Packet-based Network Traffic Classification Using Deep Learning

Hyun-Kyo Lim
Dept. of Interdisciplinary Program in Creative Engineering
Korea University of Technology and Education
Cheon-an, Korea
glenn89@koreatech.ac.kr

Ju-Bong Kim
Dept. of Computer Science Engineering
Korea University of Technology and Education
Cheon-an, Korea
jubong1992@gmail.com

Joo-Seong Heo
Dept. of Computer Science Engineering
Korea University of Technology and Education
Cheon-an, Korea
chil1207@koreatech.ac.kr

Kwihoon Kim
Dept. of Knowledge-converged Super Brain Convergence Research
Electronics and Telecommunications Research Institute
Daejeon, Korea
kwihooi@etri.re.kr

Yong-Geun Hong
Dept. of Knowledge-converged Super Brain Convergence Research
Electronics and Telecommunications Research Institute
Daejeon, Korea
yghong@etri.re.kr

Youn-Hee Han
Dept. of Computer Science Engineering
Korea University of Technology and Education
Cheon-an, Korea
yhhan@koreatech.ac.

***Abstract*— Recently, the advent of many network applications has led to a tremendous amount of network traffic. A network operator must provide quality of service for each application on the network. To accomplish this goal, various studies have focused on accurately classifying application network traffic. Network management requires technology to classify network traffic without the intervention of the network operator. In this study, we generate packet-based datasets through our own network traffic pre-processing. We train five deep learning models using the convolutional neural network (CNN) and residual network (ResNet) to perform network traffic classification. Finally, we analyze the network traffic classification performance of packet-based datasets using the f1 score of the CNN and ResNet deep learning models, and demonstrate their effectiveness.**

Keywords—network traffic classification, payload-based classification, deep learning, convolutional neural network, residual network

I. INTRODUCTION

Recently, the importance of network operation and management has been emphasized owing to the emergence of various services and applications. In particular, a wide variety of application services are provided through the network. Some services (e.g., video and voice services) require fast transmission. On the other hand, text services can provide adequate performance without fast transmission. In addition, peer-to-peer (P2P) services, such as BitTorrent, account for a significant proportion of the global Internet traffic, and thus have a significant impact on the overall network speed. Therefore, network operators try to provide smooth quality of service (QoS) by assigning different priorities to each service.

This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) and funded by the Ministry of Education (Nos. 2018R1A6A1A03025526 and NRF-2016R1D1A3B03933355).

* Corresponding Author: Prof. Youn-Hee Han

Network traffic classification [1–3], for providing different QoS according to each application, is being actively researched. Research in the field of machine learning has progressed actively, and machine learning has been adopted in various areas [4]. Thus, network traffic classification using machine learning has been researched extensively [5-10]. In particular, deep learning models have been recently studied for network traffic classification since their performance is known to be usually better than that of other machine learning algorithms [9, 10]. The existing network traffic classification methods [9] based on deep learning models, however, usually perform classification using header information as well as packet payloads as learning features. Such an approach poses limitations when the header information is included in a limited dataset collected within a local network and the deep learning model training is performed with such dataset. In a real network, which extends beyond local limits, it is difficult to perform classification well using the previously trained model, since the header information usually varies in the real network.

In this study, we develop a traffic data preprocessing method to create deep learning datasets by using only packet payloads. We train the deep learning models with the datasets and evaluate the performance of the models. Our goal of achieving deep learning using only payload information is to enable the learned model to fit unseen packets well. A model with strong generalization ability can fit the whole data sample space well. Excluding header information that has a typical structure and inconsistent values helps to improve such generalization performance.

We treat the payload of packets as image data, which have been actively used as deep learning datasets in the artificial intelligence research area; we apply the representative deep

learning models to the network traffic classification problem and investigate their performance. We use the Convolutional Neural Network (CNN) and Residual Network (ResNet) as the deep learning models to classify payload-based network traffic. We also execute a model tuning procedure to find the optimal hyper-parameters of each deep learning model and enhance the performance of network traffic classification. Then, we compare the performance of the five above-mentioned deep learning models on the basis of the f1 score and demonstrate the effectiveness of the models.

