Computer Science and Information Systems 19(1):165–184 https://doi.org/10.2298/CSIS200424038K

Deep RNN-Based Network Traffic Classification Scheme in Edge Computing System

Kwihoon Kim[1], Joohyung Lee[2], Hyun-Kyo Lim[3], Se Won Oh[4], and Youn-Hee Han[5⋆]

[1] Department of Artificial Intelligence Convergence Education,
Korea National University of Education, Cheongju 28173, South Korea
kimkh@knue.ac.kr
[2] Department of Software, Gachon University,
Seongnam 13120, South Korea
j17.lee@gachon.ac.kr
[3] Interdisciplinary Program in Creative Engineering,
Korea University of Technology and Education, Cheonan 31253, South Korea
glenn89@koreatech.ac.kr
[4] Electronics and Telecommunications Research Institute,
Daejeon 34129, South Korea
sewonoh@etri.re.kr
[5] Future Convergence Engineering, Korea University of Technology and Education,
Cheonan 31253, South Korea
yhhan@koreatech.ac.kr

Abstract. This paper proposes a deep recurrent neural network (RNN)-based traffic classification scheme (deep RNN-TCS) for classifying applications from traffic patterns in a hybrid edge computing and cloud computing architecture. We can also classify traffic from a cloud server, but there will be a time delay when packets transfer to the server. Therefore, the traffic classification is possible almost in real-time when it performed on edge computing nodes. However, training takes a lot of time and needs a lot of computing resources to learn traffic patterns. Therefore, it is efficient to perform training on cloud server and to perform serving on edge computing node. Here, a cloud server collects and stores output labels corresponding to the application packets. Then, it trains those data and generates inferred functions. An edge computation node receives the inferred functions and executes classification. Compared to deep packet inspection (DPI), which requires the periodic verification of existing signatures and updated application information (e.g., versions adding new features), the proposed scheme can classify the applications in an automated manner. Also, deep learning can automatically make classifiers for traffic classification when there is enough data. Specifically, input features and output labels are defined for classification as traffic packets and target applications, respectively, which are created as two-dimensional images. As our training data, traffic packets measured at Universitat Politecnica de Catalunya Barcelonatech were utilized. Accordingly, the proposed deep RNN-TCS is implemented using a deep long short-term memory system. Through extensive simulation-based experiments, it is verified that the proposed deep RNN-TCS achieves almost 5% improvement in accuracy (96% accuracy) while operating 500 times faster (elapsed time) compared to the conventional scheme.

Keywords: RNN, Traffic Classification, Edge Computing, Cloud Computing.

⋆ Corresponding author

1. Introduction

To realize advanced network management, user service, and security functions according to various application traffic, service providers are required to design effective ways to inspect and identify their application traffic. Aiming for this goal, the deep packet inspection (DPI), which is a type of network packet filtering technique, has been a widely deployed approach for many years; it examines packet payloads to identify application traffic. Specifically, the DPI technique utilizes unique byte patterns (e.g., headers, data protocol structures and the payload of the message) as signatures to detect the application type. Because most recent applications are frequently updated to add new features with different versions, an accurate DPI system must periodically verify and update existing signatures, which sometimes requires much human intervention. Further, the manual task of application traffic generation and verification on multiple platforms and updated applications is highly tedious and error-prone [39,11,3,27].

This limitation has recently motivated research to establish lightweight and automated methods for classifying application traffic [35,2]. Recently, owing to the breakthroughs made by deep learning technique in various typical algorithms including deep multilayer perceptron (MLP), convolutional neural network (CNN), recurrent neural network (RNN), and long short-term memory (LSTM) [28,36,20,43], there have been broad use cases in the area of image classification (e.g., almost 98% accuracy achieved in image classification). According to Lecun et al. [19], the deep learning technology performed better in image classification than classical machine learning algorithms such as support vector machines (SVM) and Random Forest (RF). Despite its practical popularity in deep learning techniques, there has been only a limited number of research works on the automated classification of application traffic. In particular, the authors of [43] first applied and proposed a deep CNN-based traffic classification system in order to detect malware applications. However, because of the CNN's own characteristics, which are generally used to handle batch data and not for streaming data (i.e., time-series analysis), there is still room to improve its accuracy by considering time-varying packet payload patterns depending on the type of application, which inspired our work [17,14,5,15,32,16,33].

In this paper, a deep RNN-based traffic classification scheme (deep RNN-TCS) is proposed by adopting the RNN technique, which is suitable for training on streaming data. CNN is good at classifying the image data, especially such as data with shift invariant characteristics. On the other hand, RNN is good at classifying the time series data. Because network traffic flows through time series, it is appropriate to classify using RNN. The proposed deep RNN-TCS provides lightweight and automated classification of application traffic. Specifically, a novel learning platform is designed suited for detecting time-varying application traffic where input features and output labels are mapped to traffic packets and target applications, respectively. A hybrid edge computing and cloud computing architecture is considered. Here, a cloud server collects and stores output labels corresponding to the application packets. Then, it trains those data and generates inferred functions. An edge computation node receives the inferred functions and executes classification. From input features, multiple two-dimensional square matrices for sequential flows in preprocessing are created and trained in a stacked RNN model. As our training data, traffic packets measured at Universitat Politecnica de Catalunya Barcelonatech were utilized. Accordingly, the proposed deep RNN-TCS is implemented using a deep long short-term memory (LSTM) system. Through extensive simulation-based experiments, it

is revealed that the proposed scheme improves accuracy by almost 5% (96% accuracy) while operating 500 times faster (elapsed time) compared to the conventional scheme over five types of applications from the produced inferred function. In this study, network traffic classification is performed using only the payload excluding header information among network packet information. Recently, many IoT and mobile devices use private or dynamic IP addresses and changeable port numbers. So the classification of network traffic based on packet header information is no longer accurate [45]. The payload-based network traffic classification can solve the problem. The contributions of this paper can be summarized as follows.

- In our scheme, by applying the RNN mechanism to the traffic classification problem, we design new input features of the traffic payload data as the image data with a two-dimensional fixed-size matrix, so that only payload of packets, excluding TCP/IP headers, are used for training and inference data.
- We deploy the proposed deep RNN-TCS by considering a hybrid edge computing and cloud computing architecture is considered. Here, the cloud computing acts as a learner, which collects and stores output labels corresponding to the application packets received from PC clients and creates inferred functions for classification through a deep-learning process. Then, those inferred functions (i.e., deep learning model) are delivered to the edge computing. Correspondingly, the edge computing performs classification of application packets without output labels by using the deep-learning model delivered from the cloud computing.
- Through extensive simulation-based experiments, it is verified that the proposed deep RNN-TCS achieves almost 5% improvement in accuracy (96% accuracy) while operating 500 times faster (elapsed time) compared to the conventional scheme.
- Our research is not limited to this vanilla RNN. There are RNNs that have been modified recently, and it is easy to apply to reflect modified RNNs. Later, it will be the future work to apply the revised RNN to improve performance.

We can thus develop a practical deep recurrent neural network-based traffic classification scheme. Section II explains the related work of network traffic classification problem. Section III presents an overview of the proposed system model. In Section IV, we formulate the problem as a deep learning model and present the novel deep RNN-TCS. Numerical results and performance analysis are explained in Section V. We summarize and conclude this work in Section VI.

