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

NAN HU1, FANGJUN LUAN1, XIAOXI TIAN1, AND CHENGDONG WU2
1Information & Control Engineering Faculty, Shenyang Jianzhu University, Shenyang 110168, China
2Faculty of Robot Science and Engineering, Northeastern University, Shenyang 110819, China
Corresponding author: Nan Hu (zerovanila@sina.com)

This work was supported in part by the National Key Research and Development Project under Grant 2017YBF1300900 and in part by the National Natural Science Foundation of China under Grant 61973063.

ABSTRACT With the rapid development of the Internet of Things (IoT) and smart cities, more and more types of applications have been emerging. 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’ 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 statistic 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]–[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
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 99.05% 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 |

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, 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.

**C. DATA PRE-PROCESSING**

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

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

(1)

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

\[
af_{i,j}' = \frac{af_{i,j} - \min\{af_{i,1}, af_{i,2}, \cdots, af_{i,m}\}}{\max\{af_{i,1}, af_{i,2}, \cdots, af_{i,m}\} - \min\{af_{i,1}, af_{i,2}, \cdots, af_{i,m}\}}
\]  

(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 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 \( l_n_{i,j} \) denote the arbitrarily neuron which belongs to the \( i \)-th layer (denoted as \( \text{layer}_i \)) and the \( j \)-th neuron, and we have

\[
s(l_n_{i,j}) = \begin{cases} 0, & l_n_{i,j} \leq 0 \\ l_n_{i,j}, & l_n_{i,j} > 0 \end{cases}
\]

where \( s(l_n_{i,j}) \) is the status of \( l_n_{i,j} \). When the stimulus intensity reaches some level, \( l_n_{i,j} \) is activated. Furthermore, the output result of \( l_n_{i,j} \) after using the activation function is 0, which means that \( l_n_{i,j} \) 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 \( af_{i,j}^{(1)}(1) \) and \( af_{i,j}^{(2)}(2) \), and we have the conditional probability with respect to \( af_{i,j}^{(1)}(1) \) and \( af_{i,j}^{(2)}(2) \) in the high-dimensional space, as follows.

\[
hpc_{2|1} = \frac{\exp(-hd_{1,2}^2/2\sigma^2)}{\sum_{k \neq 1} \exp(-hd_{1,k}^2/2\sigma^2)} \tag{4}
\]

\[
hd_{1,2} = \| af_{i,j}^{(1)}(1) - af_{i,j}^{(2)}(2) \| \tag{5}
\]

Furthermore, we can obtain the joint probability distribution function on \( af_{i,j}^{(1)}(1) \) and \( af_{i,j}^{(2)}(2) \) 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 \( af_{i,j}^{(1)}(1) \) and \( af_{i,j}^{(2)}(2) \) in the low-dimensional space, as follows.

\[
ljp_{1,2} = \frac{(1 + ld_{1,2}^2)^{-1}}{\sum_{k \neq 1} (1 + ld_{1,k}^2)^{-1}} \tag{7}
\]

where \( ld_{1,2} \) is the Euclidean distance between \( af_{i,j}^{(1)}(1) \) and \( af_{i,j}^{(2)}(2) \). Let \( Cost \) denote the difference between \( hjpd_{1,2} \) and \( ljp_{1,2} \), and we have

\[
Cost = \sum \sum hjpd_{1,2} \log \frac{hjpd_{1,2}}{ljp_{1,2}} \tag{8}
\]

where the involved gradient training is defined as follows.

\[
\frac{\delta Cost}{\delta af_{i,j}^{(1)}(1)} = 4 \sum_{l \neq 2} ld_{1,2}(hjpd_{1,2} - ljp_{1,2})(1 + ld_{1,2}^2)^{-1} \tag{9}
\]

Let \( RFe' \) denote the reduced result of dimensionality from \( Fe' \), and we have the iteration equation (10). Among them, \( I, \eta \) and \( \alpha \) 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.

\[
RFe'(1) = RFe'(I - 1) + \eta \frac{Cost}{RFe'} + \alpha(RFe'(I - 1) - RFe'(I - 2)) \tag{10}
\]

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.

\[
p_i = \frac{e^{p_i}}{\sum_{j=1}^{m} e^{p_j}} \tag{11}
\]

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

\[
\sum_{i=1}^{m} p_i = 1 \tag{12}
\]

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

\[
loss = \sum_{k=1}^{K} \left( \lambda_k \ln p_k + (1 - \lambda_k) \ln (1 - p_k) \right) \tag{13}
\]

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 dataset (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 \( \eta = 0.42 \).

