Deep RNN-based Traffic Analyzation Scheme for Detecting Target Applications

Author 1, Author 2, Author 3, Author 4 and Author 5

***Abstract*—This letter proposes a deep RNN-based traffic alayzation scheme for detecting target applications. The proposed scheme improves the classification performance in telecommunication networks with heavy traffic. In this work, we design a novel classification learning method where input features and output labels are two-dimensional image with traffic packet and target applications, respectively. Specifically, traffic packets are provided through universitat politecnica de catalunya barcelonatech as the training data[1]. Then, the proposed scheme provides a fast and exact traffic alayzation from the produced inferred function by analyzing the training data. We implement the proposed scheme using a commercial deep long short-term memory system. Simulation-based experiments show that the proposed scheme achieves almost xx% accuracy performance with low complexity, requiring only xx ms elapsted time.**

***Index Terms*—LSTM, Deep Learning, Network Traffic, Traffic Analyzation.**

I. Introduction

VARIOUS types of applications and services are operatting in recent networks. It also provides easy access to the network from anywhere, including the smartphone, laptop, and desktop users. As the quality of demand increases, larger and more diverse traffic data is occurring. Therefore, the scope of human data analysis and management is greatly expanding[2]. Therefore, it is necessary to establish a lightweight and automated network[3]. In addition, the application of Deep Learning, which is gaining attention recently, is increasing. Some of deep learning technologies include Multilayer Perceptron (MLP), CNN, and Recurrent Neural Network (RNN). The MLP is a basic deep learning model consisting of approximately one or more hidden layers. CNN, which is a neural network that is comprised of three levels of local reception area, convolutional layer, and pooling layer, is a deep learning technique that is typically used in image analysis. RNN is a circular neural network suitable for processing random sequential data, and its characteristics are influenced by previous computational results. In this paper, RNN is used among deep learning techniques to classify network traffic based on flow. The flow is a collection of the same traffic by 5-tuple, which in turn contains network packets. Therefore, RNN, which is suitable for training sequential data, is used for flow-based network traffic classification. This requires a process of converting flow data collected on the network into a form that can be learned using RNN. To this end, through the traffic data preprocessing process, the learning data generation step makes the raw traffic data divided by each application into traffic split and splited network traffic set easy to learn. The RNN learning will then categorize the flow. Chapter 2 of this paper describes the system model, and Chapter 3 suggests deep RNN-based traffic analyzation sheme, and Chapter 4 conducts performance evaluation. Chapter 5 describes conclusions and future research plans.

II. System Model

We need to express a system model that is behind the proposed scheme.

III. Proposed deep RNN-based traffic analyzation sheme (DR-TAS)

This chapter describes traffic data preprocessing and deep RNN model for Proposed deep RNN-based traffic analyzation scheme.

A. traffic data preprocessing

The traffic data used to perform traffic classifications is a preprocessing packet capture (PCAP) files supplied by universitat politecnica de catalunya barcelonatech (UPC) [1] suitable for RNN learning. PCAP is a file that captures network packets and stores them in the form of PCAP files using programs such as Wireshark and TCPdump. PCAP provided for flow-based network traffic classification using RNN suggested in this letter need to be pre-processed into data sets classifiable in RNN traffic anlyzation. This requires a data pre-processing process to filter and shorten the PCAP files. The original PCAP file is approximately 59GB in size and has 769507 flows. In the packet_default.info file provided with the PCAP file, there is labeling for the traffic data. Labeling is divided into classes, such as the application type, protocol, and application name for the corresponding traffic flow. The labeling file gave us the correct label for ground truth in the flow-based traffic classification with RNN suggested in this letter.

For the data pre-processing process, eight types of applications were selected based on the number of flows. The top eight label names were Remote Desktop Protocol (RDP), Skype, SSH, Bittorrent, HTTP-face-Govert, HTTP-Doki, HTutube, and only the corresponding label selection flow. Only the payload of the application layer was filtered out of the selected flow internal packet, and the extracted payload data was placed and eight application layer Payload data files were generated. And for HTTP-facebook-Google, HTTP-Web, HTTP-Doki, HTTP-Youtube, we finally combined the label with the label Web to create a total of five data sets. Through the integration process, the application layer Payload data files of RDP, Skype, SSH, Bittorrent, and Web were completed. The data file has header information for each flow, and the payload separated by a # delimiter for each packet. That is, each file is a collection of flows with the same label, with packets having only a payload and the payload being separated by a packet. The payload is also a string, with a size of four bits per one

