Application-Based Online Traffic Classification with Deep Learning Models on SDN Networks

Lin-Huang Chang[1,*], Tsung-Han Lee[1], Hung-Chi Chu[2], Cheng-Wei Su[1]

[1]Department of Computer Science, National Taichung University of Education, Taichung, Taiwan

[2]Department of Information and Communication Engineering, Chaoyang University of Technology, Taiwan

Received 19 December 2019; received in revised form 10 March 2020; accepted 03 July 2020

DOI: https://doi.org/10.46604/aiti.2020.4286

Abstract

The traffic classification based on the network applications is one important issue for network management. In this paper, we propose an application-based online and offline traffic classification, based on deep learning mechanisms, over software-defined network (SDN) testbed. The designed deep learning model, resigned in the SDN controller, consists of multilayer perceptron (MLP), convolutional neural network (CNN), and Stacked Auto-Encoder (SAE), in the SDN testbed. We employ an open network traffic dataset with seven most popular applications as the deep learning training and testing datasets. By using the TCPreplay tool, the dataset traffic samples are re-produced and analyzed in our SDN testbed to emulate the online traffic service. The performance analyses, in terms of accuracy, precision, recall, and F1 indicators, are conducted and compared with three deep learning models.

Keywords: software defined network, network traffic classification, deep learning, tensorflow

1. Introduction

Artificial Intelligence (AI) has been one of the hottest topics currently. Machine learning (ML), which is a subset of AI and adjusts itself in response to the data it is exposed to, will enable the machines to improve at the task with experience. Among different machine learning schemes, shallow neural networks, such as backpropagation neural network, Bayesian network, support vector machine (SVM), and C4.5 decision tree, are commonly used to build traffic classifiers for network services in many machine learning-based classification schemes. Due to the continuous expansion of network and considerable deployment of the Internet of Things (IoTs), great amount of data and information have been collected and consequently promoted the arrival of the big data era.

Furthermore, with the significant promotion in processing speed of computing hardware such as Graphics Processing Unit (GPU) and Tensor Processing Unit (TPU), the deep learning (DL) using deep neural networks, which provide higher classification accuracy than the shallow neural networks, have been the major focus on ML. The promising ML techniques, such as deep neural networks provides a good opportunity and key to the data-driven machine learning algorithms in the network field.

On the other hand, more and more application services have emerged over the Internet. The network management issues, including quality of service (QoS) setting, network policy, network security as well as intrusion detection, majorly depend on the accuracy of network traffic classification for applications. The emerged applications or services may not be identified by

* Corresponding author. E-mail address: lchang@mail.ntcu.edu.tw

Tel.: +886-4-22183812; Fax: +886-4-22183580

the traditional schemes, such as simply checking source/destination IP addresses in the network layer or source/destination port numbers as well as protocols in the transport layer, because they might pass through different Network Address Translation (NAT) or Virtual Private Network (VPN), or they may not utilize the standard port numbers or fields. Therefore, to provide proper QoS for a specific application, it might need to manually verify or identify the network application packets. These processes will significantly reduce the efficiency of network traffic processing and consequently increase the overall delay and processing loading.

Furthermore, the development of software defined network (SDN) has made network management more effective and efficient [1-2]. Unlike traditional networks, SDN separates the network layer into a data plane and a control plane which could be a logical control unit and programmable. The flexibility and programmability of the SDN architecture design make the network routing and management easier and more efficient, especially for the design and handling of network classification which enables the optimal network solutions, such as configuration and resource allocation. The centralized SDN controller is able to collect various real-time network data due to its global network view. The AI techniques can then be applied to the SDN networks by employing network optimization and data analysis based on the real-time and historical SDN network data.

Therefore in this paper, we propose an application-based online and offline traffic classification with deep learning models on SDN networks. We design the deep learning models resigned in the SDN controller. The SDN controller establishes the match fields of the flow entry and sends them to the open virtual switch (OVS). It also extracts traffic statistics data from the OVS switch. The server IP addresses and transport port numbers of each flow plus the statistics data are designed to be the input features for the deep learning models.

The remaining sections of this paper are organized as follows. We describe the protocols, standards, tools, and related research reviews in Section 2. Section 3 addresses the experimental settings and model design. In Section 4, we present the experimental result and performance analysis in this paper followed by the conclusion and future works in Section 5.

2. Related Work

We first review some protocols, standards as well as related tools used in this paper. We also survey some literature related to this research. The following subsections will address the brief discussion of each subject.

2.1. Deep learning review

DL [3], rather than a task-specific algorithm in the ML series, is a broader representation of learning data. The DL aspect includes supervised, semi-supervised, or unsupervised learning. DL can be applied for classification, clustering, and regression. With classification, DL is able to learn the correlation between data and labels, which is known as supervised learning.

For some applications, it is not required or unable to know the labels of the data in advance. Learning without labels is called unsupervised learning. With clustering, DL does not require labels to detect similarities among groups of data. With regression, the DL is exposed to enough of the right data. It is able to establish correlations between present events and future events. This predictive analytics is different from the classification which might be called a static prediction. Given a time series, DL can run regression by reading lots of data from the past and predict some data likely to occur in the future.

DL architectures, including convolutional neural network (CNN), deep neural networks (DNNs) and recurrent neural network (RNN), multilayer perceptron (MLP), long short-term memory (LSTM), and stacked auto-encoder (SAE), have been applied to different fields such as computer vision, audio recognition and natural language processing, etc. Some of DL models or implementations have achieved comparable or even superior results as compared to human experts.

In this paper, we apply the TensorFlow [4] neural network modules as the DL platform for our application-based online traffic classification. The major programming language and development of TensorFlow is Python. TensorFlow supports application programming interfaces (APIs) such as Keras [5], with the advantage of easy operation, modular design, and flexible scalability, as the high-level development interface. Keras, working in conjunction with the back-end TensorFlow engine, provides functional APIs for building models, training models, evaluating models, as well as predicting results.