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.

| No. | Application Type | The Number of Items |
|-----|------------------|---------------------|
| 1   | WWW              | 328091              |
| 2   | MAIL             | 28567               |
| 3   | FTP-CONTROL     | 3094                |
| 4   | FTP-PASSV        | 2688                |
| 5   | ATTACK           | 1793                |
| 6   | ZIP              | 2094                |
| 7   | DATABASE         | 2648                |
| 8   | FTP-DATA         | 5797                |
| 9   | MULTIMEDIA       | 576                 |
| 10  | SERVICES         | 2099                |
| 11  | INTERACTIVE      | 110                 |
| 12  | GAMES            | 8                   |

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 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.

| Coding | Application Type |
|--------|------------------|
| 1.0    | ATTACK           |
| 0.1    | DATABASE         |
| 0.01   | FTP-CONTROL     |
| 0.0    | FTP-PASSV        |
| 0.001  | FTP-DATA         |
| 0.0    | MAIL             |
| 0.001  | MULTIMEDIA       |
| 0.00   | ZIP              |
| 0.0001 | SERVICES         |
| 0.000  | WWW              |
### 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 $3 \times 3$, 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 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.

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 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

$$F_\beta = \frac{(\beta^2 + 1) \times ReR_i \times PreR_i}{\beta^2 \times (ReR_i + PreR_i)}$$  \hspace{1cm} (14)

where $ReR_i$ and $PreR_i$ are the recall ratio and the precision ratio with respect to $app_i$ respectively. In addition, $\beta$ is a parameter which is set as 1, 2, 3 and 4 in this section. The experimental results on $F_1$, $F_2$, $F_3$ and $F_4$ 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 $\beta$. In general, the testing on $F_1$ value is the most valuable reference, and the larger $F_1$ value means that the involved method is more efficient. On this basis, we can find that the proposed CDSA has the largest $F_1$ 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 paper, it is quantified as the fluctuation coefficient [39] among CDSA, ICNP, ICAA and ICCN with respect to recall ratio, precision ratio and $F_1$ 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

| Mechanism | CDSA | ICNP | ICAA | ICCN |
|-----------|------|------|------|------|
| Fluctuation coefficient | 0.3158 | 0.6315 | 0.6054 | 0.4226 |

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

FIGURE 6. The $F_1$ value among CDSA, ICNP, ICAA and ICCN.

FIGURE 7. The $F_2$ value among CDSA, ICNP, ICAA and ICCN.

FIGURE 8. The $F_3$ value among 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.

V. CONCLUSION

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 pre-processing 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 dataset. 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 performance of CNN. Additionally, in order to take maximum advantage of CNN, the K-CV method is used to explore more useful information from the limited dataset. 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 performance of CNN. Additionally, in order to take maximum advantage of CNN, the K-CV method is used to explore more useful information from the limited dataset. 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.

Moreover, with the improved performance, the proposed CDSA can be accepted as the best mechanism to do the SDN-based application-awareness.

