IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 11, 2020, accepted December 27, 2020, date of publication December 31, 2020,
date of current version January 8, 2021.

Digital Object Identifier 10.1109/ACCESS.2020.3048348

Explaining Deep Learning-Based Traffic Classification Using a Genetic Algorithm

SEYOUNG AHN[1], JEEHYEONG KIM[2], SOO YOUNG PARK[3], AND SUNGHYUN CHO[1], (Member, IEEE)

[1]Department of Computer Science and Engineering, Major in Bio Artificial Intelligence, Hanyang University, Ansan 15588, South Korea
[2]Information and Intelligent Security Laboratory, Kennesaw State University, Marietta, GA 30060, USA
[3]Department of Internal Medicine, School of Medicine, Kyungpook National University, Kyungpook National University Hospital, Daegu 41944, South Korea

Corresponding author: Sunghyun Cho (chopro@hanyang.ac.kr)

This work was supported in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education under Grant NRF-2018R1D1A1B07049043, and in part by the Institute for Information & Communications Technology Promotion (IITP) Grant funded by the Korean Government (MSIT) (Full duplex non-orthogonal multiple access (NOMA) optimization technologies using deep learning for 5G based autonomous vehicular networks) under Grant 2018-0-00969.

ABSTRACT Traffic classification is widely used in various network functions such as software-defined networking and network intrusion detection systems. Many traffic classification methods have been proposed for classifying encrypted traffic by utilizing a deep learning model without inspecting the packet payload. However, they have an important challenge in that the mechanism of deep learning is inexplicable. A malfunction of the deep learning model may occur if the training dataset includes malicious or erroneous data. Explainable artificial intelligence (XAI) can give some insight for improving the deep learning model by explaining the cause of the malfunction. In this paper, we propose a method for explaining the working mechanism of deep-learning-based traffic classification as a method of XAI based on a genetic algorithm. We describe the mechanism of the deep-learning-based traffic classifier by quantifying the importance of each feature. In addition, we leverage the genetic algorithm to generate a feature selection mask that selects important features in the entire feature set. To demonstrate the proposed explanation method, we implemented a deep-learning-based traffic classifier with an accuracy of approximately 97.24%. In addition, we present the importance of each feature derived from the proposed explanation method by defining the dominance rate.

INDEX TERMS Traffic classification, deep learning, explainable artificial intelligence (XAI), genetic algorithm.

I. INTRODUCTION

With the proliferation of Internet-connected devices and various Internet services, it is important to control the tremendous traffic volume in an efficient manner. Traffic classification can be used to control various types of traffic in software-defined networking (SDN) or to detect malicious traffic in network intrusion detection system (NIDS) [1]. In the case of SDN, QoS management is important to mitigate the burden of the entire network and to fulfill the requirements of each type of service [2]. As the Internet services are more diverse, It is important to give each Internet service the differential QoS. Dynamic QoS can provide the differential QoS by subdividing the QoS class to support a more elaborate QoS. In addition, because numerous devices are connected to the Internet, the importance of technologies for detecting and defending against various attacks that may occur on the network has been emphasized. NIDS works as a core function in network security by detecting attacks such as the denial of service (DoS) attack based on traffic classification.

Traditional traffic classifications (TCs) are usually based on a payload-inspection, which is called a payload-based TC. A payload-based TC directly inspects the payload of packets and matches the pre-defined patterns. Although a payload-based TC shows a high performance, there are two critical problems. One problem is that payload-based TC cannot inspect the encrypted payload. Because secure communication schemes such as SSH and TLS encrypt the payload, the payload-based approaches cannot inspect the payload scrambled by the encryption scheme. Another problem is

The associate editor coordinating the review of this manuscript and approving it for publication was Xiaochun Cheng.

that inspecting the payload of packets requires enormous computational resources.

A flow-behavior-based TC has been proposed to address critical problems of a traditional TC. A flow-behavior-based TC is based on machine-learning technologies that can recognize patterns without inspecting the payload. In a flow-behavior-based TC, the machine-learning model learns various statistical features appearing in the network, such as the inter-arrival time or packet size. The statistical features show differences in each application because the network applications use different protocols and the patterns of behavior vary for each application user. Consequently, a flow-behavior-based TC has advantages that can operate within the encrypted traffic and satisfy the requirements of mission-critical applications requiring low latency.

However, there is a severe challenge to a flow-behavior-based TC that occurs based on the nature of machine-learning. The black-box problem is derived from the difficulty of explaining the results of the machine-learning model. With a lack of reliability of the results, the black-box problem has become a crucial issue in machine-learning [3]. As a crucial vulnerability owing to the black-box problem, a flow-behavior-based TC can be compromised by a black-box adversarial attack [4]. In a black-box adversarial attack scenario, an attacker deceives the machine-learning model by injecting adversarial perturbations, which is a type of noise, in the input data. When the compromised data are given to the machine-learning model, a serious attack scenario can occur by misclassifying the data. For example, in a case of traffic classification, an attacker can hijack high-priority QoS from victim traffic, such as the QoS of an autonomous driving application, which requires low latency [5]. The adversarial application can acquire high-priority QoS, which is supposed to be guaranteed for mission-critical applications. Consequently, the resources for applications that require high-priority QoS can be exhausted and no further normal operations of mission-critical applications are available without proper QoS requirements.