TABLE I
STATISTICAL INFORMATION OF EACH APPLICATION

	rdp	skype	ssh	bittorrent	http-web
Total number of filtered flows	153,349	2,041	38,831	96,222	21,715
Total number of packets(K)	6,876K	3,015K	173,911K	207,306K	47,136K
Average number of packets in a flow	44	1,477	461	2,154	2,483
Total packet size (GB)	131.3	12.0	321.7	2,170	698.4
Average packet size over all flows (KB)	20.0	4.2	18.8	11.3	15.9
Min packet size over all flows (B)	8.0	8.0	8.0	8.0	8.0
Max packet size over all flows (MB)	5.5	0.3	6.6	0.8	40.0

character. Figure 1 shows statistical information on the finished application layer Payload data file.

This step describes the process of converting LSTM into learning data using the application layer payload data generated in the previous step. A data set for learning will have flight data with 8,750 flows, 1,250 validation data, and 2,500 test data. Each label data equal to the number of each transaction, validation, test data set. Each application flow in the application layer payload data file goes through a conversion process into data for LSTM learning. The conversion job removes the head information that indicates the flow id, start time, and end time of each flow, and imports the data portion of the application layer only the payload in the flow unit and creates a data set. The LSTM models should have the same number of data input at a time to perform the learning in flow units. That is, a flow should have the same size packets. For this purpose, the user can specify as many packets per flow as P. In addition, one packet in the flow has a user-designed size, which can be set by the user to N, and it has a one-dimensional array size. The pixels of a packet's pixel have a 2^n bit size and a character-type value replaced by a floating point value. That is, the data in one flow will be the same size as the equation (1).

$$\mathrm{N_{flow}} = \mathrm{N} \quad \mathrm{x} \quad 2^{\mathrm{n}} \quad \mathrm{x} \quad \mathrm{P} \tag{1}$$

Fig.1 shows the payload of a packet in one of the completed flows in an two-dimentional image, a element of four bits in size, and a value between zero and fifteen floating point. In addition, due to the nature of the LSTM network structure, all flows should have the same number of packets.Because the length of sequence is set in advance, the number of packets per flow should be the same by that length. However, since not all Applicatoin flows can have the same number of packets, if the number of packets per flow is smaller than the N set, then the packet is generated by zero. Train, Validation and Test Label have labels for a total of five applications, which can be expressed as one-hot vectors of five lengths. A one-hot vector label having a value of only one element, can be defined as a label on which an index of one points to an application name. Table 2 shows the structure represented by the Train, Validation, and Test labels as on-hot vectors.

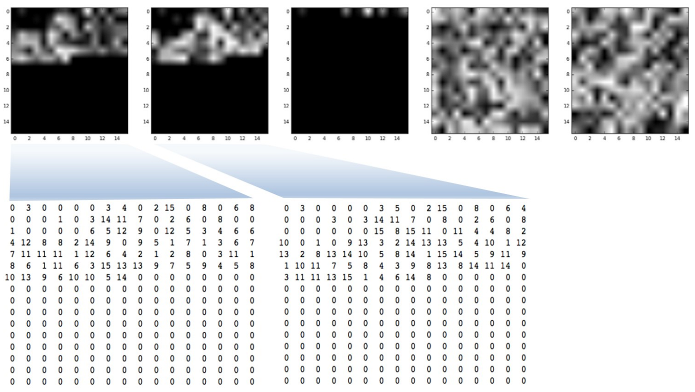

Fig. 1. Input features of flow packet

TABLE II
OUTPUT LABELS OF APPLICATION ONE-HOT VECTOR

Application	RDP	Skype	SSH	Bittorrent	Web
label	[1,0,0,0,0]	[0,1,0,0,0]	[0,0,1,0,0]	[0,0,0,1,0]	[0,0,0,0,1]

