney DT, An application awareness framework based on SDN and machine learning: defining the roadmap and challenges, in Proc. of International Conference on Communication Software and Networks, 2018: 411-416.

[28] Li Y, Li J, MultiClassifier: a combination of DPI and ML for application-layer classification in SDN, in Proc. of International Conference on Systems and Informatics, 2014: 682-686.

[29] Wang P, Lin S, Luo M, A framework for QoS-aware traffic classification using semi-supervised machine learning in SDNs, in Proc. of IEEE International Conference on Services Computing, 2016: 760-765.

[30] Yu C, Lan J, Xie J, et al, QoS-aware traffic classification architecture using machine learning and deep packet inspection in SDNs, Procedia Computer Science, 2018, 131: 1209-1216.

[31] TensorFlow, https://tensorflow.google.cn.

[32] Maaten L, Hinton G, Visualizing data using t-SNE, Journal of Machine Learning Research, 2008, 9: 2579-2605.

[33] Goodfellow IJ, Warde-Farley D, Mirza M, et al, Maxout networks, in Proc. of International Conference on Machine Learning, 2013: 1-9.

[34] Jiang M, Liang Y, Feng X, et al, Text classification based on deep belief network and softmax regression, Neural Computing and Applications, 2018, 29(1): 61-70.

[35] Cohen K, Nedic A, Srikant R, On projected stochastic gradient descent algorithm with weighted averaging for least squares regression, IEEE Transactions on Automatic Control, 2017, 62(11): 5974-5981.

[36] Wong T, Yang N, Dependency analysis of accuracy estimates in k-fold cross validation, IEEE Transactions on Knowledge & Data Engineering, 2017, 29(1): 2417-2427.

[37] Chen L, Paul H, Qu H, et al, Correntropy-based robust multilayer extreme learning machines, Pattern Recognition, 2018, 84, 357-370.

[38] Ren Z, Yang L, Correntropy-based robust extreme learning machine for classification, Neurocomputing, 2018, 313: 74-84.

[39] Lv J, Wang X, Huang M, et al, RISC: ICN routing mechanism incorporating SDN and community division, Computer Networks, 2017, 123: 88-103.