2. Related work

Recently, to solve each problem in various domains, such as smart homes, airport gate assignments, and the traveling salesmen, rule-based algorithms such as daily activation recognition, and classical machine learning algorithms such as the Support Vector Machine (SVM) and the Random Forest (RF) are being studied [24,7,6]. Classical algorithms such as the Principal Component Analysis (PCA), the Broad Learning System (BLS) techniques, a new performance degradation prediction method, and a genetic and ant colony adaptive collaborative optimization are being studied to address abnormal detection issues in manufacturing areas such as the Fault Diagnosis and the Prognostic and Health Management (PHM) [49,48,8]. For the network domain case, many researches have focused

on network traffic classification methods. Existing researches include rule-based network traffic classification method.

Recently, researches on network traffic classification method using good performance deep learning models have been actively performed [13,45]. Network traffic classification using deep learning is a method of automatically classifying packets without human intervention. Existing rule-based network traffic classification is a method of classifying packets having the network according to predefined rules [21,29,40,30]. For example, the classification methods use the header of the network packets. Therefore, rule-based network traffic classification is conducted on the basis of IP address and port number of the packet header. Li et al. proposed an approach to reduce the dependency on packet header information [21,22,38]. Using this approach, they found that their packet-shaping device uses the HTTP and TLS-handshake fields in their matching rules but only for the first packet in each direction. If there is similar information in the header information of the incoming packet compared to the header information found in the first packet, the incoming packet is classified as a packet of the same type. Although there is less dependence on the IP and port number of the packet, the method of classifying subsequent packets using the header information of the first packet still depends on the header information. However, since the rule-based network traffic classification method is highly dependent on the header information (Source IP / Port number, Destination IP / Port number), network traffic classification methods such as Correlation-based and payload-based methods have been studied. Correlation-based network classification classifies datasets by selecting packets with high correlation between traffic packets considering correlation between network traffic [47,18,9]. Zhang et al have shown that there is a strong correlation between flow size and rate [46]. The flow of application used by the user has a certain size and rate [47]. Also, user behavior might have an effect on large flows. Erman et al. consider the problem of traffic classification in the core network [9].

The packet classification at the core network is challenging because only partial header information about the flow are available. So, they use only unidirectional flow records. Specifically, they propose and evaluate a clustering-based framework for classifying network traffic using only unidirectional flow statistics. And their work is facilitated by recent full-payload Internet packet traces [22]. As the research on payload-based network traffic classification is studied, network traffic classification methods using machine learning and deep learning are being studied variously [43,9,31,44,12,37,25,26,10]. Haffner et al. used a variety of traditional machine learning techniques to compare and analyze packet payloads [12]. It reduced the amount of computation required when generating payload-based datasets. Toward this end, they used only the first few bytes of unidirectional traffic data and unencrypted TCP data. Specifically, they used NB, AdaBoost, and MaxEnt for traffic classification. AdaBoost outperforms NB and MaxEnt, yielding an overall precision of 99% with an error rate within 0.5%. Shafiq et al. used to classify network traffic by various machine learning algorithms using different kinds of datasets [37]. They used the three machine learning algorithms, multi-layer perceptron (MLP), C4.5 decision tree, and support vector machine (SVM).

However, recent developments in computing resources have led to a significant advance in deep learning fields that can be applied to network traffic classification. Especially, as the CNN and RNN models in the deep learning model are developed, they can be easily applied to the classification of network traffic. Wang et al. classified malware traf-

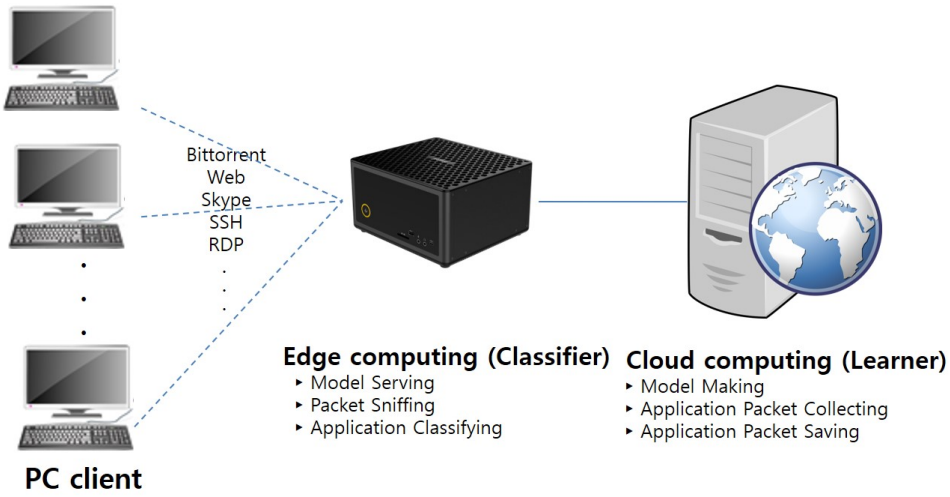


Fig. 1. Proposed system model for applying the proposed scheme

fic and normal traffic by using CNN [43]. To generate training set, header information of packet was extracted using DPI tool. Based on the 5-tuple (source IP / port number, destination IP / port number, protocol) of the extracted header information, flow-based dataset and session-based dataset is generated. The generated flow and session-based datasets again generate 28x28 training sets for each packet through an imaging process suitable for the CNN model. The trained CNN model using flow-based and session-based datasets is 100% accurate for malware traffic and normal traffic classification. Lopez-Martin et al. used to classify the network traffic using a combined CNN and LSTM [25]. The dataset is extracted from the packet headers of the network traffic and learns the dataset using a model that combines the single-layer CNN with LSTM, CNN, and LSTM. However, most network traffic classification methods that use in deep learning use the IP, port number, and MAC address of the packet header information as a feature of the training set. In this paper, a preprocessing process that extracts only the payload of packets from network traffic is implemented, and detailed comparison and analysis of the layers of CNN and LSTM are provided. Aceto et al. and Wang et al. utilizes various models of deep learning (CNN, LSTM, SAE, MLP) to classify application of network traffic using payload data as well as header information. Since the IP and port numbers, which are some information in the TCP/IP headers, are changed dynamically, they have the bad effects when they are used for classifying packets [41,1].

3. System Model

Multiple personal computer (PC) clients are considered that generate traffic while executing an application where they are connected to a cloud computing via an intermediate node called an “edge computing”. Accordingly, a hybrid edge computing and cloud computing architecture is considered. Here, the cloud computing acts as a learner, which collects and stores output labels corresponding to the application packets received from