II. Related Work

Many studies have focused on network traffic classification technologies. Classical studies involve rule-based or statistical correlation-based network traffic classification. Machine learning has been researched extensively and studies on network traffic classification using machine learning have been actively conducted [5-10].

Shafiq et al. [5] attempted to classify network traffic by machine learning using different kinds of datasets. They used the three ML algorithms, multi-layer perceptron, C4.5 decision tree, and support vector machine. The results showed that the C4.5 decision tree algorithm performed better than the other two algorithms. Singh et al. [6] used unsupervised machine learning approach for network traffic classification. The unsupervised K-means and the expectation maximization algorithm were used to cluster the network traffic application based on the similarity between them.

Network traffic classification using deep learning is a method of automatically classifying packets without human intervention. Wang et al. [9] used a CNN model to classify malware traffic and general traffic. First, if a 5-tuple (source IP/port, destination IP/port, and protocol) is the same among the packets, one flow is defined as one dataset. The constructed dataset is used to train the CNN model to classify malware traffic and general traffic. The accuracy of classifying malware traffic and general traffic was very high.

In this study, we perform the network traffic classification based on deep learning models using only the payload data (including application layer headers) of the application layer. The source and destination IP addresses and port numbers of the IP and TCP/UDP headers are becoming a hindrance to network traffic classification. Therefore, such header information of the packets is ignored for generating learning datasets.

III. Dataset Preprocessing

In this study, the labeled packet capture (PCAP) traces provided by the UPC's Broadband Communications Research Group [11] is used for training and testing the deep learning models. The PCAP trace file captures and stores network packets using programs such as Wireshark and tcpdump. The size of the original provided PCAP trace file is around 59 GB, with a total of 769,507 flows in the file. An additional information file provided with the PCAP trace file includes the labels (application name) of the traffic data and the ground truth for the prediction based on deep learning.

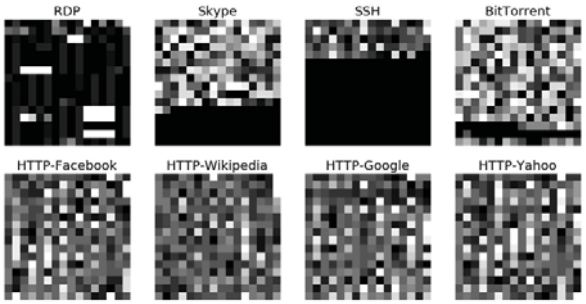

Fig. 1. Images of network packets extracted from PCAP

For the preprocessing of the supplied data, based on the number of flows, 8 types of applications with more than 1000 flows are selected. The eight applications' label names are the Remote Desktop Protocol (RDP), Skype, SSH, BitTorrent, HTTP-Facebook, HTTP-Google, HTTP-Wikipedia, and HTTP-Yahoo. The application layer payload data of the selected applications are filtered and extracted, and the learning data are generated using the extracted payload data.

Our preprocessing to convert the above-mentioned application layer payload data into learning data is suitable for deep learning models. The overall packet-based learning data for each application are generated by arbitrarily extracting packets of eight applications (RDP, SSH, Skype, BitTorrent, Facebook, Wikipedia, Google, and Yahoo) from the application layer payload data. In each application, 10,000 random packets were extracted (i.e., a total of 80,000 learning data). All bits in the payload data of a packet are divided by 4 bits and grouped into one pixel of imaged data. Therefore, one pixel of the imaged data represents the decimal numbers 0 (= 0 × 0000) to 15 (= 0 × 1111). According to the pre-defined image size values, 36 (= 6 × 6), 64 (= 8 × 8), 256 (= 16 × 16), and 1024 (= 32 × 32), the pixels of one image data are taken from the beginning of each packet, and the size of one image data is readjusted to 36, 64, 256, and 1024 pixels. Fig. 1 shows the case of the imaged packet data of 256 pixels for an arbitrary packet of each application. If the extracted payload size is smaller than the pre-defined size, the image is adjusted by zero-padding to match the pre-defined size.