2.2. Openflow protocol

OpenFlow [6] is one of the most dominated SDN communication protocols. By using the software programming, OpenFlow supports the control and communication of the rules signaling and flows forwarding from the centralized controller southbound to the SDN switch. The northbound APIs from the control plan to the management plan, on the other hand, is defined to be used for application service and operation management.

Several consoles, such as Ryu [7], Floodlight, and OpenDayLight, have been developed to support OpenFlow. We apply the Ryu controller which supports OpenFlow up to versions 1.5 to define the framework of the SDN networks in this research. Ryu, being a component-based SDN framework, provides software components with well-defined API, which is easy for developers to create new control applications and network management. It also supports various communication protocols for network devices management, such as OpenFlow and OF-config.

2.3. Tcpreplay tools

Tcpreplay [8], an open-source utility, is used to edit and replay captured network packets. It has been used primarily to replay malicious traffic patterns for network intrusion detection. We apply Tcpreplay to replay the network traffics of the real-world dataset and inject them into the SDN network environment in this research.

2.4. ISCX dataset

The ISCX [9] dataset, collected by the Canadian Cyber Security Institute, is a network traffic dataset used by many universities, companies, and independent researchers around the world. In this paper, we apply the ISCX dataset as our training dataset and offline testing dataset.

2.5. Survey on AI applied to SDN and related review

The flexibility and programmability of the SDN network make the traffic classification, routing optimization, QoS/QoE prediction, resource management, and security easier and more efficient. With the advantage of the SDN global network view, the centralized SDN controller is able to collect various real-time and historical network data from the SDN switches at per port and per-flow granularity levels and consequently, we can apply the AI techniques on the SDN networks by employing network optimization and data analysis. We will provide a brief survey of AI or ML techniques applied to the SDN network, including traffic classification, routing optimization, QoS/QoE prediction, and security [10].

First, traffic classification techniques in SDN networks include a port-based approach, Deep Packet Inspection (DPI), and AI or ML [11-12]. The port-based approach will not be effective when most applications recently utilize dynamic ports as TCP or UDP port numbers. The DPI approach results in high computational cost or difficult pattern update because all traffic flows need to be checked or updated for the exponential growth of applications. Currently, more and more encrypted packets of various applications make the DPI approach even impractical. AI or ML techniques are applied to extract knowledge from the traffic flows in SDN, thus AI or ML-based approaches are able to classify encrypted packets correctly with the lower computational cost.

Deep learning models have been applied to network traffic classification currently. The research in [13] introduced a SAE-based scheme to classify unencrypted data flows. However, their scheme was only applied to unencrypted traffic and could not be deployed to the encrypted data. Besides, the dataset used in their research was not open to the public. The research in [14] proposed a scheme to identify encrypted traffic based on SAE and CNN models. The research in [15] proposed three deep learning based models, including MLP, SAE, and CNN for traffic classification. They developed their models based on all encrypted streaming packets from the open source dataset. However, their research, including training and prediction models, could not be applied to the real network traffic or emulated online flows because they only conducted the offline dataset analysis.

Secondly, the routing optimization issues in SDN networks typically employ the Shortest Path First (SPF) algorithm and heuristic algorithms [16]. The SPF algorithm is a best-effort routing scheme. It is unable to utilize the best network resources due to the simplicity of the algorithm. The high computational cost however would be the shortcoming of heuristic algorithms [17], such as ant colony optimization algorithm. The introduction of AI or ML to the route optimization in SDN can be considered as a decision-making policy which does not need a complex mathematical model once the model has been trained. The reinforcement learning could be an effective mechanism which may provide a near-optimal routing decision quickly.

Thirdly, based on QoS/QoE prediction, the SDN network operators or service providers are able to provide suitable services to the customers according to their expectations which consequently increase customer satisfaction and service. The authors in [18] have proposed a traditional M/M/1 network model and neural network model to train the model with traffic load and overlay routing policy and then to estimate the network delay. The authors in [19] have focused on the QoE prediction for video streaming service in the SDN network by correlating the QoS parameters with the QoE values. They applied a supervised learning model to estimate the mean opinion score (MOS) value according to the SDN network parameters, such as delay, jitter, and bandwidth, to adjust video parameters, such as bitrate and resolution, in the SDN controller and consequently improve the QoE of users' experience.

Lastly, security is always an important issue in SDN network research and operation. Intrusion Detection System (IDS) is an important mechanism and system for network security which in general consists of signature-based IDS and anomaly-based IDS according to how they identify network attacks [20]. Because the signature-based IDS has some shortcomings, such as signature update difficulty and high time consumption for all signatures comparison, most researches focus on the anomaly-based IDS.

The anomaly-based IDS is a flow-based traffic identification which in general inspects the packet header information based on flow-granularity information, while the signature-based IDS being a payload-based identification which needs to inspect the whole payload of packets. The supervised learning algorithms of AI or ML techniques are often applied in anomaly-based IDS by training a predefined model to identify intrusions and normal flows. Again, the global view and programmability of SDN networks will facilitate the AI or ML-based IDS because of its simplicity and flexibility of data collection and attack reaction, respectively.

With proper feature selection and feature extraction, many studies have been conducted for AI or ML-based IDS in SDN networks. The authors in [21] employed the data preprocessing, decision-making subsystem, and response subsystem for their predictive data model as a threat-aware IDS in SDN. They used a forward feature selection strategy to select appropriate feature sets in the data preprocessing subsystem, applied the decision tree and random forest algorithms to detect malicious flows in the predictive data modeling subsystem, and then employed the reactive routing to install different flow rules and types based on the detection results in the decision making and response subsystems. The authors in [22] used five flow features, including source/destination IP, source/destination port, and packet length, to predict malicious flows for their proposed HMM-based network IDS.