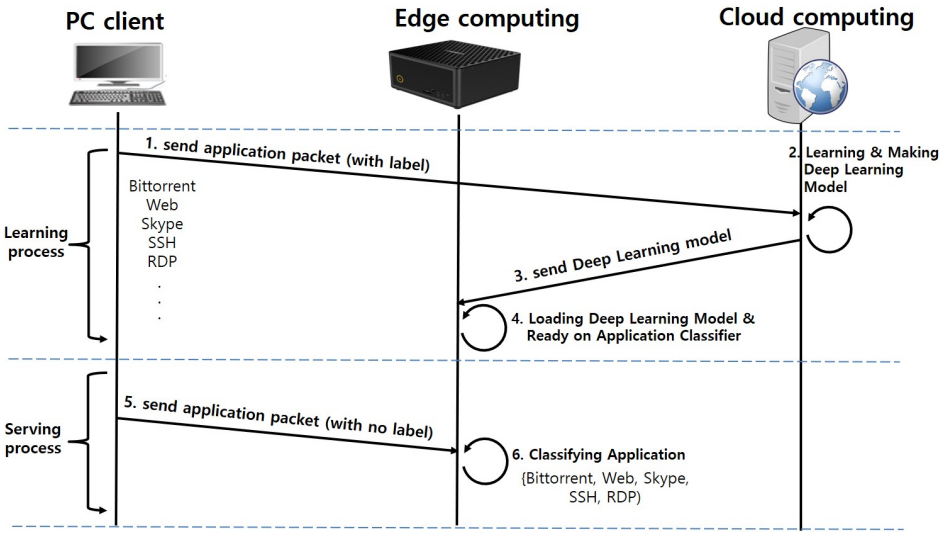
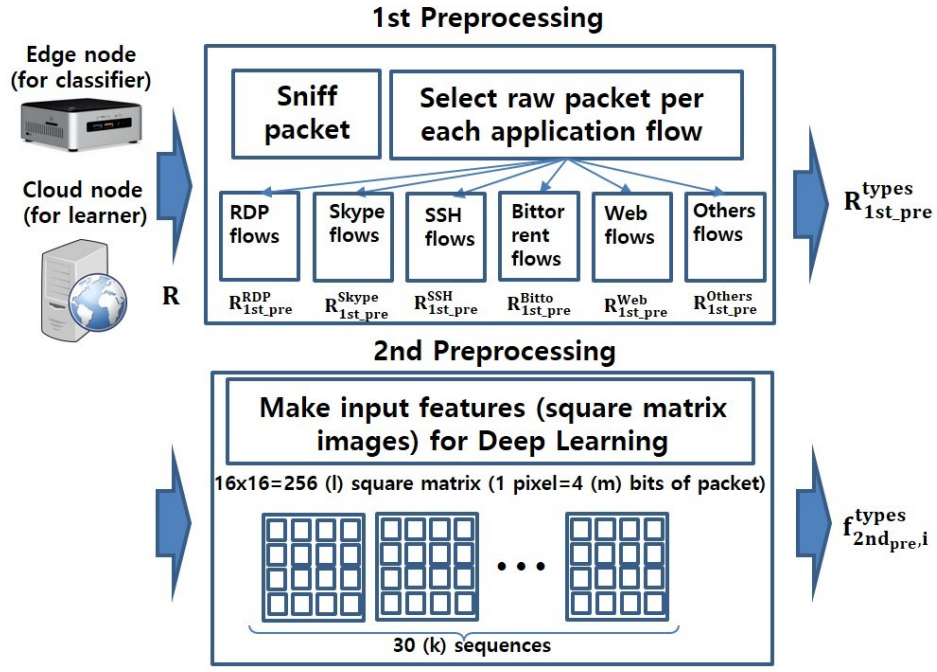


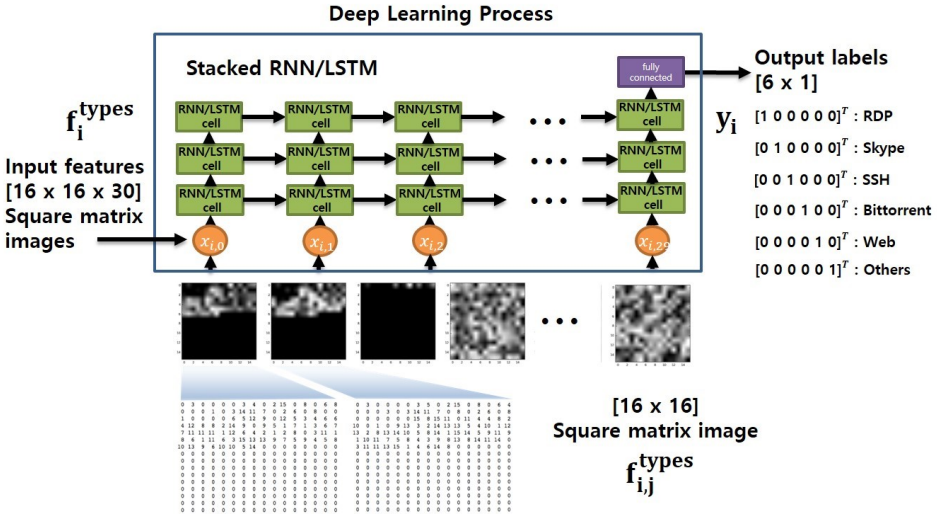
Fig. 2. Proposed procedures for applying the proposed scheme

PC clients and creates inferred functions for classification through a deep-learning process. Then, those inferred functions (i.e., deep learning model) are delivered to the edge computing. Correspondingly, the edge computing performs classification of application packets without output labels by using the deep-learning model delivered from the cloud computing. Fig. 1 shows a detailed system model and procedures of the proposed deep RNN-TCS.

As Fig. 2 is shown, detailed procedures consist of two processes: a learning process and a serving process. In the learning process, PC clients send the application packets with a corresponding output label to the cloud computing via the edge computing. In this case, five types of applications (e.g., BitTorrent, web service, Skype, secure shell (SSH), and remote desktop protocol (RDP)) are considered as classification candidates. Here, it should be noted that the proposed scheme is not limited to classifying those five applications and can be easily extended to classify different types. In the cloud computing, first the collected packets are preprocessed, and through the learning process based on the output labels corresponding to the collected packets, an appropriate deep-learning model is created. Afterwards, the result of the deep-learning model is sent to the edge computing. The edge computing loads the received deep-learning model to act as an application classifier. In the serving process, the PC clients send the application packets without the corresponding output labels. Then, the edge computing examines or sniffs the application packets and classifies the application, which would be mapped into one of the five candidates. On the basis of this automated classification function, it is expected that a service provider can effectively perform the desired network management according to various application traffic.



(a)



(b)

Fig. 3. Proposed deep RNN-based traffic classification scheme for the (a) data preprocessing and (b) deep learning process (Stacked RNN/LSTM)

Table 1. Parameters explanations

Parameters	Explanations
$\boldsymbol{R}$	The raw data set for the first preprocessing procedure
$\boldsymbol{R}_{1st\_pre}^{type}$	The first preprocessing result with a certain type, which is a subset of $\boldsymbol{R}$
$\boldsymbol{N}^{flow}$	The data size of one sequential flow
$\boldsymbol{f}_{2nd\_pre,i}^{types}$	The second preprocessing data of the flow i
$\boldsymbol{f}_{i,0}^{types}$	The sequence 0 of flow i with a certain type

4. Proposed Deep RNN-Based Traffic Classification Scheme

In this section, the detailed process of the proposed scheme is explained as shown in Fig. 3. The entire process is divided into two parts: data preprocessing and deep learning. Detailed parameters are summarized in Table 1

4.1. Proposed Data Preprocessing

The set of features entering LSTM's input is payload data for packets in each flow. The input data of the neural network changed each element of the payload to 8 bits, and then re-imaged the bitted payload data and used it as input data through the preprocessing. In the proposed scheme, preprocessing is conducted in two stages, called the first preprocessing and second preprocessing procedures. For the first preprocessing procedure, raw data with a defined set $\boldsymbol{R}$ is collected and sniffed in the cloud computing. Then, raw data is classified into a corresponding type of application. Here, because only six types of applications are considered, including "others," each flow belongs to one of six types. For convenience, $\boldsymbol{R}_{1st\_pre}^{type}$ is denoted as a subset of $\boldsymbol{R}$, which is the first preprocessing result with a certain type, i.e., $\boldsymbol{type} \in \{RDP, Skype, SSH, BitTorrent, Web, Others\}$. In this process, a conversion process of the raw data is conducted, which 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 payload in the flow unit and creates a data set. The first preprocessing result is given by

$$\boldsymbol{R} = \boldsymbol{R}_{1st\_pre}^{RDP} \cup \boldsymbol{R}_{1st\_pre}^{Skype} \cup \boldsymbol{R}_{1st\_pre}^{SSH} \cup \boldsymbol{R}_{1st\_pre}^{Bitto} \cup \boldsymbol{R}_{1st\_pre}^{Web} \cup \boldsymbol{R}_{1st\_pre}^{Others} \quad (1)$$

For the second preprocessing procedure, square matrix images are generated from the classified raw data in the first preprocessing procedure. That is, each flow corresponding to a certain type contains sequential data with time-series (e.g., streaming data). In addition, one square matrix image in each sequential datum has a user-designed size, which can be set by the user to l. The bit size of one pixel in the square matrix image has a m, and the character-type value replaced by a floating-point value is $0 \sim (2^m - 1)$. That is, the data size of one sequential flow (N^{flow}) is obtained by

$$\mathrm{N^{flow}} = \quad \mathrm{k} \quad \times \quad \mathrm{l} \quad \times \quad \mathrm{m} \quad (2)$$

where k is the number of square matrix images per one flow, l is the number of pixels per one square matrix image, and m is the size of a pixel.