Finally, for the target data of eight applications, each label is expressed as a one-hot vector with eight lengths. A one-hot vector is a 1 × 8 matrix with all 0s and a single 1 used to distinguish the label representing an application.

IV. Deep Learning Models

This section describes the deep learning models used to classify network traffic. The selected deep learning models in this study are CNN and ResNet. These models are commonly used for information extraction, sentence classification, face recognition, and image classification. CNN extracts characteristics of data and grasps patterns of features. Therefore, both CNN and ResNet are used for classification based on imaged packet-based data generated through preprocessing.

A. Convolutional Neural Network Architecture

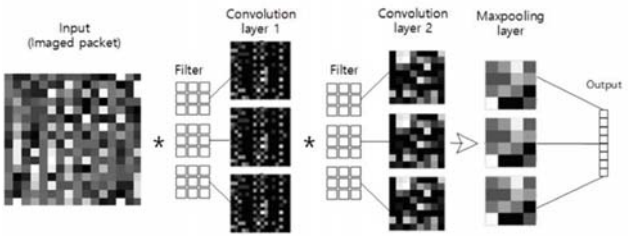

Fig. 2. Two-layer CNN learning model architecture

As shown in Fig. 2, the CNN model architecture [12] is composed of the input layer, convolution layer, pooling layer, and fully connected layer. The input layer uses the pixelized N × N (N = 6, 8, 16, and 32) image data converted from the payload of a packet. Then, the feature of each packet data is convolved through the kernel of two convolution layers, and the output is generated through a filter and activation process. The pooling layer involves the process of reducing the size of the output through the convolutional process. It simply reduces the size of the data, cancels noise, and provides consistent features in fine detail. Finally, the fully connected layer extracts the prediction value according to the last eight classes by activation.

B. Residual Network Architecture

Unlike the traditional CNN, ResNet [13] involves a unique concept called shortcut connection. A shortcut connection is added to the existing CNN model structure, and this shortcut is directly connected without any other parameters. A shortcut connection involves adding a new type of network to an input value, and the newly added network can achieve better performance while maintaining the performance of the existing learned network as much as possible.

As shown in Fig. 3, when one packet of learning data enters the input, it passes through one convolution layer and one MaxPooling layer. The result from the MaxPooling layer is then used as an input to a 3-convolution group. Furthermore, the MaxPooling layer result is used as the input of the newly added shortcut network. Thereafter, the outputs from the 3-convolution group and the shortcut network are summed. The result of the addition is passed through the activation function again, and it is used as the input of the second 3-convolution group and the shortcut network. After three more iterations of the 3-convolution group process, the output after addition is sampled through the average pooling layer, and the result of the final classification is obtained.

In terms of computation with ResNet, only one additional operation needs to be considered. Deep networks can be easily optimized through shortcut connections and can improve accuracy with increasing depth.

V. Model Tuning

Each of the models has various hyper-parameters that determine the network structure (e.g., number of filters) and how the network models are trained (e.g., type of optimizer). The performance of a model can vary considerably according to the selected set of hyper-parameters. Toward this end, we use grid search [14] as method to find hyper-parameters optimized for each deep learning model according to the datasets. The grid search method searches for the best hyper-parameter for a dataset by trying every possible combination of hyper-parameters based on the dataset. We also verify the validity of the model by performing k-fold cross-validation [15] in addition to finding the optimal hyper-parameters. In k-fold cross-validation, the dataset is randomly partitioned into k equal sized sub-datasets. Of the k sub-datasets, a single sub-dataset is retained as the validation data for testing the model, and the remaining k-1 sub-datasets are used as training data. The cross-validation process is then repeated k times, with each of the k sub-datasets used exactly once, as is the case for the validation data. The k results can then be averaged to produce a single estimation. The advantage of this method over repeated random sub-datasets is that all observations are used for both training and validation, and each observation is used for validation exactly once.

For each of the 8 applications, 10,000 packets are randomly organized into a single learning dataset. We choose 36, 64, 256, and 1024 pixels as the imaged payload sizes of a packet, so that the shapes of the total datasets are (80000, 6, 6, 1), (80000, 8, 8, 1), (80000, 16, 16, 1), and (80000, 32, 32, 1), respectively. There is one channel in an imaged payload, so that the channel corresponds to the grey level.