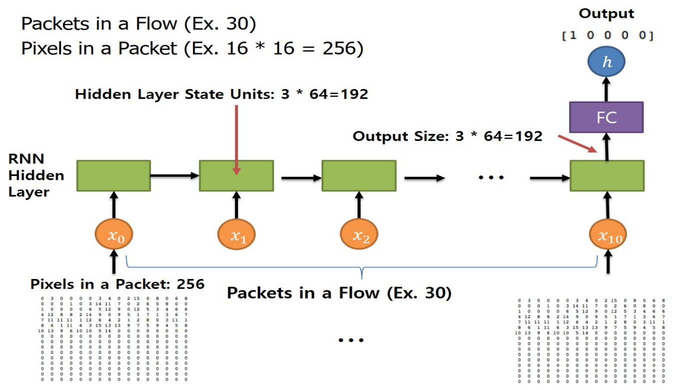
Packets in a Flow (Ex. 30)
Pixels in a Packet (Ex. 16 * 16 = 256)
Output
[1 0 0 0 0]
h
FC
Hidden Layer State Units: 3 * 64=192
Output Size: 3 * 64=192
RNN Hidden Layer
x_0
x_1
x_2
x_{10}
Pixels in a Packet: 256
Packets in a Flow (Ex. 30)

Fig. 2. Proposed deep RNN-based traffic analyzation sheme

B. Proposed deep RNN model

This subsection describes the LSTM models that will be used for flow based network traffic classification using the proposed LSTM. LSTM shows great performance for various natural language processing (NLP) issues, especially LSTM is able to learn sequential data over CNN, which is used more by Deep Learning. Therefore, we believe that the network traffic classification through LSTM will be accurate in analyzing the flow that contains sequential information of packets. Fig.2 shows the learning structure of the Proposed deep RNN-based traffic analyzation sheme used to classify traffic, and the learning model consists of a single layer. The time step for learning LSTM is to preset the number of packets throughout the flow to be learned. Time steps represent the number of inputs a flow enters sequentially from a single layer to an LSTM. Therefore, the form of learning data sets is represented in the train data set (number of flows, number of packets per flow, and payload size per packet), and the label data (number of flows, number of labels). The data sets of tests and validation are also inputted in the same form and learned through LSTM. The problem with the existing vanilla RNN is that of long-term dependency if the length of sequence is longer. Thus, this network architecture utilizes LSTM, a solution to long-term dependency problems.

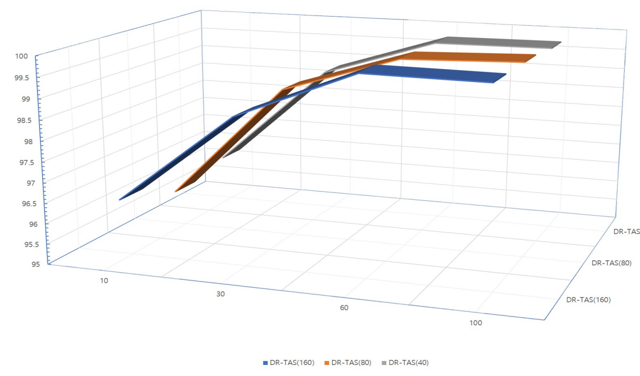

Fig. 3. Accuracy of deep RNN-based traffic analyzation scheme

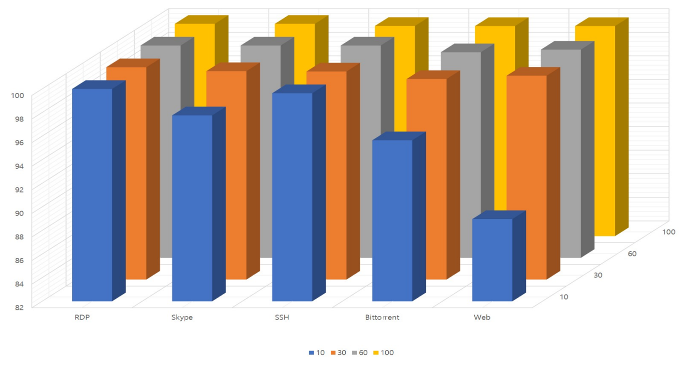

Fig. 4. Accuracy of DR-TAS for each target application

IV. Performance evaluation

In this section, we present the results of classification of traffic data through the DR-TAS and DC-TAS experiments, and compare the network traffic classification results of DR-TAS with the ones of DC-TAS.

A. Experiment environment

Our experiments were conducted on Ubuntu 16.04 LTS, using 32GB of RAM and two NVIDIA GTX 1080Ti 11GB. We also used Tensorflow 1.8 in Python 3.6 environment to configure the LSTM and CNN deep learning models. For the experiments, we have 2000 flows for each application in train data, and the number of packets per flow is set to 10, 30, 60, and 100. In addition, the payload size of each packet was set to 40, 80, and 160.