Detecting abnormal data in the dataset can produce significant clues for network engineers to improve the traffic classification model based on machine-learning. Explainable artificial intelligence (XAI) is a technology that describes the way machine-learning models operate [6]. Traditional machine-learning models work by comparing distributions of training and test dataset by formulating metrics such as the distance or score. After sufficient training, these metrics build the classification criteria formed as hyper-planes that distinguish data. For example, the mechanism of traditional machine-learning models such as a decision tree and support vector machine can be explained by visualizing or formulating classification criteria [7]. By contrast, explaining the mechanism of deep learning is more difficult than that of a traditional machine-learning model. Because deep learning models are based on multi-layer perceptrons and are trained by simply updating the parameters defined to each neuron, it is difficult to define a certain score or distance used to compare hyper-planes and data. Therefore, explaining the mechanism of deep learning is more difficult than explaining that of traditional machine-learning. Eventually, flow-behavior-based approaches also face the black-box problem along with the introduction of deep learning in traffic classification.

We propose a dominant feature selection method to explain how the proposed deep learning-based traffic classifier operates. We define a fitting score as a feature importance quantification and create a feature selection mask that finds the optimal trade-off between the high classification accuracy and a reduction of the unnecessary features based on a genetic algorithm. A genetic algorithm is an evolutionary algorithm that can solve the various NP-hard problems such as the traveling salesman problem (TSP) or the design of very large scale integration (VLSI). Finally, we describe the deep-learning-based traffic classifier by defining a dominance rate indicating the extent to which each deep learning model refers to each feature. In conclusion, the proposed method has two technical contributions.

- We propose a dominant feature selection method using a genetic algorithm to explain how the deep-learning-based traffic classifier operates. In particular, the proposed method can determine which part of the entire feature the classifier focuses on by quantifying the importance of each feature.
- We implement the flow-behavior-based traffic classifier as the evaluation method that classifies the traffic and produces the accuracy to compute the fitting score. Although the proposed method also works well in any granularity of the types of traffic, we implement a service-specific traffic classification model to figure out the characteristics of internet services.

The rest of this paper is organized into four sections. Related works on traffic classification and XAI are introduced in Section II. The construction of deep-learning-based traffic classifier and dominant feature selection method are introduced in Section III. Experimental results and a performance evaluation are presented in Section IV. An analysis of how the traffic can be classified into each service is also described in Sub-section c of Section IV. Finally, we provide some concluding remarks in Section V.

II. RELATED WORKS

A. FLOW BEHAVIOR-BASED TRAFFIC CLASSIFICATION

The most crucial issue of recent studies in traffic classification is to classify the encrypted traffic. Directly inspecting the payload was a barrier to encrypted traffic classification. Behavior statistics became a clue for classifying encrypted traffic because the statistics can be extracted without inspecting a scrambled payload. Flow-behavior-based approaches enable encrypted traffic to be classified by leveraging the behavior statistics. The authors of [8] introduced three representative encryption mechanisms of traffic and extracted the statistics from the encrypted traffic. Moreover, they evaluated

the performance of several machine-learning algorithms such as a support vector machine, random forest, naive Bayes, logistic regression, and neural networks. They presented the practicality of flow-behavior-based approaches by evaluating various machine-learning technologies.

With the significant advances in deep learning, many studies on traffic classification have adopted its strengths. The core advantage of deep learning over traditional machine-learning technologies is to enable the classifier to automatically extract features from the raw data. The authors of [9] adopted a convolutional neural network (CNN) for traffic classification. Representation learning is a method used to automatically extract features from raw data and the CNN is a typical method of representation learning in deep learning. The convolution layer enables a CNN to extract the local features from the raw data. The authors integrate feature extraction and training by leveraging the advantages of the CNN. In [10], the authors evaluated the two different types of typical deep learning models, a CNN and a recurrent neural network (RNN). An RNN is designed to handle sequential data such as time-series data. Several types of statistics can present the time-related nature, and the authors deal with the time-related statistics using an RNN. The authors of [11] proposed a deep learning-based traffic classification scheme for mobile encrypted traffic. The authors suggested that traffic classification schemes using a manually extracted feature set for mobile traffic generated by a moving target are impractical. In addition, they address the limitations of traditional traffic classification schemes by leveraging the advantages of deep learning, which can automatically extract the feature set.

Although deep learning shows an incredible performance, introducing deep learning directly can result in a deteriorated performance. Revising and applying a novel model is necessary because of the nature of features shown by behavior statistics. In [12], the authors described an issue in which many studies on deep learning-based traffic classification have usually adopted all the features equally without considering the type of statistics. The authors reflect the multimodality of behavioral statistics using a multimodal deep learning model. Consideration of traffic generated by anonymity tools (ATs) was introduced in [13]. Because it becomes important to preserve the privacy of users on the Internet, several ATs have been developed, including Tor. Consequently, several malicious usages of ATs produce crucial issues. The authors proposed an AT-specific traffic classification by leveraging a hierarchical classification that enables an efficient fine-grained tuning. The authors of [14] proposed a traffic classification scheme using a hierarchical classification. Flow-behavior-based approaches have a disadvantage in that they cannot classify unknown traffic classes because of the nature of machine-learning. Moreover, increasing the granularity of the traffic class exacerbates the classification performance. The authors compose the sub-classifier hierarchically based on the granularity of the traffic class. In [15], the authors addressed a problem in which an unknown traffic class cannot be classified using meta-learning. Deep learning needs sufficient dataset for the fine-tuning when an unknown traffic class appears. However, it is difficult to collect a sufficient dataset of an unknown traffic class. Few-shot learning, namely meta-learning, enables deep learning to train the relationship of each data.