In this paper, we will mainly focus on the deep learning models applied to the traffic classification, especially application-based traffics, in the realm of SDN. The SDN switches send the traffic statistics and Packet_In messages to the SDN controller for features extraction, which include server IP and port of each flow plus the statistics data. The SDN controller with deep learning models, including MLP, CNN, and SAE models, will conduct the application-based traffic classification for each flow by establishing the match fields of the flow.

3. Experimental Settings and Model Design

3.1. Experimental setting

Fig. 1 illustrates the network topology for our proposed SDN network testbed. As shown in Fig. 1, we first apply the ISCX dataset [9] to emulate the offline and online network traffic by injecting them into Tcpreplay to reproduce the client/server communications in the SDN testbed. As mentioned earlier, the ISCX dataset, including VPN and non-VPN datasets, was collected by the Canadian Institute for Cybersecurity at the University of New Brunswick. Both VPN and non-VPN datasets consist of six application services, including Skype, Facebook, Hangouts, Youtube, and etc.

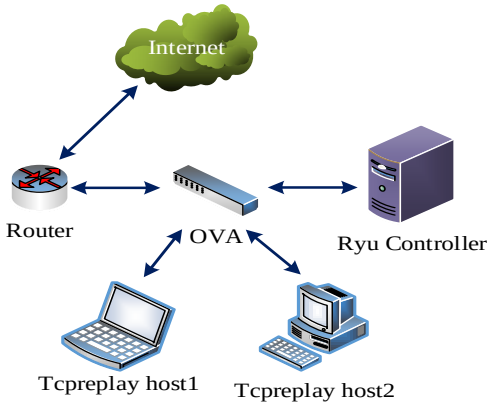

Fig. 1 Network topology of the SDN testbed

In this research, we deploy TP-Link TL-WR1043ND as the open virtual switch (OVS) based on the OpenFlow protocol for the SDN data plan. The OVS is implemented and installed using OpenWRT [23]. The Ryu controller [7] is used in this paper to define the framework of the SDN networks. As defined by OpenFlow standard, whenever a new coming packet does not match any flow entry in the flow table of OVS, the OVS switch will trigger the Packet_in mechanism and send it to the Ryu controller as routing setting request.

For all packets in the same flow, only one Packet_in message is sent to the controller for the same flow. In this research, besides the Packet_in message for each flow, we also collect the flow statistics data. For the OpenFlow statistics mechanism, the OVS sends the periodical statistics reports to the controller in response to the controller's statistics request. The OVS statistics reported in 1sec interval, including flow statistics, table statistics, port statistics, queue statistics, group statistics, and meter statistics, could be used as the deep learning features for traffic flow behavior learning.

The flow statistics data combined with the Packet_in message will be the learning features of the deep learning models designed in the Ryu controller for a specific flow. For each SDN flow, keeping the session connected, the OVS statistics report will be sent to the Ryu controller continuously in 1sec interval. This will avoid the stochastic learning results due to inconstant or random statistics reports.

Because of the traffic flow characteristics of SDN networks, the processing data including Packet_in message and statistics data in 1sec interval for deep learning in this research is less than the traditional traffic classification which processes data from all packets.

3.2. Data pre-processing

In this paper, the dataset was collected in a precise and pre-defined amount and the experiments were conducted for three different deep learning models ten times. Most input data are the features with a one-hot input scheme except two features, i.e. byte counts and packet counts, which will be discussed in more detail shortly. Since deep learning is a mathematical operation of non-linear regression, it is necessary to limit the eigenvalues input to the classification model to a fixed range.

Therefore, the keys and values are stored in mapping mode using Hash handles, and MinMaxScaler in scikit-learn [24] to normalize the values and scale them to the range of 0 to 1. The MinMaxScaler is shown in Eq. (1), where X_i represents the selected feature value, X_{nom} represents the normalized value of one specific feature, and $Min_{(x)}$ and $Max_{(x)}$ are the minimum and maximum values of the feature, respectively.

$$X_{nom} = \frac{X_i - Min_{(x)}}{Max_{(x)} - Min_{(x)}} \in [0, 1] \quad (1)$$

Table 1 shows the ISCX dataset used in this experiment, which is injected into Tcpreplay to emulate the offline and online network for the client/server communications in the SDN testbed. There are six applications, including Facebook, Gmail, Hangouts, Netflix, Skype, and YouTube, for each VPN and non-VPN dataset in the ISCX dataset. For the Skype application, the dataset was further divided into two classes, SkypeAudio and SkypeVideo.

Table 1 Dataset used in deep learning model

Labels	Applications	Training data amount	Validation data amount	Testing data amount
facebookChat	facebook	9981	1109	11091
gmailChat	gmail			
hangoutChat	hangout			
netflix	netflix			
skypeAudio	skype			
skypeVideo				
youtube	youtube			