B. The overall accuracy of DR-TAS

For five applications, we examine the overall accuracy of the LSTM model with regard to various numbers of packets per flow and payload size of each packet, and the results are shown in Fig. 3. As shown in the figure, the more the number of packets per flow, the higher the accuracy, and the accuracy becomes high as the payload size increases. If particular, when the number of packets per flow is 100 and the payload size is 160, the overall accuracy is the highest (i.e., 99.85%). The lowest accuacy is 96.00% when the corresponding values are 10 and 40, respectively.

C. LSTM model evaluation for each application

By using confusion matrix, we also compare the performance of DR-TAS for each application. The Confusion matrix is the most widely used method for representing binary classification evaluation results. The Confusion matrix shows how well the deep-running model is categorized for each application. Figure x shows the Confusion Matrix, where the rows represent the negative class and the positive class, and the columns represent the predicted by negative and the predicted by positive. For example, TN (True Negative) refers to the case where the prediction result of the model is a negative class, which is actually a negative class. FP (False Positive) means that the model is predicted as a positive class, but actually it is a negative class. FN (False Negative) indicates that the model is predicted as a negative class, but the actual result indicates a positive class, and TP (True Positive) indicates a case where both the predicted result and the actual result are positive.

We calculate precision, recall and f1-score based on Confusion Matrix. The precision means the percentage of actual positive predictions out of the positive predicted class, and the equation of precision is as follows.

$$precision = \frac{TP}{TP + FP} \tag{2}$$

The recall is called the positive detection rate, and it means the percentage of the total positive class predicted by a positive class, and and the equation of recall is as follows.

$$recall = \frac{TP}{TP + FN} \tag{3}$$

Finally, f1-score means the harmonic mean of precision and recall, and it is defined as:

$$f1 - score = \frac{2 \times precision \times recall}{precision + recall} \tag{4}$$

The recall and precision are in conflict, and the f1-score is an index that shows the accuracy at once by integrating precision and recall.

Fig. 4 and 5 show the accuracy and the f1-score of DR-TAS for five applications, respectively. As shown in Fig. 4, the accuracy of each application increases as the number of packets per flow increases. Also in the case of f1-score, it can be seen that the value increases as the number of packets per flow. Is is noted that, in the case of Web and $Bittorrent$, the accuracy and the f1-score are not high particularly when the number of packets per flow are 10. ....

D. Comparison of DR-TAS and DC-TAS performance

We compare the accuracy of each application with regard to four sets of applications:

$$Set\ I = \{RDP, Skype\} \tag{5}$$
$$Set\ II = \{RDP, Skype, SSH\} \tag{6}$$
$$Set\ III = \{RDP, Skype, SSH, Bittorrent\} \tag{7}$$
$$Set\ IV = \{RDP, Skype, SSH, Bittorrent, Web\} \tag{8}$$

Figure 5 shows the accuracy comparison results for the number of applications of CNN and LSTM. Both CNN and LSTM show that accuracy is lowered when Bittorrent and