B. EXPLAINABLE ARTIFICIAL INTELLIGENCE (XAI)

Explainable artificial intelligence (XAI) techniques have been studied to demonstrate the mechanism of the machine-learning model. In [6], the authors proposed the concept of explainable artificial intelligence (XAI), and devised a novel explainable model that allows the machine learning model to derive such classification results based on the feature subset of the input data. In [16], the authors visualized the important features that are used for classifying certain input data and explained why the deep learning model can recognize such data. To explain this, the authors proposed a sensitivity analysis (SA) that explains the impact of the transition of each pixel. Moreover, the authors also proposed layer-wise relevance propagation (LRP), which explains the importance of each pixel. In [17], the authors proposed EXPLAIN-IT, a framework that explains how to cluster an unlabeled YouTube traffic dataset acquired from the network using an unsupervised learning technique. EXPLAIN-IT explains the clustering method using LIME, which selects the feature most relevant to a specific decision from the input data. Hence, the key feature selection can explain why the deep learning model classifies the data. In [18], the authors describe the relationship between the input and output by inserting artificial perturbations in certain features. The input-output relationship can provide some interpretation rules for black-box predictors such as deep learning. Neural image caption generation with a visual attention scheme is proposed in [19]. The authors extracted the key features in the image using convolutional feature extraction. The extracted features are used to train the RNN for image captioning. During this procedure, the attention mechanism implemented through a convolutional feature extraction can highlight an important part of the image.

TABLE 1. The comparison of existing studies on XAI.

Ref.	Objectives	Methods
[16]	Image classification	Importance of features
[17]	Traffic classification	Key feature selection in unsupervised learning
[18]	Image classification	Inserting artificial perturbation
[19]	Image caption generation	Attention mechanism

Table 1 describes the comparison of existing studies on XAI. Many studies on XAI aim to explain the machine learning model for image classification. However, the traffic classification problem has different characteristics from the image classification problem. In the image classification problem, all elements of the data have the same semantic such as RGB color value. The attention mechanism proposed in [19] selects a feature subset by detecting an object from data composed of pixels having the same meaning. In the

traffic classification problem, the dimension of the data is smaller than that of the image data. In addition, since each element of the data has different meanings, a feature selection method that can reflect all of these characteristics is required. We designed a dominant feature selection method that is suitable for the low-dimensional behavioral statistics based on a genetic algorithm.

III. THE PROPOSED METHOD

The overview of the proposed dominant feature selection method is illustrated in Figure 1. The proposed method consists of two parts: (1) the construction of a traffic classifier, and (2) dominant feature selection. The traffic classifier is designed by a residual network (ResNet), which is known as a state-of-the-art deep learning technique [20]. The traffic classification applies data pre-processing and training step. The data pre-processing step collects packets from traffic flows and extracts the statistical features of each flow. After the pre-processing step, in which the traffic dataset is created, the traffic classifier is trained using the dataset composed of statistical features.

After the classifier is trained, the proposed dominant feature selection method generates a feature selection mask based on a genetic algorithm. The dominant feature selection conducts a mask selection and an offspring mask generation. The mask selection evaluates the masks by counting the zero-elements and calculating the accuracy using a masked input dataset and a pre-trained classifier. After the evaluation, with the mask selection picks a few masks are chosen using a roulette wheel selection method for the mask creation of the next generation. With the roulette wheel selection method, the probability of selecting the masks with a higher fitting score is higher than the others. The offspring mask generation creates a mask pool using the selected masks and gives variety to the mask pool through a crossover and mutation. The mask pool generated by the offspring mask generation is inherited by the next generation. After the iteration of two steps, the feature selection masks for each service are made and the masks are used to pick the features necessary to classify each service from all statistical features. Finally, we analyze the mechanism of the traffic classifier by computing the importance of each feature using the feature selection masks.

A. THE CONSTRUCTION OF A TRAFFIC CLASSIFIER

The construction of a traffic classifier consists of three steps: packet gathering, data pre-processing, and classifier training. The packet gathering step collects packets and groups them by the traffic to construct the training dataset. Because most packets are encrypted, a packet itself cannot be used as a training dataset, although the grouped packet dataset that shares the same end-to-end network address such as the IP address or TCP port number is needed. The packets in a grouped dataset may serve the same application service because they have the same application source, and such packets form a network flow. The packets in a network flow have a similar behavior, which is represented by the statistical features such as the packet size and inter-arrival time. After gathering packets from a network flow, the data pre-processing step computes the statistical features from the group of gathered packets. Finally, the classifier training step trains the deep-learning-based classifier using the dataset. Algorithm 1 shows the procedure of the construction of a traffic classifier.

Algorithm 1 Procedure of Traffic Classification

Require: Training packet trace P collected with short time duration, the traffic flow F composed of packets p_i, the function $\Omega(F)$ returning the 5-tuple of the flow F.

Ensure: Pre-trained traffic classifier.

1: $D \leftarrow \varnothing$
2: $\Omega = \{\Omega(F_1),\ \Omega(F_2),\ \ldots,\ \Omega(F_N)\}$
3: Perform clustering packets in P by 5-tuple set Ω to form a bidirectional flow set $\{F_1, F_2, \ldots, F_N\}$
4: **for** $i = 1 \rightarrow N$ **do**
5: $\quad \mathbf{t} \leftarrow \bigcup_{j=1}^{n-1} \{\tau(p_{j+1}) - \tau(p_j)\}$
6: $\quad \mathbf{s} = \{s_k | s_k$ is packet size of packet $p_k,\ 1 \leq k \leq n\}$
7: $\quad$ Compute total bytes b in the flow
8: $\quad$ Compute feature vector by using traffic flow statistical features

$$\psi = [m_{\mathbf{t}}\ M_{\mathbf{t}}\ \mu_{\mathbf{t}}\ \sigma_{\mathbf{t}}\ m_{\mathbf{s}}\ M_{\mathbf{s}}\ \mu_{\mathbf{s}}\ \sigma_{\mathbf{s}}\ n\ b]$$