Table 2 Examples of selected features and labels

Data #	Top1	Top2	Top3	Top4	Top5	Top6	…	Top305	Other	Byte Count	Packet Count	labels
1	0	0	0	0	0	0	…	0	0	0.08	0.08	skypeAudio
2	1	0	0	0	0	0	…	0	0	5.87	0.16	skypeAudio
3	0	0	0	0	0	0	…	0	0	0.12	0.12	skypeAudio
4	1	0	0	0	0	0	…	0	0	7.57	0.14	skypeAudio
5	1	0	0	0	0	0	…	0	0	8.12	0.15	skypeAudio
6	0	0	0	0	0	0	…	0	0	0.06	0.06	netflix
7	1	0	0	0	0	0	…	0	0	152.17	0.9	skypeVideo
…	…	…	…	…	…	…	…	…	…	…	…	…
8	0	0	0	0	0	0	…	0	0	0.09	0.09	hangoutChat
9	0	0	0	0	0	0	…	0	0	0.21	0.1	hangoutChat
10	0	0	0	0	0	0	…	0	0	10.89	0.06	netflix
11	0	0	0	0	0	0	…	0	0	0.06	0.06	youtube
12	0	0	0	0	0	0	…	0	0	0.08	0.08	skypeAudio
13	0	0	0	0	0	0	…	0	0	1.53	6	gmailChat
14	0	0	0	1	0	0	…	0	0	1.36	9	gmailChat
15	0	0	0	0	0	0	…	0	0	0.06	1	netflix
16	0	0	0	0	0	0	…	0	0	2.54	7	facebookChat
17	0	1	0	0	0	0	…	0	0	0.5	1	gmailChat
…	…	…	…	…	…	…	…	…	…	…	…	…

So, we will have seven Labels for all deep learning models. The total amount of training dataset is 11090, with which 90% of them is used for the deep learning training process and 10% of them is used for the deep learning validation process. The total amount of testing dataset is 11091, which is also injected into Tcpreplay and then replayed one by one to emulate the online traffic in the SDN network.

Because we emulate the real traffic by using the ISCX open-source dataset, the dataset packets are handled as Tcpreplay pcap packets. The Packet_in messages due to the flow mismatch in OVS switch and the periodical statistics data responded from OVS switch to the controller are collected on the Ryu controller.

The server IP addresses and transport port numbers in each flow and the byte counts and packet counts collected from statistics data are selected as the deep learning features in our designed models. The server IP addresses and transport port numbers are handled using the one-hot input scheme. Table 2 shows the examples of the selected features and labels in our designed application-based deep learning models. The top 305 most frequent server IP addresses and transport port numbers are selected as the one-hot input for deep learning models. The Other column in one-hot input is used for any other IP or port number, not in the top 305 most frequent lists.

3.3. Deep learning model design

In this paper, we apply three deep learning models, including CNN, MLP, and SAE models, for our application-based offline and online traffic classification.

3.3.1. MLP model

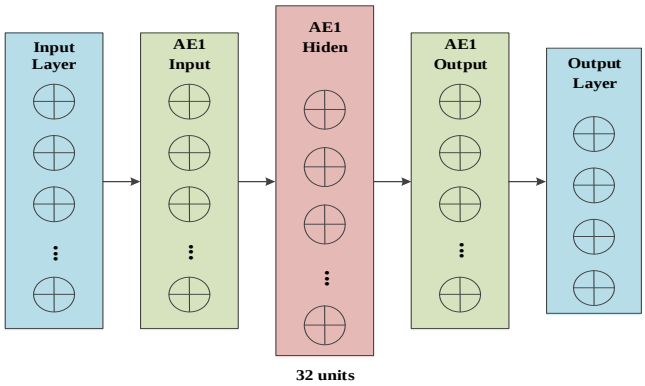

Fig. 2 MLP deep learning in our model

The first deep learning model we apply in this research is the multilayer perceptron (MLP) model. The designed MLP deep learning model, shown in Fig. 2, employs back-propagating supervised learning techniques in artificial neural networks. It consists of an input layer, three hidden layers followed by a dropout layer, and an output layer. The input layer consists of 305 one-hot inputs representing the most frequent server IP addresses and transport port numbers plus one Other column one-hot input representing the less frequent server IP addresses and port numbers from the Packet_in headers. Besides, two inputs are representing the byte counts and packet counts in every one-second periodical statistics data. Each of the three hidden layers is composed of 256 neurons. The dropout layer is used to prevent the overfitting issue. The output layer consists of seven neurons, and the Softmax function is applied as a classifier for the MLP deep learning model.

3.3.2. SAE model

In this research, we also apply the stacked autoencoder (SAE) as one of the deep learning training models. The proposed SAE architecture, containing one single encoding and decoding, is illustrated in Fig. 3. The encoders of SAE deep learning model, used for dimension reduction or feature extraction, contains 32 neurons in this paper.

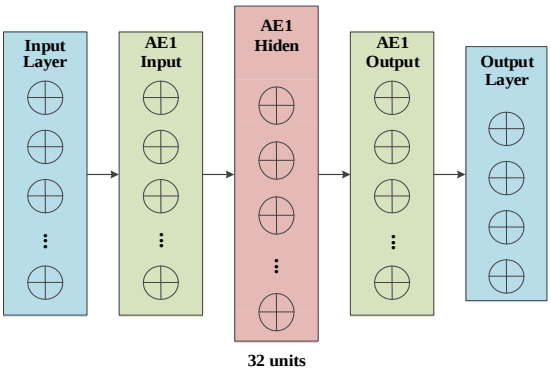

Fig. 3 SAE deep learning in our model

3.3.3. CNN model

The third deep learning model we apply in our deep learning training is the CNN model which is a typical deep learning model used for classification. The designed CNN model is specified in Fig. 4 which consists of three convolution layers, three max-pooling layers as well as one fully connected layer with Relu function acting as an activation function, followed by a dropout layer and an output layer. The convolution layer uses 64 filters to process input data. Again, the dropout layer is used to handle the overfitting issues.