For CNN and ResNet models, seven important hyper-parameters, namely, 1) number of filters, 2) kernel size, 3) kernel initializer, 4) padding method, 5) activation type, 6) optimization type, and 7) batch size, are selected. The number of filters represents the number of the output filters per convolution layer. The kernel size is the size of the kernel used in one filter. The kernel initializer represents the strategy to

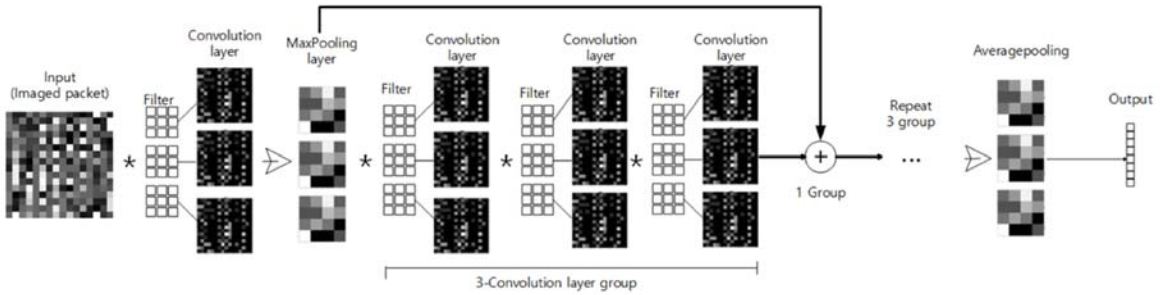

Fig. 3. ResNet learning model architecture

initialize the weight vectors of each layer. The padding method designates the method to control the size of output filtered by one convolution layer. The activation type represents the type of activation function that produces non-linear and limited output signals inside the CNN and ResNet models. The optimization type designates the optimization algorithm to tune the internal parameters so as to minimize the CNN and ResNet loss function. Finally, the batch size refers to the number of training examples utilized in one optimization iteration. Table I lists the values used to perform the grid search for the hyper-parameters used in the CNN and ResNet models.

TABLE I. TABLE I. THE GRID SEARCH HYPER-PARAMETER SET FOR THE CNN AND RESNET MODELS

	Values of hyper-parameters
number of filters	Payload size/2
kernel size	{3 × 3, 5 × 5, 7 × 7}
kernel initializer	{normal, uniform, glorot_uniform}
padding method	{valid, casual, same}
activation type	{tanh, relu, softmax}
optimization type	{adam, rmsprop}
batch size	{1, 10, 100}

For each size of the imaged payload used as input data in the CNN and ResNet models, Tables II and III represent the optimal hyper-parameter values found through the cross-validated grid search process. For producing the best performance in the CNN and ResNet models, as shown in the tables, the number of filters increases when the imaged payload size becomes large. However, we cannot find the optimal kernel size pattern to produce the best model performance. On the other hand, the optimal padding method and activation type are always "same" and "softmax," respectively. It indicates that the model provides its best performance when the output of each layer in the models has the same length as the input of the layer, and the output values are activated by a non-linear logistic function. In most cases, we can also ascertain that the "adam" optimizer produces the best model performance.

TABLE II. THE OPTIMA HYPER-PARAMETER VALUES FOR THE CNN (U: UNIFORM, N: NORMAL, G: GLOROT UNIFORM)

	Imaged payload size (unit: pixels)			
	36	64	256	1024
number of filters	18	32	256	512
kernel size	3 × 3	5 × 5	5 × 5	3 × 3
kernel initializer	G	U	U	U
padding method	same	same	same	same
activation type	softmax	softmax	softmax	softmax
optimization type	adam	adam	adam	adam
batch size	100	100	10	10

TABLE III. THE OPTIMA HYPER-PARAMETER VALUES FOR THE RESNET (U: UNIFORM, N: NORMAL, G: GLOROT UNIFORM)

	Imaged payload size (unit: pixels)			
	36	64	256	1024
number of filters	18	32	256	512
kernel size	3 × 3	7 × 7	5 × 5	7 × 7
kernel initializer	G	G	U	G
padding method	same	same	same	same
activation type	softmax	softmax	softmax	softmax
optimization type	adam	adam	adam	rmsprop
batch size	100	100	100	100

I. EXPERIMENTAL EVALUATION

In this section, we compare the model prediction performance across the two models.