9: $\quad$ Compute reverse directional feature vector $\bar{\psi}$
10: $\quad \mathbf{x_i} = [\psi,\ \bar{\psi}]$
11: $\quad$ Detect the application layer l_i by the packet gathering step
12: $\quad D \leftarrow D \cup \{(\mathbf{x_i}, l_i)\}$
13: **end for**
14: Normalize dataset D
15: **for** $i = 0 \rightarrow N$ **do**
16: $\quad$ Pick $(\mathbf{x_i}, l_i) \in D$
17: $\quad$ **for** $j = 0 \rightarrow$ number of ResNet layers **do**
18: $\quad\quad e := x_i$
19: $\quad\quad x_i :=$ batch_normalization (x_i)
20: $\quad\quad x_i :=$ ReLU (x_i)
21: $\quad\quad x_i :=$ convolution (x_i)
22: $\quad\quad x_i := e + x_i$
23: $\quad$ **end for**
24: $\quad$ Calculate the loss between the result of ResNet and l_i, and backpropagate the gradient of the loss to the model.
25: **end for**

1) PACKET GATHERING

The packet gathering step is represented in lines 2 and 3 of Algorithm 1. To create a dataset using statistical features, the packets should be gathered from the various application sources first and then grouped by a network address. The flow F is defined as a 5-tuple $\Omega(F)$ that contains five elements: the source IP address, source port number, destination IP address, destination port number, and transport layer protocol [21]. In general, if the packets have the same 5-tuple information in

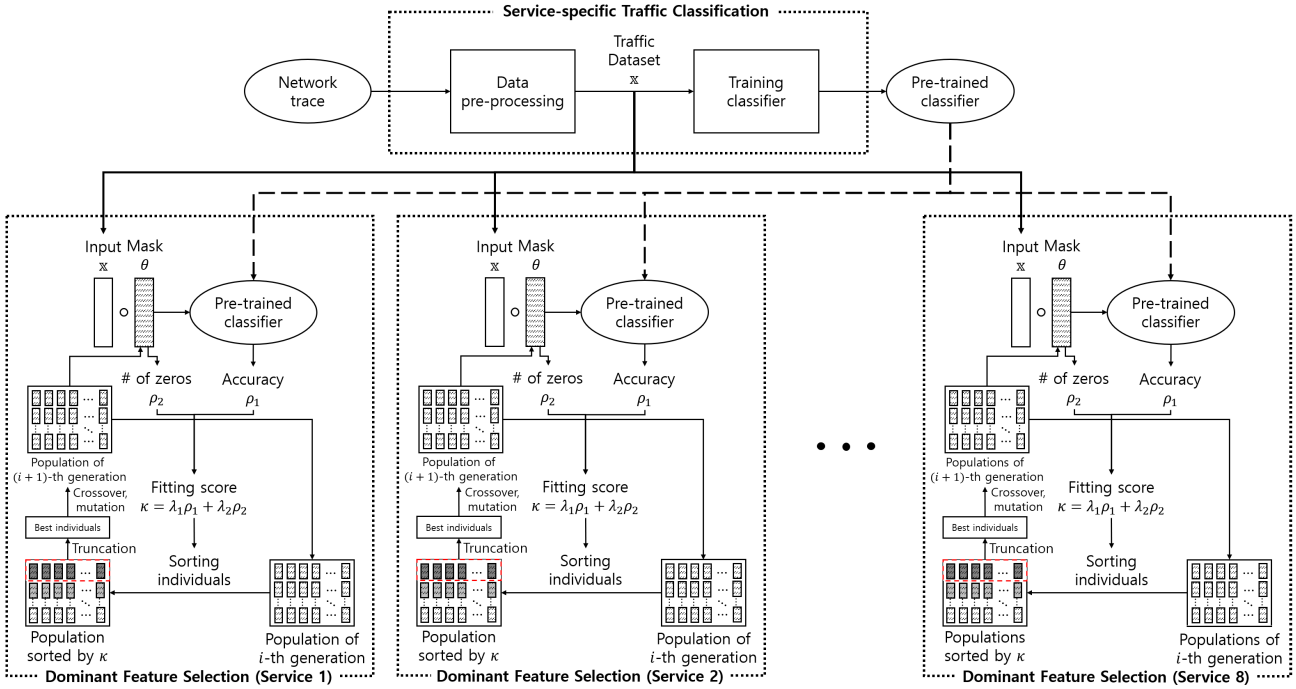


FIGURE 1. An overview of the traffic classification and dominant feature selection.

a certain TCP session duration, they are in the same flow. Most of the packets in a network flow that shares the same 5-tuple can be considered to serve the same application service because the application server provides one application service using one port number. Therefore, the packets in a flow that shares the same 5-tuple may have a similar behavior in the network, as shown by similar statistical features. The traffic classifier uses a bidirectional flow set that is composed of both F and its reverse directional flow $\bar{F}$ because most network architectures apply a server-client communication approach. Note that $\Omega(F)$ is the function returning the 5-tuple of the flow F.

2) DATA PREPROCESSING

The data pre-processing step computes the statistical features of the bidirectional flow set shown in lines 4-14 of Algorithm 1. The behaviors of the packets in the network are represented as statistical features, which are mainly revealed by the inter-arrival time, packet size, number of packets, and number of bytes [21]. Although the packets are encrypted, the packets serving the same application layer protocol have unique behaviors, and the protocols that serve a similar type of service show similar behaviors. For example, instant messaging services can cause bursty traffic, which can be shown in the statistical features such as a relatively short inter-arrival time and packet size. Therefore, the deep-learning-based traffic classifier can classify packets by service regardless of encryption by learning the distribution of statistical features that are different for each service. The traffic classifier uses bidirectional flow features and extracts 20 types of features, as shown in Table 2.