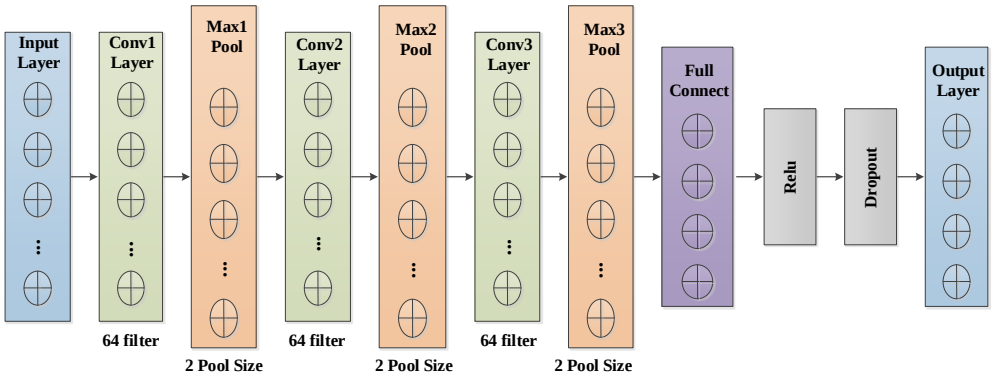

Fig. 4 CNN deep learning in our model

3.3.4. Deep learning model training parameters

The training dataset is divided in 9:1 ratio into training and verification processes for all three MLP, ASE and CNN models in our deep learning training processes. The training parameters, such as epochs, batch size, learning rate, as well as optimizers, of our deep learning models are tabulated in Table 3.

Table 3 Deep learning model training parameters

Models	Epochs	Batch	Rate	Optimizers
MLP	100	512	0.01	Adam
SAE				
CNN				

4. Experimental Result and Analysis

4.1. Offline training result

In this paper, we first apply three deep learning models, including MLP, SAE, and CNN models, for our application-based offline traffic classification. For the model training of offline deep learning processes, the training history for three different deep learning models is presented in Fig. 5. The experimental results of the training history include the accuracy and loss

function evolution for both training and validation processes. The accuracy and loss rate within ten epochs of validation process for CNN deep learning model is about 94.95% and 20.26%, respectively. The CNN deep learning model shows fairly steady convergence within ten epochs. However, the loss function of validation process keeps almost unchanged after ten epochs and even goes little bit higher than that of training process. This could be due to the overfitting issue. It might need further optimization of the CNN parameters to reduce the overfitting issue.