For instance, in the case of 30 sequences per flow, 256($= 16 \times 16$) pixels per sequence (square matrix image), and 4 bits per pixel for RNN models detecting time-varying target applications, N^{flow} has a value 30,720 ($= 30 \times 256 \times 4$).

Fig. 3(a) shows the payload of a packet in one of the completed flows in a two-dimensional image, an element of four bits in size, and a value between 0 and 15 (floating point). In addition, because of the nature of the LSTM network structure, all flows should have the same number of packets. Because the length of a sequence is set in advance, the number of packets per flow should be the same as the predefined length. However, because application flows can have a varying number of packets, the number of packets should be processed with a defined size. Specifically, if the number of packets per flow is smaller than N^{flow}, the packet is padded by zero. Contrarily, if the number of packets per flow is greater than N^{flow}, then the packets over N^{flow} are discarded. Finally, the second preprocessing data of the flow i ($\mathrm{f}^{\mathrm{types}}_{\mathrm{2nd\_pre,i}}$) is given by

$$\mathrm{f}^{\mathrm{types}}_{\mathrm{2nd\_pre,i}} = [f^{types}_{i,0} \quad f^{types}_{i,1} \quad \cdots \quad f^{types}_{i,29}], \tag{3}$$

where $f^{types}_{i,0}$ is denoted as a sequence 0 of flow i with a certain type.

4.2. Proposed Deep-Learning Process

In the deep-learning process, these generated input features with corresponding output labels are inserted and trained to create an acceptable inferred function (i.e., deep learning model) where a stacked RNN model with three layers was utilized in this study to improve accuracy with consideration of time-varying target applications. In particular, for the existing vanilla RNN model, the length of the sequences should be short; otherwise, acceptable accuracy cannot be achieved, which is called the "long-term dependency problem." To alleviate this issue while using 30 sequences per flow, RNN with the LSTM method is additionally considered, which is composed of a memory cell, an input gate, an output gate, and a forget gate. Then, each LSTM cell takes an input and stores it for some period of time to solve the "long-term dependency problem." The output labels used for the train, validation, and test tasks were designed as one-hot vectors as shown in Table 2, where the size of the one-hot vectors is $[5 \times 1]$ because of the five application types. LSTM [48, 49] is a type of the RNN model. It is useful for training datasets with long-term dependency, so that it is commonly used to train speech and text dataset. In LSTM, the previous learning data is reflected in the current learning data using the cyclic structure. As a result, LSTM is suitable for the classification of the flow-based network traffic dataset with sequential feature. As explained in Section II, collection of the application packets and learning for the deep learning model is conducted in the cloud computing, and the classification of an application packet is served by the edge computing. For learning, an equal amount of data for the six application types should be prepared, which is used as an input for generating the deep learning model. As depicted in Fig. 3(b), $f^{types}_{i,j}$, with $= 16 \times 16$ square matrix images, is used as an input feature, where j is a sequence index from 0 to 29. In addition, each input feature has a corresponding output label designed as in Table 2. Finally, in the serving process, classification is conducted on the basis of the result of output labels $[5 \times 1]$ such that the largest vector can be selected as an inferred application type. For instance, if the result represents a vector with [0.01 0.12 0.76 0.04 0.07], SSH with [0 0 1 0 0] is selected.

Table 2. Output labels of applications (one-hot vector)

Application	BitTorrent	Web	Skype	SSH	RDP
Output label	$\begin{bmatrix}1\\0\\0\\0\\0\end{bmatrix}$	$\begin{bmatrix}0\\1\\0\\0\\0\end{bmatrix}$	$\begin{bmatrix}0\\0\\1\\0\\0\end{bmatrix}$	$\begin{bmatrix}0\\0\\0\\1\\0\end{bmatrix}$	$\begin{bmatrix}0\\0\\0\\0\\1\end{bmatrix}$

Table 3. Deep-learning system environment

	Case	**Description**
Cloud Computing	DL Toolkit	Tensorflow 1.12
	Language	Python 3.6
	OS	Ubuntu 16.04 LTS
	RAM	32 GB
	GPU	Two NVIDIA GTX 1080Ti, 11GB
	CPU	Intel Core i9-7900X @ 3.30GHz
	Hyper parameter	Optimizer: Adam, Batch: 100, Epoch: 200
Edge Computing	DL Toolkit	Tensorflow 1.12
	Language	Python 3.6
	OS	Ubuntu 16.04 LTS
	RAM	32 GB
	GPU	One NVIDIA GTX 1080Ti, 11GB
	CPU	AMD Ryzen 5 1400 Quad-Core Processor
	Hyper parameter	Optimizer: Adam, Batch: 100, Epoch: 200

5. Performance Evaluation

In this section, the details regarding the performance evaluation of the proposed scheme is discussed. The proposed deep RNN-TCS is compared with the conventional approach, a deep CNN-traffic classification scheme (deep CNN-TCS) [43].

The experiments were conducted on Ubuntu 16.04 LTS, using 32 GB of RAM and two NVIDIA GTX 1080Ti GPUs with 11 GB for the cloud computing and using 16 GB of RAM and one NVIDIA GTX 1080Ti GPU with 11 GB for the edge computing. Tensorflow 1.12 in the Python 3.6 environment is used to configure the LSTM and CNN deep-learning models. For the experiments, 2000 flows for each application are used in the training data, and the number of packets per flow was set to 10, 30, 60, and 100. In addition, the payload size of each packet was set to 40, 80, and 160. Table 3 shows the detailed hyper-parameters used in deep RNN-TCS and deep CNN-TCS. There is a limitation of the DL-PAS in applications to a high-user-density nature owing to the factorial increase in both output nodes and multi-layer perceptions (MLP) weight. The LSTM model used in RNN-TCS consists of a single LSTM layer, and an output layer. The number of cells in the LSTM layer is 320, and we use the 'uniform' distribution for the model parameter initializer. The dropout rate is set to 0.2 in order to prevent the overfitting. The activation function is 'softmax', optimization type is 'adam', and batch size is 100.