TABLE 2. Statistical flow features.

Features	Description	Value
Packet Size	The maximum, minimum, average, standard deviation of packet sizes in a flow.	8
Inter-Arrival Time	The maximum, minimum, average, standard deviation of inter-packet time in a flow.	8
Packets	Total number of packets in a flow.	2
Bytes	Total amount of bytes in a flow.	2

Data preprocessing involves feature extraction and service labeling. The former part aims to extract features from the flow $F = \{p_1, p_2, p_3, \ldots, p_n\}$, where F has n packets and p_i is the i-th packet. Because we deal with a bidirectional flow, the features of the reverse direction flow $\bar{F}$ are also needed. Therefore, 10 types of statistical features in one direction can be extracted, and there are 20 types of features in one bidirectional flow composed by F and $\bar{F}$. In flow F, we compute statistical features such as the inter-arrival time and packet size as follows:

- Statistical features: minimum, maximum, average, standard deviation are computed from the behavior vector $\mathbf{f} = [f_1\, f_2\, f_3\, \ldots\, f_k]$ as follows.

$$m_{\mathbf{f}} = \min\{f_1, f_2, \ldots, f_k\},\ M_{\mathbf{f}} = \max\{f_1, f_2, \ldots, f_k\}$$

$$\mu_{\mathbf{f}} = \frac{1}{k}\sum_{i=1}^{k} f_i,\ \sigma_{\mathbf{f}} = \frac{1}{k}\sum_{i=1}^{k}(f_i - \mu_{\mathbf{f}})^2$$

where $m_{\mathbf{f}}$, $M_{\mathbf{f}}$, $\mu_{\mathbf{f}}$, and $\sigma_{\mathbf{f}}$ are the minimum, maximum, average, and standard deviation of elements of $\mathbf{f}$, respectively. Note that the standard deviation is the sample standard deviation.

- The inter-arrival time: Arrival time of one packet is measured using UNIX time $\tau(p)$. The inter-arrival time features between two packets are computed as follows:

$$t_i = \tau(p_i) - \tau(p_{i-1})$$

The behavior vector of the inter-arrival time is $\mathbf{t} = [t_1\ t_2\ \ldots\ t_{n-1}]$. Thus, four types of statistical features can be computed, i.e. $m_{\mathbf{t}}$, $M_{\mathbf{t}}$, $\mu_{\mathbf{t}}$, and $\sigma_{\mathbf{t}}$.

- The packet size: the payload length field in the IP header gives the packet size feature. The DPI can look into the packet and search fields of the IP header. Therefore, the behavior vector of the packet size is $\mathbf{s} = [s_1\ s_2\ \ldots\ s_n]$ and four types of statistical features are computed: $m_{\mathbf{s}}$, $M_{\mathbf{s}}$, $\mu_{\mathbf{s}}$, and $\sigma_{\mathbf{s}}$.

Finally, the input vector $\mathbf{x}$ is composed as follows:

$$\psi = [m_{\mathbf{t}}\ M_{\mathbf{t}}\ \mu_{\mathbf{t}}\ \sigma_{\mathbf{t}}\ m_{\mathbf{s}}\ M_{\mathbf{s}}\ \mu_{\mathbf{s}}\ \sigma_{\mathbf{s}}\ n\ b]$$
$$\mathbf{x} = [\psi\ \bar{\psi}]$$

Note that ψ and $\bar{\psi}$ are input vectors of F and $\bar{F}$, respectively.

3) TRAINING CLASSIFIER

Before training, we need to conduct additional pre-processing steps, i.e., normalization and reshaping, because the deep-learning-based traffic classifier has to accept the two-dimensional input. The deep-learning-based traffic classifier is designed based on the architecture of a CNN, which is one of the most broadly used deep learning models, and is mainly used for image classification. The main idea of a CNN is to extract the local features from two-dimensional input data using kernels that extract the different local features. Statistical features are suitable for the method of extracting local features because those are composed of several features that appear in one feature. For example, the statistical features of the inter-arrival time in one direction are represented by four features such as minimum, maximum, average, and standard deviation. The input data should be normalized as [0, 1] to avoid biases because the domain of each feature is different. The input vector is reshaped to a matrix that can be used for the input of the CNN. Note that the rectified linear unit (ReLU) is used as the activation function, which is formulated as $\mathrm{ReLU}(x) = max(0, x)$. In addition, batch normalization is also used for regularization in each convolution layer. To adjust the capacity of our model, we also use the architecture of the residual network (ResNet) which is one of the models with the highest performance among the deep learning architectures [20]. To utilize complex data, a larger capacity model should be used to avoid the overfitting problem, and the deep-learning-based traffic classifier applies a ResNet model to allow the use of complex input data.

B. DOMINANT FEATURE SELECTION

We proposed a dominant feature selection method to explain how the deep learning model classifies traffic. In classification problems, there are key elements within the data that are the basis for classification. For example, in natural language processing (NLP), the subject and verb are the key elements, and the others are qualifiers used to explain them in the word tokens. Utilizing data with too many or unnecessary components for training may cause a higher complexity of the model. In fact, data with many components may lead to a higher accuracy. In other words, the classification accuracy also decreases because low-dimensional data have less information for the decision. Therefore, there is a trade-off between the classification accuracy and dimensions of the data, and the classifier needs a dimension-reduction technique that maximizes the accuracy.