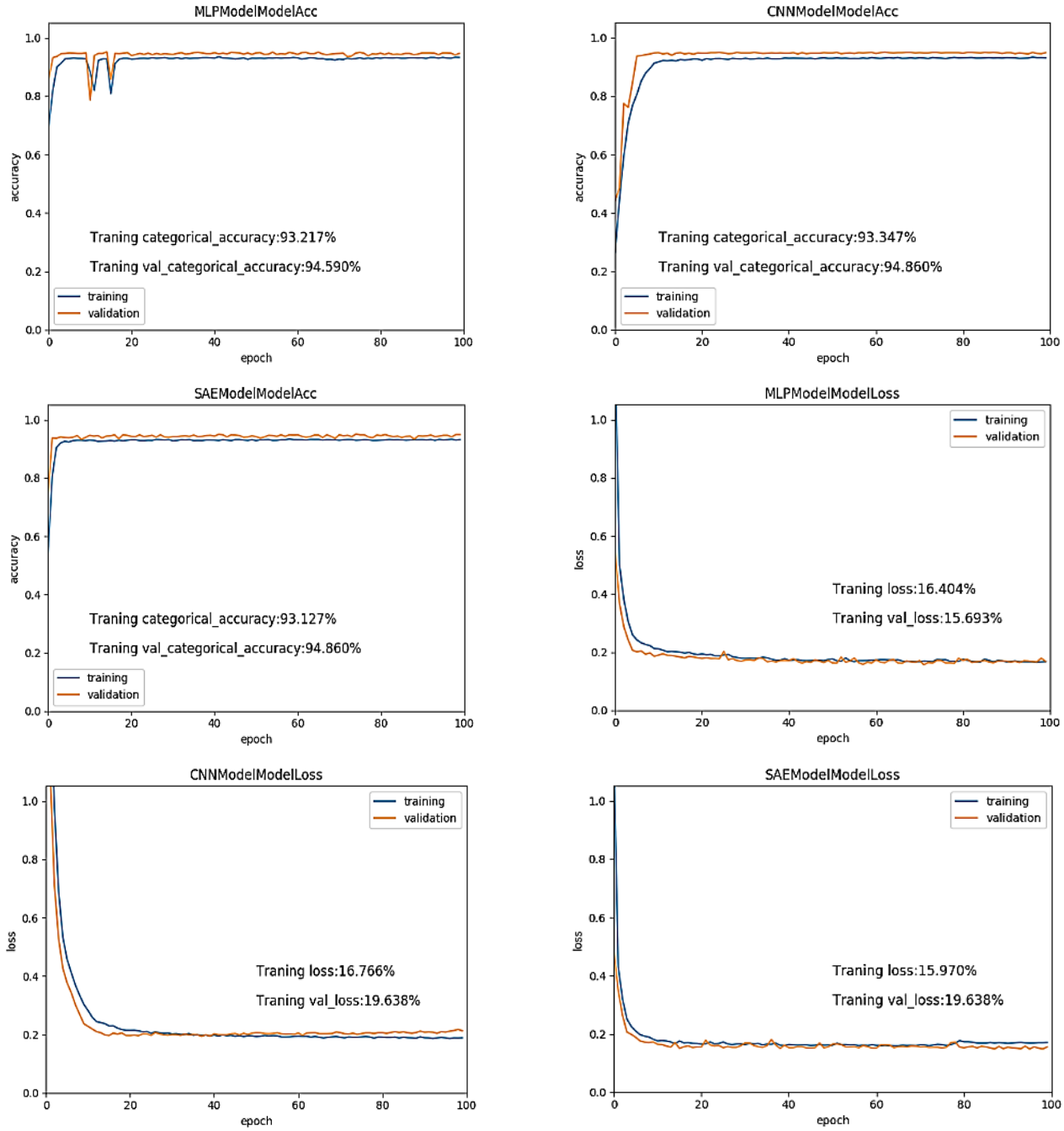

Fig. 5 Offline training results for three different deep learning models

The accuracy and loss rate within ten epochs of validation process for MLP deep learning model is about 94.95% and 17.63%, respectively. There are some turbulences in training and validation processes after ten epochs. However, it achieves steady convergence after about 30 epochs. The accuracy and loss rate within ten epochs of the validation process for SAE deep learning model is about 94.86% and 19.638%, respectively. Unlike MLP model, the SAE deep learning model shows fairly good convergence within five to ten epochs. This could be due to the fine-tuning training mechanism of the SAE pre-training. However, more training epochs, less than 20 epochs, are enough for them to obtain similar error rates.

Table 4 Offline training result for different models

	CNN-1D	MLP	SAE
Training process accuracy	93.35%	93.21%	93.13%
Training loss rate	16.77%	16.40%	15.97%
Validation process accuracy	94.86%	94.59%	94.86%
Validation loss rate	19.64%	15.69%	19.64%

The offline training results for three different deep learning models, MLP, SAE, and CNN, are summarized in Table 4. More than 93% accuracy is achieved for all three different models.

The offline training time, directly measured from the system, for CNN, MLP and SAE models are 1,518, 111, 25 sec, respectively. The complexity of ML models, especially for the deep learning models, has been significantly increasing over recent years. This is mainly due to the increasing number of deep learning network layers and the volume of dataset which consequently increases the computational cost corresponding to the execution time required training a deep learning network. Fortunately, this computational cost is only effective during offline training phase. In this research with online traffic classification in SDN during testing phase, the execution time to identify an application for each flow is within few seconds which is relatively small. The trained models for online testing can reach a very high confidence on classification within several statistics data which was collected in every second. The detailed online testing results will be discussed in the next sub-section.

4.2. Online testing result

As mentioned earlier, we have injected the ISCX dataset into Tcpreplay host and then replayed them one by one according to the same processing time and sequence to emulate the online traffic for the client/server communications in the SDN testbed. The server IP addresses and transport port numbers in each flow and the byte counts and packet counts collected from statistics data are stored and selected in Ryu controller for further data pre-processing so that the server IP addresses and transport port numbers can be handled using one-hot input scheme.

For online testing, we use other dataset files with the same applications and pre-processing as the training model to match the input size of the training model. In order to response to the unknown applications, we add a server to mirror packet in message in controller. The mirror server will label the online applications or services for predicted result comparison. Furthermore, in the case of vast amount of packets or flows injected into the OVS switch, such as abnormal intrusion, the mirror server can play a role of OVS buffering through the dropping of the vast injected packets on OVS switch to avoid the bandwidth exhaustion.

Because of the limitation of the processing speed of the Ryu controller and OVS switch hardware, we might encounter statistics packets drop or Tcpreplay packets drop when the testing dataset injected into the OVS switch according to the same processing time and sequence. Therefore, the incomplete dataset due to the packets drop might cause the accuracy decline of the online prediction for different deep learning models as compared with the offline cases.

In Fig. 6, we illustrate the predicted result, in terms of confusion matrix, of application-based online traffic classification with MLP deep learning model. The diagonal values in the matrix correspond to the true positives of the predicted result of FacebookChat, gmailChat, HandoutsChat, Netflix, SkypeAudio, SkypeVideo, and Youtube categories, respectively. From the illustration shown in Fig. 6, the number of false positives between SkypeAudio and SkypeVideo categories is relatively high. This is understandable because the flows from these two categories own the same server IP and transport port features. The only possible features which we can identify could be byte counts. However, the dynamic encoding schemes of the audio and video could dim the byte count feature.

The numbers of false positives between Netflix and SkypeAudio or between YouTube and SkypeAudio also show relatively higher ratio. Although the server IP addresses are different for these applications, the similar multimedia behavior in terms of byte counts and packet counts features for deep learning will lead the miss-identification of such streaming category to the SkypeAudio category. As a simple deep back-propagating learning technique, the MLP deep learning model may suffer from such miss-identification easily. Therefore, a relatively higher error rates were observed between SkypeAudio and Skypevideo or between Netflix and SkypeAudio applications. However, we still achieve about 87% accuracy on average for application-based online traffic classification with MLP deep learning model. Especially, we conduct the application services with very similar characteristics, such as multimedia, audio and video streaming, which could be very difficult for network traffic classification.