REFERENCES

[1] M. Karakus and A. Durresi, “Quality of service (QoS) in software defined networking (SDN): A survey,” J. Netw. Comput. Appl., vol. 80, pp. 200–218, Feb. 2017.
[2] M. El-hajj, A. Fadlallah, M. Chamoun, and A. Serhrouchni, “A survey of Internet of Things (IoT) authentication schemes,” Sensors, vol. 19, no. 5, p. 1141, Mar. 2019.
[3] R. Du, P. Santi, M. Xiao, A. V. Vasilakos, and C. Fischione, “The sensible city: A survey on the deployment and management for smart city monitoring,” IEEE Commun. Surveys Tuts., vol. 21, no. 2, pp. 1533–1560, 2nd Quart., 2019.
[4] S. Deng, H. Wu, D. Hu, and J. Leon Zhao, “Service selection for composition with QoS correlations,” IEEE Trans. Services Comput., vol. 9, no. 2, pp. 291–303, Mar. 2016.
[5] K. Benzekki, A. El Fergougui, and A. Elbelhiti Elaouni, “Software-defined networking (SDN): A survey,” Secur. Commun. Netw., vol. 9, no. 18, pp. 5803–5833, Dec. 2016.
[6] C. Zhang, X. Wang, A. Dong, Y. Zhao, Q. He, and M. Huang, “Energy efficient network service deployment across multiple SDN domains,” Comput. Commun., vol. 151, pp. 449–462, Feb. 2020.
[7] A. A. Barakabizite, A. Ahmad, R. Mijumbi, and A. Hines, “5G network slicing using SDN and NFV: A survey of taxonomy, architectures and future challenges,” Comput. Netw., vol. 167, Feb. 2020, Art. no. 106984.
[8] 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), 2008, pp. 1–12.
[9] J. Touch, E. Lear, and A. Mankin. (2020). Service Name and Transport Protocol Port Number Registry. [Online]. Available: https://www.iana.org
[10] P. Haffner, S. Sen, O. Spatscheck, and D. Wang, “ACAS: Automated construction of application signatures,” in Proc. ACM SIGCOMM Workshop Mining Nett. Data (MineNet), 2005, pp. 197–202.
[11] R. Smith, C. Estan, S. Jha, and S. Kong, “Deflating the big bang: Fast and scalable deep packet inspection with extended finite automata,” ACM SIGCOMM Comput. Commun. Rev., vol. 38, no. 4, pp. 207–218, 2008.
[12] 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.
[13] 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.
[14] R. Archanana, V. Athulya, T. Rajasundari, and M. V. K. Kiran, “A comparative performance analysis on network traffic classification using supervised learning algorithms,” in Proc. 4th Int. Conf. Adv. Comput. Commun. Syst. (ICACCS), Jan. 2017, pp. 1–5.
[15] P. Xiao, N. Liu, Y. Li, Y. Lu, X. J. Tang, H. W. Wang, and M. X. Li, “A traffic classification method with spectral clustering in SDN,” in Proc. Int. Conf. Parallel Distrib. Comput., 2016, pp. 391–394.
[16] X. Bian, “PSO optimized semi-supervised network traffic classification strategy,” in Proc. Int. Conf. Intell. Transp., Big Data Smart City (ICTRBS), Jan. 2018, pp. 179–182.
[17] Z. M. Fadlullah, F. Tang, B. Mao, N. Kato, O. Akashi, T. Inoue, and K. Mizutani, “State-of-the-Art deep learning: Evolving machine intelligence toward tomorrow’s intelligent network traffic control systems,” IEEE Commun. Surveys Tuts., vol. 21, no. 2, pp. 1533–1560, 4th Quart., 2017.
[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in Proc. Adv. Neural Inf. Process. Syst. (NIPS), 2012, pp. 1097–1105.
[19] H. Meiky, F. Hao, S. Mukherjee, Z.-L. Zhang, and T. V. Lakshman, “Application-aware data plane processing in SDN,” in Proc. 3rd Workshop Hot Topics Softw. Defin. Netw. (HotSDN), 2014, pp. 13–18.
[20] D. Sanvito, D. Moro, and A. Capone, “Towards traffic classification offloading to stateful SDN data planes,” in Proc. IEEE Int. Conf. Network Softw. Virtualization (NetSoft), Jul. 2017, pp. 1–4.
[21] L. He, C. Xu, and Y. Luo, “VTC: Machine learning based traffic classification as a virtual network function,” in Proc. ACM Work. Secur. Softw. Defin. Netw. Netw. Function Virtualization SDN-NFV Secur., 2016, pp. 53–56.
[22] P. Amaral, J. Dinis, P. Pinto, L. Bernardo, J. Tavares, and H. S. Mamede, “Machine learning in software defined networks: Data collection and traffic classification,” in Proc. IEEE 24th Int. Conf. Netw. Protocols (ICNP), Nov. 2016, pp. 1–5.
[23] F. Tang, L. Li, L. Barolli, and C. Tang, “An efficient sampling and classification approach for flow detection in SDN-based big data centers,” in Proc. IEEE 31st Int. Conf. Adv. Inf. Netw. Appl. (AINA), Mar. 2017, pp. 1106–1115.

[24] C.-C. Liu, Y. Chang, C.-W. Tseng, Y.-T. Yang, M.-S. Lai, and L.-D. Chou, “SVM-based classification mechanism and its application in SDN networks,” in Proc. 10th Int. Conf. Commun. Softw. Netw. (ICCSN), Jul. 2018, pp. 45–49.