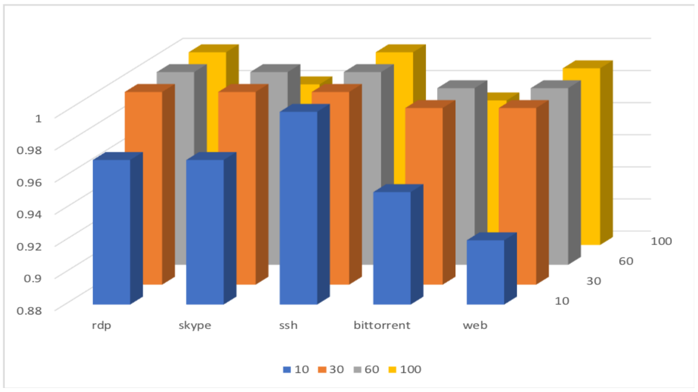

Fig. 5. f1-score of DR-TAS for each target application

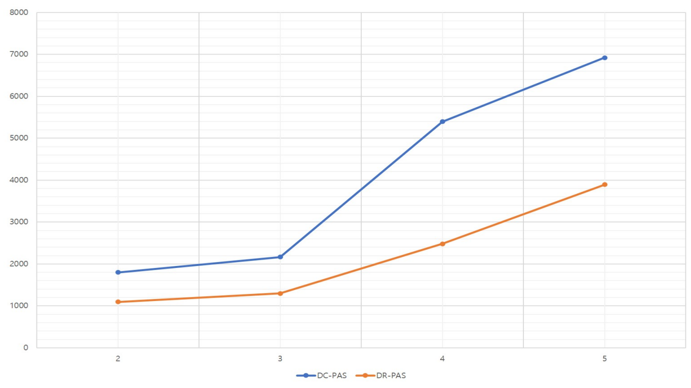

Fig. 7. Comparison on elapsed time of DR-TAS vs. DC-TAS

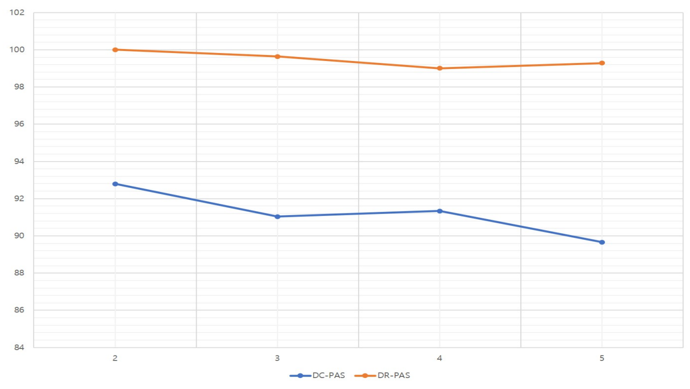

Fig. 6. Comparison on accuray of DR-TAS vs. DC-TAS

web are added. However, the overall accuracy of the LSTM is higher than that of CNN.

E. Comparison of CNN and LSTM elapsed time

We compare elapsed time from data preprocessing of CNN and LSTM to result prediction. The elapsed time is the sum of the preprocessing time for making the appropriate data for each model and the time taken to input the training data into the CNN and LSTM model. Figure 6 shows the elapsed time of CNN and LSTM, and shows the time taken to increase the number of applications as above. As the number of applications increases, the elapsed time increases. The elapsed time has increased because the amount of data to learn has increased. However, it can be seen that the elapsed time of 4 and 5 applications including *Bittorrent* and web data increases dramatically. The reason is that the two application data are larger than RDP, Skype, and ssh, so that the preprocessing process, which makes the data suitable for the model, takes more time. In addition, *Bittorrent* and *Web* are usually composed of image or video data, so the *Bittorrent* and *Web* data size is larger than the data size of other applications. However, in the case of LSTM, the elapsed time is less than half that of CNN as a whole. In the case of CNN, one packet is arbitrarily fetched from each application-specific flow in the preprocessing process, so that it takes a long time to read one flow. On the other hand, in the case of LSTM, since the flow data are generated for each application, the preprocessing time is shorter than that of CNN.

V. Conclusion

As a result of learning from the LSTM model using network traffic data as a unit of flow, its accuracy was over 99%. This confirmed that the classification was nearly 100%. Future research will explore the classification of network traffic through deployment in a real network and also how new packets, other than those already learned, will be classified as they enter the real network.

References

[1] Valentín Carela-Español, Tomasz Bujlow, and Pere Barlet-Ros: "Is Our Ground-Truth for Traffic Classification Reliable?", *In Proc. of the Passive and Active Measurements Conference (PAM'14),* Los Angeles, CA, USA, March 2014.

[2] Jinwan Park, "statistics signiture based application traffic classification," *Korea Communication Journal,* vol. 34, pp. 1234–1244, Nov. 2009.

[3] F. Risso, "Lightweight, Payload-Based Traffic Classification: An Experimental Evaluation," *IEEE International Conference on Communications 2008,*, 2008.

[4] Yann LeCun, "Deep Learning," *Nature International Weekly Journal of Science,*

[5] T. Mikolov, S. Kombrink, L. Burget, J. Cernocky and S. Khudanpur, "Extensions of recurrent neural network language model," *2011 IEEE International Conference on Acoustics, Speech and Signal Processing(ICASSP),* pp. 5528–5531, 2011.

[6] H. Sak, Andrew Senior and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," *Proceedings of the Annual Conference of the International Speech Communication Association(INTERSPEECH),* pp. 338–342, Jan. 2014.