The traffic data used to classify applications were preprocessing packet capture (PCAP) files supplied by Universitat Politecnica de Catalunya Barcelonatech (UPC) [4] suitable

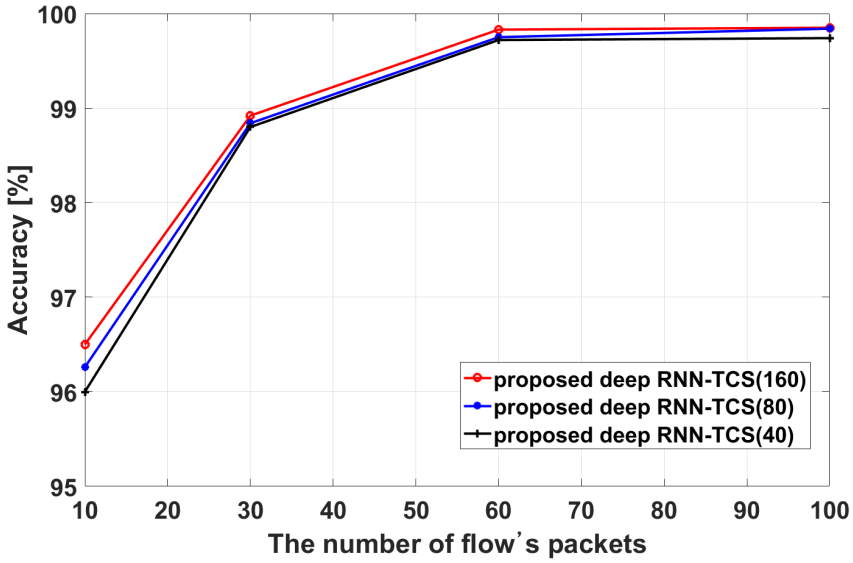


Fig. 4. Accuracy of the deep RNN-based traffic classification scheme

for RNN learning. For the data preprocessing process, six types of applications are considered, where web applications included HTTP-Facebook-Google, HTTP-Web, HTTP-Wiki, and HTTP-Youtube. The learning data set included flight data with 8,750 flows, 1,250 validation data, and 3,000 test data. For five applications, the overall accuracy of the deep RNN-TCS with respect to the number of packets per flow are examined. In this investigation, the deep RNN-TCS is tested with different payload sizes, i.e., 40, 80, or 160 bytes, denoted by proposed deep RNN-TCS(40), deep RNN-TCS(80), and deep RNN-TCS(160), respectively. As shown in Fig. 4, as the number of packets per flow increases, the accuracy increases. Similarly, the accuracy also increases as the payload size increases, because a lot of packets and big payload size provides more information to the classifier to make an accurate decision. Specifically, the proposed scheme achieves accuracy of 96.00% - 99.85% with varying conditions.

Fig. 5 shows the accuracy of the proposed scheme for each five applications. As shown in the figure, the accuracy of each application increases as the number of packets per flow increases. In particular, SSH, RDP and Skype usually consist of text or control data with fewer bits, thus requiring fewer packets per flow. Correspondingly, as the number of packets per flow and payload size increase in the all applications, the accuracy of RNN-TCS also reaches about 99%. However, if the dataset size is small (i.e., 10 packets per flow with 40 payload size), especially in case of Web and BitTorrent, the accuracy is not as high (88.97% and 95.65%, respectively). Because BitTorrent and Web are usually composed of image or video data with a large amount of data, our scheme requires more packets per flow to classify these applications accurately.

A clearly presenting the predictions of the deep RNN-TCS model is to use a confusion matrix. Table 4 shows the confusion matrix generated by the test process of the proposed deep RNN-TCS as a flow-based dataset. For example, the deep RNN-TCS ac-

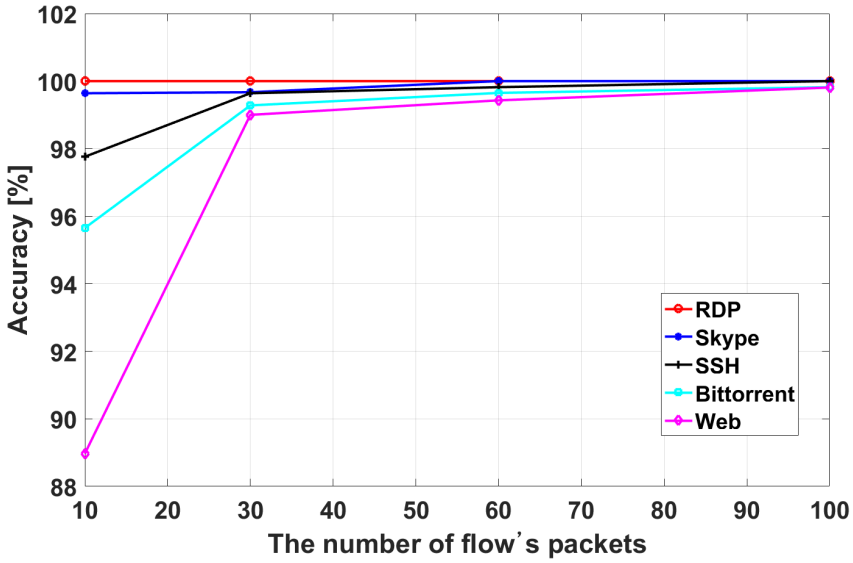


Fig. 5. Accuracy of deep RNN-TCS for each target application

curately predicts 574 BitTorrents, 593 Web, 578 Skype, 594 SSH and 603 RDP. It also incorrectly predicted 58 cases (all cases except the diagonal position in Table 4) from the total number of all estimates. In addition, the F1 score of each application label remains high with strong robustness, demonstrating that the deep RNN-TCS model can effectively and reliably predict network applications. If the number of application labels is different, the accuracy indicator may be misrepresented. The model predicts most applications for all predictions and achieves high classification accuracy, but the model may not be useful for problem areas. Therefore, we replace the confusion matrix, which is the prediction result for all application labels, with the binary confusion matrix for each label. Table 5 is an example of a binary confusion matrix of BitTorrent labels created based on Table 4. In the binary confusion matrix in Table 5, the deep RNN-TCS predicts 600 (= 574 + 26) of the total 3000 test datasets as BitTorrent and 2400 (= 19 + 2381) as the remainder. In fact, 593 (= 574 + 19) of the test dataset is bit torrent and 2407 (= 26 + 2381) is the remainder. The TP represents the cases in which the actual label is positive (BitTorrent) and the prediction result is also positive correctly. The FN represents the cases in which the actual label is positive, but the prediction result is negative incorrectly. The FP represents the cases in which the actual label is negative (that is, not BitTorrent), but prediction result is positive incorrectly. Lastly, the TN represents the cases in which the actual label is negative, and the prediction result is also negative correctly.

To solve the reliability issue of accuracy, the deep RNN-TCS is evaluated by calculating the F1-score in this paper. Using a binary confusion matrix, it is defined according to the following equation:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

Table 4. The all applications confusion matrix by the proposed deep RNN-TCS test with 100 packets per flow and 160 pixels.

Applications		Predicted						
		BitTorrent	Web	Skype	SSH	RDP	SUM	F1 score
Actual	BitTorrent	574	0	2	17	0	593	0.99
	Web	6	593	1	1	0	601	1.00
	Skype	19	1	578	1	0	599	1.00
	SSH	1	3	0	594	5	603	0.99
	RDP	0	0	0	1	603	604	1.00
Total F1-score								0.99

Table 5. The Binary confusion matrix of BitTorrent application labels in deep RNN-TCS tests.

n=3000		Prediction	
		Positive	**Negative**
Actual	Positive	574 (True Positive: TP)	19 (False Negative: FN)
	Negative	26 (False Positive: FP)	2381 (True Negative: TN)

$$\mathrm{F1-score} = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{5}$$

where Recall = TP/(TP + FP) and Precision = TP/(TP + FN).

The F1-score expresses the harmonic mean of precision and recall, and shows the predicted results performance of a deep learning model accurately. In this paper, we compute the accuracy, recall, precision, and F1-score values of all five application labels.