[25] J. Xu, J. Wang, Q. Qi, H. Sun, and B. He, “Deep neural networks for application awareness in SDN-based network,” in Proc. IEEE 28th Int. Workshop Mach. Learn. Signal Process. (MLSP), Sep. 2018, pp. 1–6.

[26] P. Amaral, P. F. Pinto, L. Bernardo, and A. Mazandarani, “Application aware SDN architecture using semi-supervised traffic classification,” in Proc. IEEE Conf. Netw. Function Virtualization Softw. Defined Netw. (NFV-SDN), Nov. 2018, pp. 1–6.

[27] H. Z. Jahromi and D. T. Delaney, “An application awareness framework based on SDN and machine learning: Defining the roadmap and challenges,” in Proc. 10th Int. Conf. Commun. Softw. Netw. (ICCSN), Jul. 2018, pp. 411–416.

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

[29] 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. 2016, pp. 760–765.

[30] C. Yu, J. Lan, J. Xie, and Y. Hu, “QoS-aware traffic classification architecture using machine learning and deep packet inspection in SDNs,” Procedia Comput. Sci., vol. 131, pp. 1209–1216, Jan. 2018.

[31] TensorFlow. Accessed: Sep. 3, 2020. [Online]. Available: https://tensorflow.google.cn.

[32] L. van der Maaten and G. Hinton, “Visualizing data using t-SNE,” J. Mach. Learn. Res., vol. 9, pp. 2579–2605, Nov. 2008.

[33] I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, “Maxout networks,” in Proc. Int. Conf. Mach. Learn., 2013, pp. 1–9.

[34] M. Jiang, Y. Liang, X. Feng, X. Fan, Z. Pei, Y. Xue, and R. Guan, “Text classification based on deep belief network and softmax regression,” Neural Comput. Appl., vol. 29, no. 1, pp. 61–70, Jan. 2018.

[35] K. Cohen, A. Nedic, and R. Srikant, “On projected stochastic gradient descent algorithm with weighted averaging for least squares regression,” IEEE Trans. Autom. Control, vol. 62, no. 11, pp. 5974–5981, Nov. 2017.

[36] T.-T. Wong and N.-Y. Yang, “Dependency analysis of accuracy estimates in k-fold cross validation,” IEEE Trans. Knowl. Data Eng., vol. 29, no. 11, pp. 2417–2427, Nov. 2017.

[37] C. Liangjun, P. Honeine, Q. Hua, Z. Jihong, and S. Xia, “Correntropy-based robust multilayer extreme learning machines,” Pattern Recognit., vol. 84, pp. 357–370, Dec. 2018.

[38] Z. Ren and L. Yang, “Correntropy-based robust extreme learning machine for classification,” Neurocomputing, vol. 313, pp. 74–84, Nov. 2018.

[39] J. Lv, X. Wang, M. Huang, J. Shi, K. Li, and J. Li, “RISC: ICN routing mechanism incorporating SDN and community division,” Comput. Netw., vol. 123, pp. 88–103, Aug. 2017.

**NAN HU** received the B.S. degree in electric engineering and automation from Shenyang Jianzhu University, Shenyang, China, in 2010, and the M.S. degree in system engineering and the Ph.D. degree in pattern recognition and intelligent system from Northeastern University, Shenyang, in 2012 and 2016, respectively. He is currently an Assistant Professor with Shenyang Jianzhu University. His research interests include wireless sensor networks, artificial intelligence, and robot navigation.

**FANGJUN LUAN** received the B.S. degree from Shenyang Jianzhu University, Shenyang, China, and the M.S. and Ph.D. degrees from the University of Jilin, Changchun, China, in 2003 and 2007, respectively. He has been a Professor with Shenyang Jianzhu University. His current research interests include pattern recognition and intelligent building.

**XIAOXI TIAN** received the M.S. degree in control science and engineering from Shenyang Jianzhu University, Shenyang, China, in 2013. He is currently an Assistant Professor with Shenyang Jianzhu University. His research interests include distributed networks and neural networks.

**CHENGDONG WU** received the B.S. degree in electrical automation from the Shenyang Architectural and Civil Engineering Institute, Shenyang, China, in 1983, the M.S. degree in control theory and its applications from Tsinghua University, Beijing, China, in 1988, and the Ph.D. degree in industrial automation from Northeastern University, Shenyang, in 1994. He is currently a Professor with the Faculty of Robot Science and Engineering, Northeastern University. His research interests include wireless sensor networks, robot intelligent control, and artificial intelligence.

* * *