A. Experimental Environment

Our experiments are executed on Ubuntu 16.04 LTS with 32 GB of RAM and two GPU cards (NVIDIA GTX 1080Ti 11 GB). For the experimental implementation, we use Tensorflow GPU 1.8 and Keras 2.2.0 operated with Python 3.6. That is, the five models are constructed, trained, and tested by Keras using the Tensorflow-gpu backend.

The 3-fold cross-validated grid search is first performed for one dataset (that is, k = 3) and the optimal hyper-parameters are found through each model tuning. The model training is performed using the found optimal hyper-parameters for each dataset, as listed in Tables II and III. The number of learning epochs is set to 200 for such model training.

B. Performance Metrics

An unambiguous and thorough way to present the prediction results of a deep learning model is to use a confusion matrix. However, the accuracy can be misleading when the number of application labels is imbalanced. A model can predict the label of the majority application for all predictions and achieve high classification accuracy, and the model is not useful in the problem domain. For every application label, therefore, we convert the multi-class confusion matrix into the binary confusion matrix.

TABLE IV. TABLE III. THE BINARY CONFUSION MATRIX

		Predictive	
		Positive	**Negative**
Actual	**Positive**	True Positive: TP	False Negative: FN
	Negative	False Positive: FP	True Negative: TN

In the binary confusion matrix described by Table IV, the true positive (TP) indicates the cases in which the actual label is positive (that is, RDP) and the model prediction is also correctly identified as positive. The false negative (FN) indicates the cases in which the actual label is positive, but the model prediction is

incorrectly identified as negative. The false positive (FP) indicates the cases in which the actual label is negative (that is, not RDP), but the model prediction is incorrectly identified as positive. Finally, the true negative (TN) indicates the cases in which the actual label is negative and the model prediction is also correctly identified as negative.

To overcome the problem of accuracy measurement, we compute the additional measurements, namely, recall, precision, and f1-score, to evaluate the five models [16]. They are defined by using the binary confusion matrix as follows:

$$Recall = \frac{TP}{TP+FP} \quad (1)$$

$$Precision = \frac{TP}{TP+FN} \quad (2)$$

$$f1-score = \frac{2\times Precision\times Recall}{Precision+Recall} \quad (3)$$

The f1-score represents the harmonic mean of precision and recall, and indicates the classification performance of a deep learning model relatively accurately. It is expressed in the range of 0 to 1, where the best value is 1. We compute the recalls, precisions, and f1-score of all eight application labels, and then finally average them to get a single overall f1-score measurement for performance comparison of the five models.

C. Experiments Results

The first experiment is performed to compare the performance of the CNN and ResNet models with the packet-based datasets. Fig. 4 shows the comparison of the overall f1-score for the two models with the eight application labels in terms of the four imaged payload sizes. Intuitively, as shown in the figure, a larger payload size of a dataset can enhance the performance of two deep learning models, since more information is trained by the two models. It is noted that, overall, the f1-score of the CNN model is higher than that of the ResNet model when the payload size of the dataset is small. However, as the payload size increases, the f1-score of the ResNet model becomes larger than that of the CNN model, and the gap between the two models' f1-score becomes bigger. The reason for this result may be attributed to the fact that the ResNet learning model is more complex than the CNN model, and the ResNet model can improve its generalization capability to a greater extent when the payload size is large.