We propose a dominant feature selection method based on a genetic algorithm as a dimension reduction technique. The aim of the proposed method is to find the optimal feature selection masks, minimizing the number of selected features and maximizing the classification accuracy. Hence, we formulated the objective function as a linear combination of two factors, namely, the number of masked elements and the classification accuracy. Here, ρ_1 is the number of dropped features and ρ_2 is the classification accuracy. Moreover, we maximize ρ_1 because maximizing the number of dropped features is equal to minimizing the number of selected features. This problem is formulated as follows:

$$\begin{aligned} \underset{\rho_1}{\text{maximize}} \quad & \lambda_1\rho_1 + \lambda_2\rho_2 \\ \text{subject to} \quad & \rho_1 = 0, 1, \ldots, l_{mask} \\ & 0 \le \rho_2 \le 1 \\ & \lambda_1 + \lambda_2 = 1 \\ & 0 \le \lambda_1 \le 1, 0 \le \lambda_2 \le 1 \end{aligned}$$

where λ_1 and λ_2 are the weights of ρ_1 and ρ_2, respectively, and are hyper-parameters that should be set beforehand. In addition, ρ_1 is an integer from zero to l_{mask}, where l_{mask} is the total number of features. Note that ρ_1 should be normalized in [0, 1] because ρ_1 is an integer and ρ_2 is a decimal value with the domain of [0, 1].

It is difficult to maximize the objective function because ρ_1 is an integer. Moreover, ρ_2 can differ even if the number of zeros is the same mask because the positions of the zero components in the mask determines the key component of the data for classifying the traffic. In other words, it can be difficult to maximize the objective function using optimization methods that simply adjust ρ_1. Therefore, the proposed method finds the feature selection masks using a genetic algorithm, which can maximize the accuracy by considering the position of the zero components. A genetic algorithm is a meta-heuristic algorithm inspired by inheriting the best chromosomes through the generations to allow the fittest to survive. With the proposed method, the chromosomes are considered as the feature selection masks, and the algorithm

generates a mask pool to maximize the objective function. Based on the algorithm, the proposed method generates an optimal feature selection mask that selects the least number of features without significantly compromising the classification accuracy.

The feature selection method is conducted as several iterations consisting of two steps, namely, best mask selection and offspring mask generation. The best mask selection step evaluates the fitting score of the parent masks and selects a few of the best masks through a roulette-wheel selection. After mask selection, the offsprings are created through a crossover and mutation. The optimal masks are created through sufficient iterations of the above steps toward a maximization of the objective function represented by the fitting score. Note that the fitting score is an indicator that represents the optimality. Algorithm 2 shows the entire procedure of the feature selection.

1) BEST MASK SELECTION

The best mask selection aims to pick a few best feature selection masks leaving the next generation. The genetic algorithm defines the chromosome with several genes composed of binary encoding for the expression of solutions. The proposed feature selection mask is presented as a chromosome shown as a binary string. The elements of the mask represented by genes choose the important features, as indicated by a "1" and remove the other features, as indicated by a "0". Therefore, the optimal feature vector $\mathbf{x}'$ is computed as the element-wise multiplication of the optimal feature selection mask and the original input vector.

$$\mathbf{x}' = \theta^\star \circ \mathbf{x}$$

where $\mathbf{x}$ is the original feature vector and $\theta^\star$ is the mask.

During the initial generation, the proposed method randomly creates M chromosomes composing the population and measures the fitting score. The chromosomes that are passed to the next generation are selected from the population through a roulette-wheel selection. The roulette-wheel selection is one of the methods used to pass on chromosomes with high fitting scores to the next generation. To avoid convergence to the local minimum in chromosome exploration, the method gives all candidate chromosomes a chance to be passed on to the next generation but gives a high probability of selection of chromosomes with high fitting scores.

In the i-th generation, the fitting score evaluation and best chromosome selection are conducted. The proposed method evaluates the fitting score of chromosomes in the parent population. The optimal feature selection mask should select the least number of features to maximize the accuracy. That is, the optimal masks consider the number of dropped features ρ_1 and classification accuracy ρ_2, which is derived using a pre-trained traffic classifier. The fitting score κ is formulated as the objective function to be maximized. The proposed method computes the fitting score κ as follows:

$$\kappa = \lambda_1 \rho_1 + \lambda_2 \rho_2$$

Algorithm 2 Procedure of Dominant Feature Selection

Require: Pre-trained traffic classifier, flow statistical feature dataset $\mathbf{x}$, the weight of dropped feature numbers λ_1, the weight of classification accuracy λ_2, the number of whole generation N, the number of individuals in a population M.

Ensure: Optimal feature selection mask $\theta^\star$ of a service.

Randomly generates the initial population θ_0.
for $i = 0 \rightarrow N-1$ **do**
  $\kappa_i = \varnothing$
  for $j = 1 \rightarrow M$ **do**
    Pick a individual θ_i^j from the i-th generation population θ_i.
    Compute ρ_1^j, which is the number of zeros in θ_i^j.
    $\mathbf{x}' = \theta_i^j \circ \mathbf{x}$
    Compute the accuracy ρ_2^j from the pre-trained traffic classifier using $\mathbf{x}'$.
    Compute $\kappa_i^j = \lambda_1 \rho_1^j + \lambda_2 \rho_2^j$
    $\kappa_i = \kappa_i \cup \{\kappa_i^j\}$
  end for
  Compute best individuals $\hat{\theta}_i$ by truncating the population based on κ_i
  $\theta_{i+1} = \varnothing$
  for $j = 1 \rightarrow M$ **do**
    Decide to perform elitism, crossover and mutation operation in Monte-Carlo manners.
    if Perform elitism operation **then**
      Randomly pick a individual $\hat{\theta}_i^1$ from $\hat{\theta}_i$
      $\theta_{i+1}^j = \hat{\theta}_i^1$
    end if
    if Perform crossover operation **then**
      Randomly pick two individuals $\hat{\theta}_i^1, \hat{\theta}_i^2$ from $\hat{\theta}_i$
      Compute θ_{i+1}^j by crossover operation.
    end if
    if Perform mutation operation **then**
      Randomly pick two individuals $\hat{\theta}_i^1, \hat{\theta}_i^2$ from $\hat{\theta}_i$
      Compute θ_{i+1}^j by mutation operation.
    end if
    $\theta_{i+1} = \theta_{i+1} \cup \{\theta_{i+1}^j\}$
  end for
end for
$\theta^\star = \theta_N$