Finally, Fig. 6, Fig. 7 and Fig. 8 represent that the performance of the accuracy and elapsed time for the different schemes (the proposed deep RNN-TCS and the traditional deep CNN-TCS). The CNN model used in CNN-TCS consists of two convolutional layers, a maxpooling layer, and finally an output layer. Like RNN-TCS, we use the 'uniform' distribution for the model parameter initializer. The number of CNN model filters used in the experiment is the same as the payload size of the dataset used for training. Also, the size of the kernel is $\mathbf{3} \times \mathbf{3}$ and the output size of the convolutional layers maintains as the input size by using the 'same' padding method. The activation function is 'softmax', the optimization type is 'adam', and the batch size is 100. It should be noted that the elapsed time is the sum of the preprocessing time for generating the appropriate data for each model and the time taken to train the CNN and LSTM model. As shown in Fig. 6 and Fig. 7, as the number of applications increases, the proposed scheme, which utilizes streaming training data, achieves almost 96%-99% accuracy and 0.995-0.998 F1-score, whereas the conventional deep CNN-TCS is 91%-96% accurate and 0.926-0.948 F1-score, for a nearly 5% accuracy gap and 0.06 F1-score gap. The elements of the application types according to the number of applications is summarized in Table 6. Furthermore, with regard to elapsed time, as depicted in Fig. 8, the proposed scheme is almost 500 times faster than the conventional scheme, because the conventional scheme reads arbitrary packets of each application flow in the data preprocessing step. On the other hand, the proposed scheme

Table 6. Elements of application types with the number of applications

Number of apps.	Set of apps.
2	{BitTorrent, Web}
3	{BitTorrent, Web, Skype}
4	{BitTorrent, Web, Skype, SSH}
5	{BitTorrent, Web, Skype, SSH, RDP}

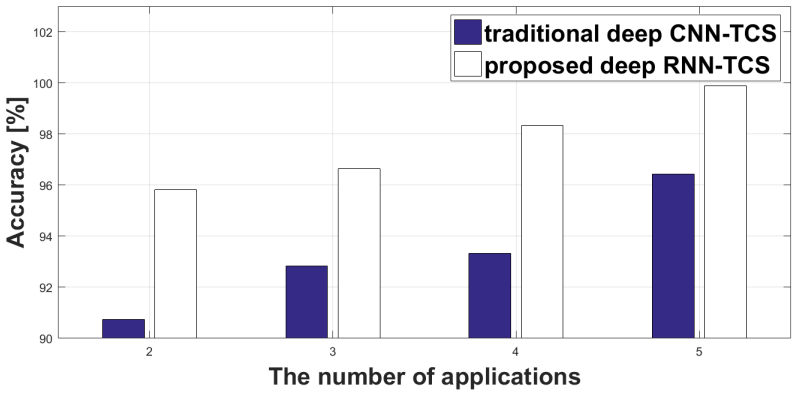


Fig. 6. Performance evaluations for accuracy of the proposed deep RNN-TCS vs. the traditional deep CNN-TCS

reads and generates data in each application flow unit. In particular, the elapsed time of RNN-TCS is faster than that of CNN-TCS because CNN-TCS in the pre-processing process converts the payload into an image after reading all data sets of network traffic. On the other hand, in the case of RNN-TCS, the network traffic dataset is extracted as the number of packets per flow, and the payload is converted to an image only for the extracted dataset. Here, the overall tendency of both schemes is that as the number of applications increases, because of the increased training data for classification, the overall accuracy is improved at the cost of more elapsed time.

In addition, recently, network traffic classification schemes using various machine learning methods have been actively studied. Parsaei et al. [31] described a method of classifying network traffic using the four neural network models (Feedforward Neural Network, Multi-layer Perceptron, Levenberg-Marquardt, and Naive Bayes) in Software-Defined Networking. The four neural network models show accuracy of 95.6%, 97%, 97% and 97.6%, respectively. Parsaei's scheme performed the classification based on the header information as well as the payload one. But, our RNN-TCS performs it based only on the packet payload. It achieves a 99% accuracy, so that it is better than the above four neural network models.

Wang et al. [42] worked on classifying malware traffic and normal traffic using a CNN model. They preprocessed all of the header and payload information of the network traffic into images for input data from the CNN model. The CNN model is trained from the imaging dataset, and the accuracy of malware traffic and normal traffic classification was almost 100%. But, the binary classification of the Wang's scheme is the simplest kind

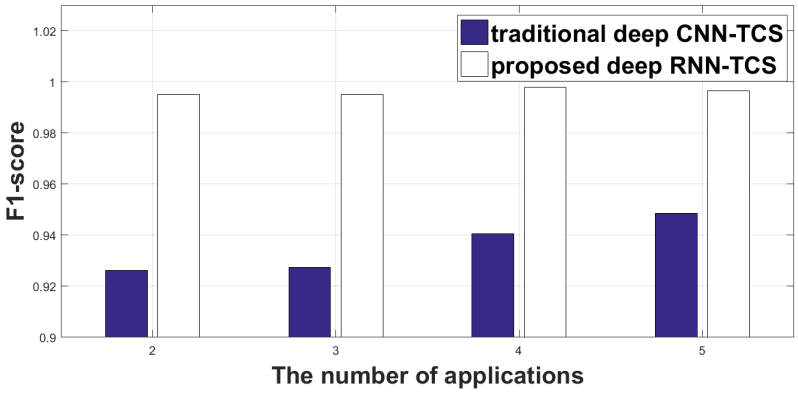


Fig. 7. Performance evaluations for F1-score of the proposed deep RNN-TCS vs. the traditional deep CNN-TCS

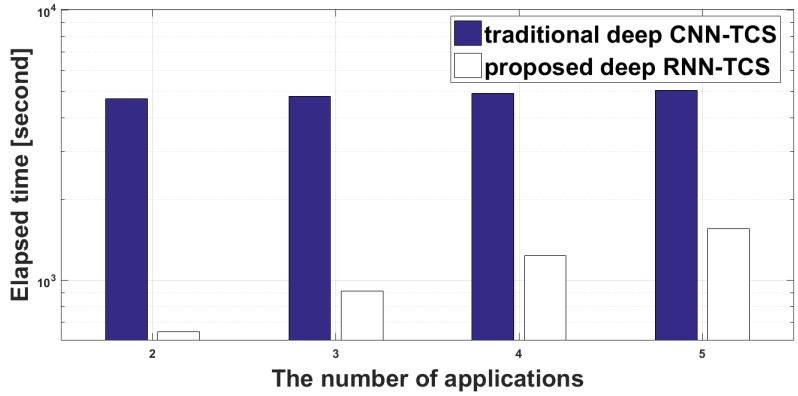


Fig. 8. Performance evaluations for preprocessing elapsed time of the proposed deep RNN-TCS vs. the traditional deep CNN-TCS

of machine learning problem. On the other hand, our proposed RNN-TCS is not binary classification, but multiple classification.

In our previous work [23], we collected payload-based network traffic which excluded the TCP/UDP headers, and used CNN and ResNet models for network traffic classification method. According to the network traffic classification results, the F1-score values for CNN and ResNet models are 0.948 and 0.969, respectively. The proposed scheme in this paper is also payload-based network traffic classification, but it uses the RNN model unlike previous work. We experimented with a comparison of our RNN-based scheme with the previous one. As a result, our RNN-based scheme outperforms the previous one.

6. Discussion

When new traffic data are generated from a new application (or protocol), the proposed traffic classification scheme does not classify them correctly, since the new data were not used to train the model. Even with traditional methods including DPI, however, it is impossible to know new applications in advance and the traffic from the new application will be unclassified with the methods. Rather, the proposed method can better respond to traffic generated by a new application, since a manager only creates a new label for the new application data, and retrain the model with the new training data augmented with the new traffic images and label. The DPI-based traditional methods are more difficult because it must come up with new rules (e.g., a specific values or structure on headers or payload) for the new application data.