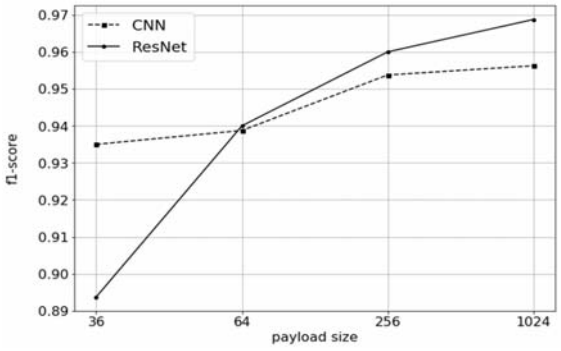

Fig. 4. Performance comparison of the CNN and ResNet models

II. Conclusions

In this paper, we classify network traffic using two deep learning models: CNN and ResNet. In order to achieve this, the imaged packet data are generated through our own pre-processing method, and the packet-based dataset is created by gathering such imaged packets for each of eight applications. We also execute the cross-validated grid search to find the optimal hyper-parameters that maximize the performance of the deep learning models. Through our intensive experiments, we can know that the deep learning models can classify the network traffic fairly well, and ResNet outperforms the CNN model. Therefore, when using the ResNet model, it is possible to classify network traffic with high accuracy and to provide better network QoS.

Future works will classify network traffic using various deep learning models. We will also study packet-based as well as flow-based network traffic classifications.

References

[1] P. Gupta and N. McKeown, "Algorithms for packet classification," *IEEE Network: The Magazine of Global Internetworking*, vol. 15, 2001, pp. 24–32

[2] L. Li and K. Kianmehr, "Internet traffic classification based on associative classifiers," *IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*, 2012, pp. 263–268.

[3] Y. Meidan, M. Bohadana, A. Shabtai, J. D. Guarnizo, M. Ochoa, N. O. Tippenhauer and Y. Elovici, "Profiliot: A machine learning approach for IoT device identification based on network traffic analysis," *Proceedings of the Symposium on Applied Computing*, 2017, pp. 506–509.

[4] R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrad-solano and O. M. Caicedo, "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities," *Journal of Internet Services and Applications*, vol. 9, 2018.

[5] M. Shafiq, X. Yu and D. Wang, "Network traffic classification using machine learning algorithms," *Advances in Intelligent Systems and Computing*, vol. 686, 2018, pp. 621–627.

[6] H. Singh, "Performance analysis of unsupervised machine learning techniques for network traffic classification," *2015 Fifth International Conference on Advanced Computing & Communication Technologies*, 2015, pp. 401–404.

[7] M. R. Parsaei, M. J. Sobouti, S. R. Khayami and R. Javidan, "Network traffic classification using machine learning techniques over software defined networks," *International Journal of Advanced Computer Science and Applications*, vol. 8, 2017.

[8] C. Yu, J. Lan, J. Xie and Y. Hu, "QoS-aware traffic classification architecture using machine learning and deep packet inspection in SDNs," *Procedia Computer Science*. vol. 131, 2018, pp. 1209–1216.

[9] W. Wang, M. Zhu, X. Zeng, X. Ye and Y. Sheng, "Malware traffic classification using convolutional neural network for representation learning," *2017 International Conference on Information Networking (ICOIN)*. 2017, pp. 712–717.

[10] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas and J. Lloret, "Network traffic classifier with convolutional and recurrent neural networks for internet of things," *IEEE Access*. vol. 5, 2017.

[11] V. Carela-Espanol, T. Bujlow and P. Barlet-Ros, "Is our ground-truth for traffic classification reliable?," *In Proceeding of the Passive and Active Measurements Conference (PAM'14)*, Los Angeles, CA, USA, 2014, 8362.

[12] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Advances in Neural*

Information Processing Systems 25 (NIPS 2012), vol. 1, 2012, pp. 1097–1105.

[13] K. He, X. Zhang, S. Ren and J. Sun, "Deep residual learning for image recognition," *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778

[14] F. Pedregosa, G. Varoquaux and A. Gramfort, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol.12, 2011, pp. 2825–2830

[15] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, vol. 2, 1995, pp. 1137–1143.

[16] D. M. W. Powers, "Evaluation: From precision, recall and f-measure to ROC, informedness, markedness & correlation," *Journal of Machine Learning Technologies*, vol. 2, 2011, pp. 37–63.