where λ_1 and λ_2 are the weights of ρ_1 and ρ_2, respectively, and they are hyper-parameters that should be set beforehand. If ρ_1 is high, the number of dropped features is more important than the accuracy. The minimum features are selected, although the accuracy is slightly low. However, if λ_2 is high, the accuracy is more important and the number of zeros in the mask is relatively small. Therefore, there is a trade-off between the number of dropped features and accuracy.

After the measurement of the fitting score, the best chromosome selection is conducted to select chromosomes evaluated with high fitting scores through a roulette-wheel

selection. The population of selected chromosomes consists mostly of masks evaluated with a high fitting score and is inherited by the next generation. The masks converged to the optimal masks that have a high fitting score from sufficient iterations of the survival of the fittest.

2) OFFSPRING MASK GENERATION

The second step aims to generate offspring masks based on the selected masks. Basically, offspring masks are generated by two transformation techniques, i.e., crossover and mutation. These techniques are a process used to explore the entire possible chromosome pool and find better chromosomes. For example, if the number of components of the input data is l_{mask}, the number of possible masks that can apply the input data is $2^{l_{mask}}$. When the dimensions of the input data are more complex, the possible mask pool is exponentially expanded and the exhausting search method has extreme difficulty finding the optimal masks.

A crossover is an operation that merges a portion of the genes of parents. Specifically, a crossover chooses one random gene α in the parent chromosomes g_1, g_2 and separates each chromosome into two parts $g_1[0, \alpha]$, $g_1[\alpha, 20]$, $g_2[0, \alpha]$, and $g_2[\alpha, 20]$, where $g[\alpha, \beta]$ is a sub-array of array g from α to $\beta - 1$. It swaps parts of g_1 and g_2 as follows:

$$g_1 = g_1[0, \alpha] + g_2[\alpha, 20]$$
$$g_2 = g_2[0, \alpha] + g_1[\alpha, 20]$$

A mutation is an operation that changes a few genes in a chromosome. A bit flipping operation is used as the mutation operation because the proposed method uses binary-encoded chromosomes.

The proposed method has two hyper-parameters that have a probability of operating a crossover and mutation, such as the crossover and mutation rates. It operates a crossover and mutation with a certain probability, and thus some chromosomes have been changed, although some others have been preserved. Therefore, the proposed method provides optimal feature selection masks in a stochastic manner. Because offspring mask generation method are based on the genetic algorithm, they are of a probabilistic nature. In other words, with the genetic algorithm, a crossover and mutation randomly occur. Consequently, finding better chromosomes and maintaining the best individuals produces the optimal feature selection mask.

IV. PERFORMANCE EVALUATION AND EMPIRICAL ANALYSIS

In this section, we describe the performance of the deep-learning-based traffic classifier used to evaluate the accuracy, learning cost. To evaluate the performance of the proposed method, we carried out numerous experiments using real-world data.

A. EXPERIMENT SETTINGS

For fair evaluations, the public pcap datasets are used to build the training dataset. We adopted public pcap datasets from ISCX VPN-nonVPN, MACCDC, and WRCCDC, which have also been frequently used in other studies in traffic classification and include both encrypted and non-encrypted packets. Although the public pcap dataset has many packets that operate various protocols, the number of flows grouped by packets that share the same 5-tuple is insufficient to train deep-learning-based traffic classification model. To supply more training data, we gather the additional pcap data utilizing the server which generates packets of various protocols from a campus network. As a result, the entire dataset is composed of 49 applications, as shown in Table 3. In Table 3, a number column represents the number of flows. We set the number of data by each service similar to avoid biased training. Moreover, for practical use, packets of one flow are gathered for 900 seconds without considering a TCP session timeout. To create a training dataset, we implemented the

TABLE 3. Applications and services for the training dataset.

Services	Applications	Flows	Total
Instant messaging	MSN	1,381	5,940
	KakaoTalk	3,259	
	Facebook Messenger	181	
	Jabber	1,119	
E-mail	SMTP	751	5,546
	IMAP	751	
	SMTPS	751	
	IMAPS	751	
	POP3	751	
	POPS	751	
	GMail	751	
	LotusNote	289	
File transfer	HTTP_Download	731	5,848
	FTP_Control	731	
	FTP_DATA	731	
	RSYNC	731	
	AFP	731	
	GoogleDrive	731	
	MS_OneDrive	731	
	Dropbox	731	
P2P	Bitcoin mining	1,856	5,193
	BitTorrent	948	
	Gnutella	2,389	
Remote access	SSH	951	5,706
	RDP	951	
	VNC	951	
	NFS	951	
	Telnet	951	
	TeamViewer	951	
Streaming	YouTube	1,801	5,667
	RTSP	1,801	
	NetFlix	1,437	
	Twitch TV	400	
	SoundCloud	105	
	RTP	123	
VoIP	Skype	2,906	5,563
	SMPP	718	
	SIP	1,568	
	H323	173	
	CiscoSkinny	198	
Web surfing	Amazon	901	5,729
	Google	901	
	Twitter	901	
	Facebook	901	
	Yahoo	617	
	eBay	190	
	Wikipedia	901	
	Instagram	104	
	LinkedIn	313	
Total			45,192