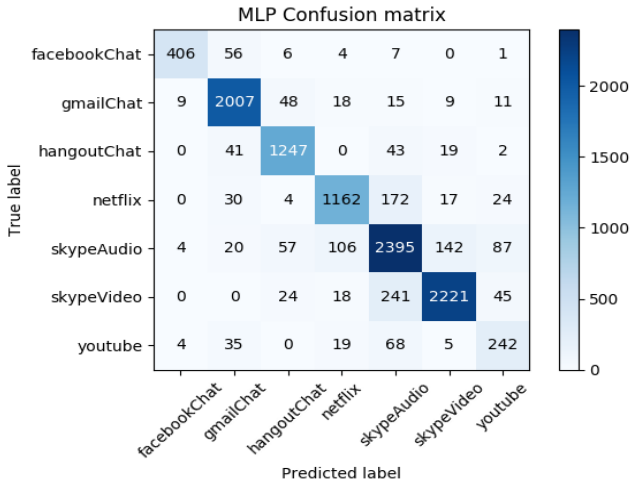

Fig. 6 Confusion matrix of MLP online test result

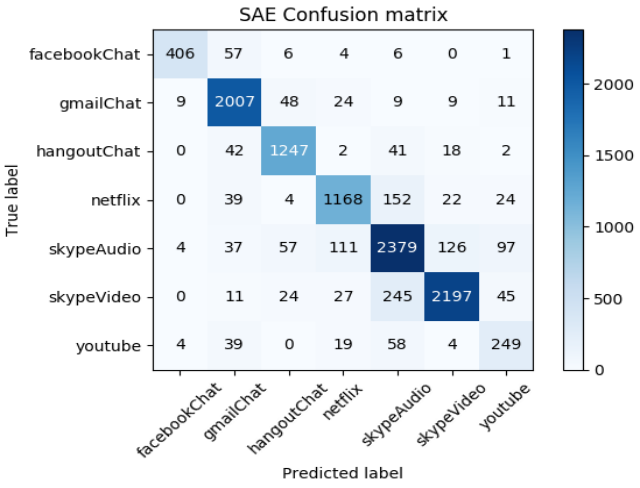

Fig. 7 Confusion matrix of SAE online test result

The predicted result, in terms of confusion matrix, of application-based online traffic classification with SAE deep learning model is illustrated in Fig. 7. Similarly, the number of false positives between SkypeAudio and Skypevideo categories is relatively high again for the SAE deep learning model. The numbers of false positives between Netflix and SkypeAudio or between YouTube and SkypeAudio also shows relatively higher ratio. Because the SAE deep learning model uses encoder and decoder for learning, the selected features need to be further adjusted to distinguish the similar streaming behavior between SkypeAudio and SkypeVideo categories or among different multimedia streaming properties.

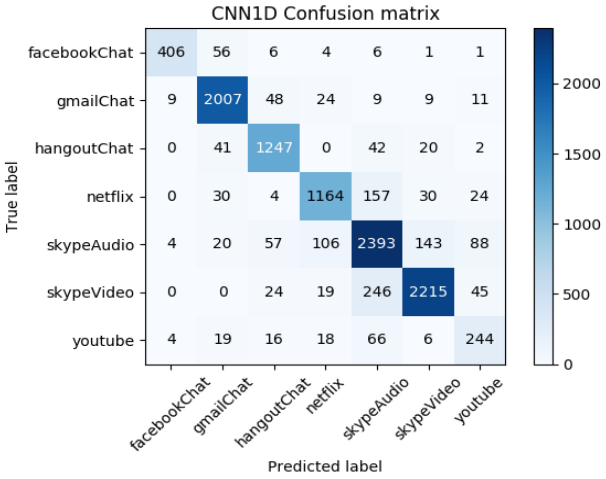

Fig. 8 Confusion matrix of CNN online test result

Besides, the predicted result, in terms of confusion matrix, of application-based online traffic classification with CNN deep learning model is illustrated in Fig. 8. Similarly, the numbers of false positives between SkypeAudio and Skypevideo categories are relatively high for the CNN deep learning model. The numbers of false positives between Netflix and

SkypeAudio or between YouTube and SkypeAudio also show relatively higher ratio which is believed due to the similar multimedia behavior in terms of byte counts and packet counts features for deep learning. The average accuracy for application-based online traffic classification with CNN deep learning model is about 87.3%.

In this paper, we reduce the problem of multiclass classification to multiple binary classification problems by using one vs rest transformation technique [25]. The one vs rest transformation technique will treat a single classifier per class with the predicted labels of that class as positive samples and all other predicted labels as negatives. Thus, four indicators, Accuracy, Precision, Recall, and F1, were calculated using the confusion matrix to evaluate the performance of application-based online traffic classification with different deep learning models. The online testing result with four indicators in our SDN testbed is shown in Table 5.

Table 5 Online testing result for different models

	Accuracy	Precision	Recall	F1-score
CNN	87.208%	85.142%	84.428%	84.857%
MLP	87.167%	84.428%	87.714%	84.714%
SAE	87.079%	85.142%	84.285%	84.714%

Based on the experimental results, the average accuracies for different deep learning models are about the same, 87%, which is smaller than our offline training result, above 93%. Due to the hardware limitation, we might encounter possible statistics packets drop or Tcpreplay packets drop when the testing dataset injected into the OVS switch according to the same processing time and sequence. Consequently, the incomplete dataset due to the packets drop might cause the accuracy decline of the online prediction for different deep learning models as compared with the offline cases. On the other hand, the increasing number and variation of application services might increase the difficulty on application-based online traffic classification on SDN Networks with Deep Learning Models.

Although the numbers of false positives between SkypeAudio and Skypevideo categories are relatively high and the numbers of false positives between Netflix and SkypeAudio or between YouTube and SkypeAudio also show relatively higher ratio for all deep learning models, the values of precision, recall and F1-score are all above 84% and quite close to the accuracy value. As we commented, the application services conducted in this paper with very similar characteristics, such as multimedia, audio and video streaming, could be very difficult for network traffic classification. However, with the help of deep learning models, including CNN, MLP and SAE, the average online testing accuracies for all models are all above 87% with quite close precision, recall and F1-score values.

5. Conclusions

Deep learning techniques have become one of the most interesting and practical topics being applied on all kinds of fields, such as computer vision, audio recognition and natural language processing. In this paper, we have proposed an application-based online and offline traffic classification, based on deep learning mechanisms, over SDN testbed. The designed deep learning architecture and scheme consists of three deep learning models, MLP, SAE, and CNN, in the SDN testbed. We have applied an open network traffic dataset with seven most popular applications as the deep learning training and testing datasets. By using the Tcpreplay tool, the dataset traffic samples are re-produced and analyzed in our SDN testbed to emulate the online traffic service.

We have conducted performance analysis, in terms of accuracy, precision, recall, and F1 indicators, and compared the results with three deep learning models. The offline training results have achieved more than 93% accuracy on identifying seven popular applications for all three different models. Furthermore, we have achieved 87% accuracy for the application-based online testing prediction.