From a system perspective, the proposed scheme includes the ability to collect data about the new application with a cloud computing system in a centralized manner, update the model through retraining with the new application data, and apply the results to edge computing systems in a distributed manner.

Regarding the classification of encrypted packets, much works have been already studied [34]. In case the packets are encrypted, as we can see from the previous studies, it should be possible measures 1) to add metadata to encrypted packets in the preprocessing phase, and utilize them in the classification phase, 2) to utilize the various data interpolation (Nearest value based, Bi-linear Interpolation based, Bicubic based) schemes to process data packets with various lengths to a fixed size, or 3) to extract the features through the packet header and encrypted payload using deep packet toolkit with deep learning. Therefore, by using such extended measures, classification of encrypted packets can be performed with the deep learning-based scheme proposed in this paper.

7. Conclusion

In this paper, a RNN-based traffic classification scheme (RNN-TCS) is proposed to classify applications from traffic patterns. RNN-TCS is a payload-based network classification method, so that it can classify network traffic well even though dynamic IP addresses or port numbers are used on the packet header. In addition, the architecture of the proposed technique is a hybrid edge computing system capable of parallel execution by performing data preprocessing and model training in the cloud and classifying network traffic with the trained model in edge computing. For evaluation, traffic packets measured at Universitat Politecnica de Catalunya Barcelonatech for our training data are utilized and the proposed scheme is implemented using a deep LSTM system. Finally, it is shown that the proposed deep RNN-TCS achieves almost 96%-99% accuracy and 0.995-0.998 F1-score with low elapsed time. In the future, we plan to classify more than 100 services based on deep RNN-TCS in edge computing studied in this paper. Also, based on each classified service, we will conduct a research to control Quality of Service using Software-Defined Networking.

Acknowledgments. This research was supported by the National Research Council of Science & Technology (NST) grant by the Korea government (MSIP) (No. CRC-15-05-ETRI).

References

1. Aceto, G., Ciuonzo, D., Montieri, A., Pescapé, A.: Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges. IEEE Transactions on Network and Service Management 16(2), 445–458 (2019)
2. Bernaille, L., Teixeira, R., Salamatian, K.: Early application identification. in Proceedings of the 2006 ACM CoNEXT Conference pp. 1–12 (2006)
3. Boutaba, R., Salahuddin, M.A., Limam, N., Ayoubi, S., Shahriar, N., Estrad-solano, F., Caicedo, O.M.: A comprehensive survey on machine learning for networking: evolution, applications and research opportunities. Journal of Internet Services and Applications (2018)
4. Carela-Español, V., Bujlow, T., Barlet-Ros, P.: Is Our Ground-Truth for Traffic Classification Reliable? In Proc. of the Passive and Active Measurements Conference (PAM'14) (Mar 2014)
5. Connor, J.T., Martin, R.D., Atlas, L.E.: Recurrent neural networks and robust time series prediction. IEEE Transactions on Neural Networks pp. 240–254 (1994)
6. Deng, W., Xui, J., Zhao, H.: An Improved Ant Colony Optimization Algorithm Based on Hybrid Strategies for Scheduling Problem. IEEE Access pp. 20281–20292 (2019)
7. Deng, W., Zhao, H., Yang, X., Xiong, J., Sun, M., Li, B.: Study on an improved adaptive PSO algorithm for solvingmulti-objective gate assignment. Applied Soft Computing pp. 288–302 (2017)
8. Deng, W., Zhao, H., Zou, L., Li, G., Yang, X., Wu, D.: A novel collaborative optimization algorithm in solving complex optimization problems. Soft Computing pp. 4387–4398 (2017)
9. Erman, J., Mahanti, A., Arlitt, M., Williamson, C.: Identifying and discriminating between web and peer-to-peer traffic in the network core. in Proceedings of the 16th International Conference on World Wide Web pp. 883–892 (2007)
10. Finsterbusch, M., Richter, C., Rocha, E., Muller, J.A., Hanssgen, K.: A survey of payload-based traffic classification approaches. IEEE Communications Surveys & Tutorials pp. 1135–1156 (2014)
11. Gupta, P., McKeown, N.: Algorithms for packet classification. IEEE Network: The Magazine of Global Internetworking pp. 24–32 (2001)
12. Haffner, P., Sen, S., Spatscheck, O., Wang, D.: automated construction of application signatures. in MineNet (2005)
13. Hatcher, W.G., Yu, W.: A survey of deep learning: platforms, applications and emerging research trends. IEEE Access. pp. 24411–24432 (2018)
14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. IEEE Conference on Computer Vision and Pattern Recognition 2016 pp. 770–778 (2016)
15. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation pp. 1735–1780 (1997)
16. Kohavi, R.: A study of cross-validation and bootstrap for accuracy estimation and model selection. in Proceedings of the 14th International Joint Conference on Artificial Intelligence pp. 1137–1143 (1995)
17. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems 25 (NIPS 2012) pp. 1097–1105 (2012)
18. Lan, K., Heidemann, J.: On the correlation of Internet flow characteristics. Technical Report ISI-TR-574, USC/Information Sciences Institute (2003)
19. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. in Proceedings of the IEEE vol. 86(11), 2278–2324 (Nov 1998)
20. LeCun, Y., Bengio, Y., Hinton, G.: Deep Learning. Nature International Journal of science pp. 436–444 (May 2015)
21. Li, F., Kakhki, A.M., Choffnes, D., Gill, P., Mislove, A.: Classifiers unclassified: An efficient approach to revealing IP traffic classification rules. in proceedings of the 2016 Internet Measurement Conference pp. 239–245 (2016)