In the future, we will correlate the deep learning parameters, models and accuracy to improve the overall learning performance. The real online packets measurement and analysis in SDN network will be conducted in the next step. The network slicing in SDN network with QoS mechanism followed by successful traffic classification could be the further research.

Acknowledgement

This research was supported by research grants from Ministry of Science and Technology, Taiwan (MOST 107-2221-E-142 -002 -MY2) as well as Mobile Broadband Promotion Project, Ministry of Education, Taiwan (No. 1070208431G).

Conflicts of Interest

The authors declare no conflict of interest.

References

[1] M. N. A. Sheikh, "SDN-Based approach to evaluate the best controller: internal controller NOX and external controllers POX, ONOS, RYU," Global Journal of Computer Science and Technology, vol. 19, no. 1-E, February 2019.

[2] H. C. Chu, Y. X. Liao, L. H. Chang, and Y. H. Lee, "Traffic light cycle configuration of single intersection based on modified Q-Learning", Applied Sciences, vol. 9, no. 21, article 4558, October 2019.

[3] S. Rezaei and X. Liu, "Deep learning for encrypted traffic classification: an overview," IEEE Communications Magazine, vol. 57, no. 5, pp. 76-81, May 2019.

[4] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, et al, "Tensorflow: A system for large-scale machine learning," 12th USENIX Conference on Operating Systems Design and Implementation, August. 2016, pp. 265-283.

[5] "Keras," https://keras.io/, June 2019.

[6] "Open Networking Foundation," https://www.opennetworking.org/, June 2019.

[7] "Ryu SDN Framework Community," https://ryu-sdn.org/, June 2019.

[8] "Tcpreplay," https://tcpreplay.appneta.com/.

[9] G. Draper-Gil, A. H. Lashkari, M. Mamun, and A. A. Ghorbani, "Characterization of encrypted and VPN traffic using time-related features," Proc. 2nd International Conference on Information Systems Security and Privacy, February 2016, pp. 407-414.

[10] J. Xie, F. R. Yu, T. Huang, R. Xie, J. Liu, C. Wang, and Y. Liu, "A survey of machine learning techniques applied to software defined networking (SDN): Research issues and challenges," IEEE Communications Surveys & Tutorials, vol. 21, no. 1, pp. 393-430, 2019.

[11] 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," IEEE 24th International Conference on Network Protocols, IEEE Press, November 2016, pp. 1-5.

[12] P. Wang, S. C. Lin, and M. Luo, "A framework for QoS-aware traffic classification using semi-supervised machine learning in SDNs," Proc. IEEE International Conference on Services Computing, IEEE Press, June 2016, pp. 760-765.

[13] R. Thupae, B. Isong, N. Gasela, and A. M. Abu-Mahfouz, "Machine learning techniques for traffic identification and classification in SDWSN: a survey," 44th Annual Conference of the IEEE Industrial Electronics Society in Iecon, IEEE Press, October 2018, pp. 4645-4650.

[14] H. K. Lim, J. B. Kim, J. S. Heo, K. Kim, Y. G. Hong, and Y. H. Han, "Packet-based network traffic classification using deep learning," International Conference on Artificial Intelligence in Information and Communication, IEEE Press, February 2019, pp. 046-051.

[15] P. Wang, F. Ye, X. Chen, and Y. Qian, "Datanet: deep learning based encrypted network traffic classification in SDN home gateway," vol. 6, article 18174852, pp. 55380-55391, September 2018.

[16] R. Hajlaoui, H. Guyennet, and T. Moulahi, "A survey on heuristic-based routing methods in vehicular ad-hoc network: Technical challenges and future trends," IEEE Sensors Journal, vol. 16, no. 17, pp. 6782-6792, September 2016.

[17] A. Azzouni, R. Boutaba, and G. Pujolle, "NeuRoute: predictive dynamic routing for software-defined networks," 13th International Conference on Network and Service Management, IEEE Press, January 2017, pp. 1-6.

[18] J. Carner, A. Mestres, E. Alarcn, and A. Cabellos, "Machine learning based network modeling: an artificial neural network model vs a theoretical inspired model," Ninth International Conference on Ubiquitous and Future Networks, IEEE Press, July 2017, pp. 522-524.

[19] T. H. Lee, L. H. Chang, and W. C. Cheng, "Design and implementation of SDN-based 6LBR with QoS mechanism over heterogeneous WSN and internet," KSII Transactions on Internet & Information Systems, vol. 11, No. 2, pp. 1070-1088, February 2017.

[20] N. Sultana, N. Chilamkurti, W. Peng, and R. Alhadad, "Survey on SDN based network intrusion detection system using machine learning approaches," Peer-to-Peer Networking and Applications, vol. 12, no. 2, pp. 493-501, January 2018.

[21] C. Song, Y. Park, K. Golani, Y. Kim, K. Bhatt, and K. Goswami, "Machine-learning based threat-aware system in software defined networks," 26th International Conference on Computer Communication and Networks, IEEE Press, July 2017, pp. 1-9.

[22] T. Hurley, J. E. Perdomo, and A. Perez-Pons, "HMM-based intrusion detection system for software defined networking," 15th IEEE International Conference on Machine Learning and Applications, IEEE Press, December 2016, pp. 617-621.

[23] "OpenWrt," https://openwrt.org/, June 2019.

[24] "scikit-learn," https://scikit-learn.org/stable/index.html, June 2019.

[25] M. L. Zhang and Z. H. Zhou, "A review on multi-label learning algorithms," IEEE Transactions on Knowledge and Data Engineering, vol. 26, no. 8, pp. 1819-1837, August 2014.