22. Li, L., Kianmehr, K.: Internet traffic classification based on associative classifiers. IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER) pp. 263–268 (2012)
23. Lim, H., Kim, J., Heo, J., Kim, K., Hong, Y., Han, Y.: Packet-based Network Traffic Classification Using Deep Learning. In Proceedings of the 2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC) pp. 46–51 (2019)
24. Liu, Y., Wang, X., Zhai, Z., Chen, R., Zhang, B., Jiang, Y.: Timely daily activity recognition from headmost sensor events. ISA Transactions pp. 379–390 (2019)
25. Lopez-Martin, M., Carro, B., Sanchez-Esguevillas, A., Lloret, J.: Network traffic classifier with convolutional and recurrent neural networks for internet of things. IEEE Access. pp. 18042–18050 (2017)
26. McGregor, A., Hall, M., Lorier, P., Brunskill, J.: Flow clustering using machine learning techniques. in Passive and Active Network Measurement. Berlin, Heidelberg: Springer Berlin Heidelberg pp. 205–214 (2004)
27. Meidan, Y., Bohadana, M., Shabtai, A., Guarnizo, J.D., Ochoa, M., Tippenhauer, N.O., Elovici, Y.P.: A machine learning approach for IoT device identification based on network traffic analysis. in proceedings of the Symposium on Applied Computing pp. 506–509 (2017)
28. Mikolov, T., Kombrink, S., Burget, L., Cernocky, J., Khudanpur, S.: Extensions of recurrent neural network language model. 2011 IEEE International Conference on Acoustics, Speech and Signal Processing(ICASSP) pp. 5528–5531 (2011)
29. Nguyen, T.T.T., Armitage, G.: A survey of techniques for internet traffic classification using machine learning. IEEE Communications Surveys Tutorials pp. 56–76 (2008)
30. Park, J.: Statistics signiture based application traffic classification. Korea Communication Journal 34, 1234–1244 (Nov 2009)
31. Parsaei, M.R., Sobouti, M.J., Khayami, S.R., Javidan, R.: Network traffic classification using machine learning techniques over software defined networks. International Journal of Advanced Computer Science and Applications (2017)
32. Pedregosa, F., Varoquaux, G., Gramfort, A.: Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research pp. 2825–2830 (2011)
33. Powers, D.M.W.: Evaluation: From precision, recall and f-measure to ROC, informedness, markedness & correlation. Journal of Machine Learning Technologies pp. 37–63 (2011)
34. Rezaeij, S., Liu, X.: Deep learning for encrypted traffic classification: An overview. IEEE Communications Magazine pp. 76–81 (2019)
35. Risso, F., Baldi, M., Morandi, O., Baldini, A., Monclus, P.L.: Lightweight, payload-based traffic classification: An experimental evaluation. IEEE International Conference on Communications pp. 5869–5875 (2008)
36. Sak, H., Senior, A., Beaufays, F.: 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)
37. Shafiq, M., Yu, X., Wang, D.: Network traffic classification using machine learning algorithms. Advances in Intelligent Systems and Computing pp. 621–627 (2018)
38. Singh, H.: Performance analysis of unsupervised machine learning techniques for network traffic classification. 2015 Fifth International Conference on Advanced Computing & Communication Technologies pp. 401–404 (2015)
39. Trivedi, U., Patel, M.: A fully automated deep packet inspection verification system with machine learning. 2016 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS) (Nov 2016)
40. Udrea, O., Lumezanu, C., Foster, J.S.: Rule-based static analysis of network protocol implementations. Information and Computation p. 130–157 (2008)
41. Wang, W., Sheng, Y., Wang, J., Zeng, X., Ye, X., Huang, Y., Zhu, M.: Hast-ids: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection. IEEE Access 6, 1792–1806 (2018)

42. Wang, W., Zhu, M., Zeng, X., Ye, X., Sheng, Y.: Malware trac classification using convolutional neural network for representation learning. In Proceedings of the 2017 International Conference on Information Networking (ICOIN) pp. 712–717 (2017)
43. Wang, W., Zhu, M., Zeng, X., Ye, X., Sheng, Y.: Malware Traffic Classification Using Convolutional Neural Network for Representation Learning. IEEE ICOIN 2017 (2017)
44. Yu, C., Lan, J., Xie, J., Hu, Y.: QoS-aware traffic classification architecture using machine learning and deep packet inspection in SDNs. Procedia Computer Science pp. 1209–1216 (2018)
45. Zander, S., Nguyen, T., Armitage, G.: Automated traffic classification and application identification using machine learning. The IEEE Conference on Local Computer Networks 30th Anniversary (LCN'05) pp. 250–257 (2005)
46. Zhang, J., Xiang, Y., Wang, Y., Zhou, W., Xiang, Y., Guan, Y.: Network traffic classification using correlation information. IEEE Transactions on Parallel and Distributed Systems pp. 104–117 (2013)
47. Zhang, Y., Breslau, L., Paxson, V., Shenker, S.: On the characteristics and origins of internet flow rates. SIGCOMM 02 Proceedings of the 2002 conference on Applications pp. 309–322 (2002)
48. Zhao, H., Liu, H., Xu, J., Deng, W.: Performance Prediction Using High-Order Differential Mathematical Morphology Gradient Spectrum Entropy and Extreme Learning Machine. IEEE Transactions on Instrumentation and Measurement pp. 379–390 (2019)
49. Zhao, H., Zheng, J., Xu, J., Deng, W.: Fault Diagnosis Method Based on Principal Component Analysis and Broad Learning System. IEEE Access pp. 99263–99272 (2019)

Kwihoon Kim is currently a professor in the Department of Artificial Intelligence Convergence Education, Korea National University of Education (KNUE), South Korea. He received the B.S, M.S. and Ph.D. degrees from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea in 1998, 2000 and 2019, respectively. He worked in LG DACOM 2000 2005. From 2005 to 2020, he was a Principle Researcher with Electronics and Telecommunications Research Institute (ETRI). He is an editor and rapporteur of ITU-T SG11 since 2006. His interested fields are Fog/edge computing, Internet of Things, 5G/IMT2020, deep learning, machine learning, reinforcement learning, GAN and knowledge-converged intelligent service.

Joohyung Lee (S'09–M'14–SM'19) is currently an Assistant Professor in the School of Computing, Gachon University, South Korea. He received the B.S, M.S. and Ph.D. degrees from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2008, 2010 and 2014, respectively. From 2012 to 2013, he was a Visiting Researcher with the Information Engineering Group, Department of Electronic Engineering, City University of Hong Kong, Hong Kong. From 2014 to 2017, he was a Senior Engineer with Samsung Electronics. He has contributed several articles to the International Telecommunication Union Telecommunication (ITU-T) and the 3rd Generation Partnership Project (3GPP). His research work is resource management at the intersection of mobile systems and machine learning focusing on edge computing architectures to optimize the trade-off between latency, energy, bandwidth and accuracy for data analytics.

Hyun-Kyo Lim received the B.S. degree in computer science and engineering and the M.S. degree in computer science engineering from the Korea University of Technology

and Education, in 2015 and 2017, respectively, where he is currently pursuing the Ph.D. degree with the Department of Interdisciplinary Program in Creative Engineering. He studied mobility management during his master course and he especially researched distributed mobility management in software-defined networking. He is studying deep learning and reinforcement learning during his doctoral studies. He is also exploring ways to apply deep learning and reinforcement learning to the network and is working on applying deep learning and reinforcement learning to a variety of applications.

Se Won Oh received the B.S.(1999) and the M.S. degrees(2001) from Pohang University of Science and Technology (POSTECH), and received the Ph.D.(2018) from Chungnam National University (CNU), Daejeon, South Korea, respectively. Since joining Electronics and Telecommunications Research Institute (ETRI) in 2001, he is currently a Principal Researcher working for Knowledge-converged Super Brain (KSB) Convergence Research Department in ETRI, Daejeon, South Korea. He has been involved in several research projects on software platform (such as RFID Event Management, USN Middleware, Internet of Things Platform) which integrates legacy applications with various data resources and sensors. He has made several contributions in international standardization activities, particularly on automatic identification and data capture techniques of JTC 1/SC 31. His recent interested areas include machine learning application, anomaly detection, and knowledge-converged intelligent service solutions.

Youn-Hee Han received the B.S. degree in mathematics and the M.S. and Ph.D. degrees in computer science and engineering from Korea University, Seoul, South Korea, in 1996, 1998, and 2002, respectively. From 2002 to 2006, he was a Senior Researcher with the Next Generation Network Group, Sam- sung Advanced Institute of Technology. Since 2006, he has been a Professor with the School of Computer Science and Engineering, Korea University of Technology and Education, Cheonan, South Korea. Since 2002, his activities have been focusing on mobility management, media independent handover, and cross-layer optimization for efficient mobility support. He has published approximately 250 research articles on the theory and application of mobile computing and has filed 40 patents on information and communication technology domain. His current research interests include theory and application of computer networks, including protocol design and math- ematical analysis, mobile sensor/actuator networks, social network analysis, machine learning, deep learning, and reinforcement learning. He has made several contributions in IETF and IEEE standardization. He has served as the Co-Chair for working group in the Korea TTA IPv6 Project Group. He has been serving as an Editor for the Journal of Information Processing Systems since 2011.

Received: April 24, 2020; Accepted: July 25, 2021.