data pre-processing program using the nDPI library, which can detect the application. The nDPI is an open-source DPI library that provides information on both the payload and header of the packet. Based on the nDPI, we gather information about the application layer protocol, IP address, and TCP port number. Although the public pcap dataset applies the preassigned labels, we need to assign labels to the additional dataset collected by our campus networks. We leverage the nDPI library to assign the additional dataset collected by our campus networks. After gathering the information, we implemented the rest of the pre-processing program that groups the packets into the flow and extracts the flow statistics. The deep-learning-based traffic classifier is implemented using TensorFlow. The entire experiments are conducted by a server with an Intel i9-8980XE CPU, 64GB of RAM, and an NVIDIA GeForce GTX 2080.

B. PERFORMANCE EVALUATIONS OF SERVICE-SPECIFIC TRAFFIC CLASSIFICATION

For training the deep learning model, we divided the 70% of the dataset into training dataset and 30% into test dataset, and all evaluations are based on the test dataset. The parameters are initialized at random, and a batch normalization layer is used to mitigate the effort required to regularize the parameters by forming a similar distribution in each layer. There are some hyper-parameters to be tuned before the training, such as batch size and number of epochs. We found the two hyper-parameters above through an adequate number of experiments with a batch size of 300 and 5,000 epochs. Moreover, we conducted experiments by adjusting other hyper-parameters such as the number of filters in the convolution layer and the number of layers in the residual block. Note that one residual block consists of several convolution layers and batch normalization layers, and the entire model is constructed by stacking several residual blocks. Figures 2(a) and 2(b) show the test cost and test accuracy according to the iterations. We conducted experiments by changing the number of layers from 4 to 8 and using 64 and 128 filters. It can be seen that the greater the number of filters and the number of layers, the higher the classification accuracy, and the faster the cost convergence. In general, if the data has a high dimension, a more complex deep learning model is required [22]. ResNet has an advantage that easily controls the complexity of the model by tuning the number of residual blocks and convolution filters. Hence, the model should have sufficient complexity to adequately describe the dataset by increasing the number of layers and filters.

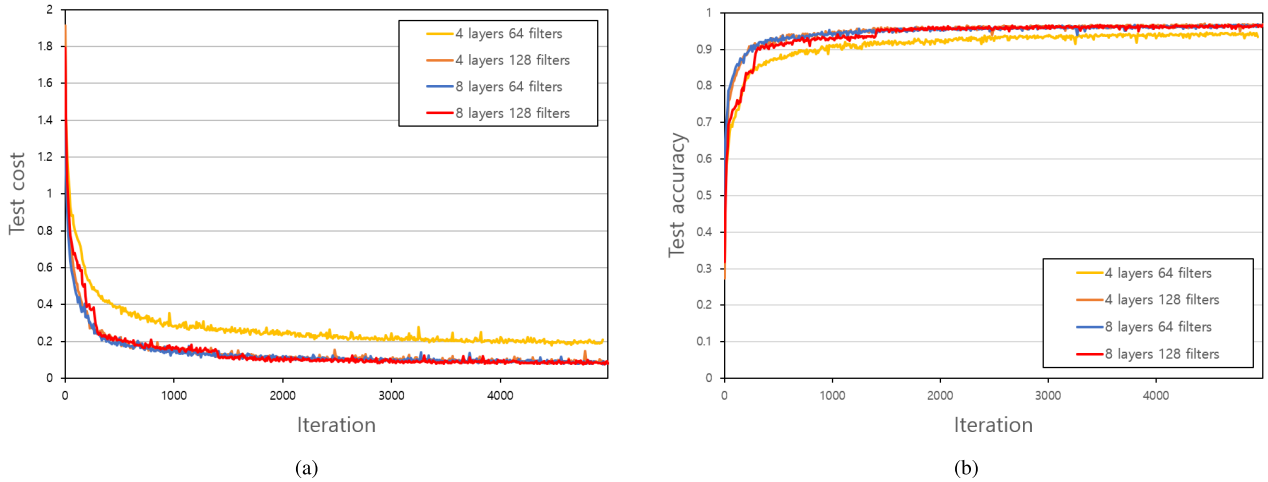
Figures 3(a) and 3(b) show the test cost and test accuracy according to hyper-parameters such as the layers and filters. As shown in the figures, the model with 128 filters shows the maximum accuracy and minimum cost. It can be seen that when the deep learning model is trained by 128 filters, the classifier can achieve a sufficient performance. Hence, we use a sufficiently trained model whose numbers of layers and filters are 16 and 128, respectively.

The confusion matrix shows how much of the data are correctly classified and helps to calculate several metrics such as true positive, true negative, false positive, and false negative. Figure 4(a) shows the confusion matrix according to the classes. In the confusion matrix, it can be seen that the overall classification accuracy of the model is approximately 97.24%. Moreover, the precision, recall, and F1-score of each service can also be calculated based on the confusion matrix, as shown in Figure 4(b). Precision is the ratio of the amount of actual data 'A' from the total amount of data predicted as 'A', and recall is the ratio of the amount of data predicted as 'A' from the total amount of actual data 'A'. However, evaluating the performance based on precision and recall may be complicated because an imbalanced dataset may cause different trends in the precision and recall. The F1-score is the harmonic mean of precision and recall, and thus the F1-score can explain the performance of the model, which has different trends in the precision and recall.

C. EMPIRICAL ANALYSIS OF SERVICE-SPECIFIC TRAFFIC CLASSIFICATION

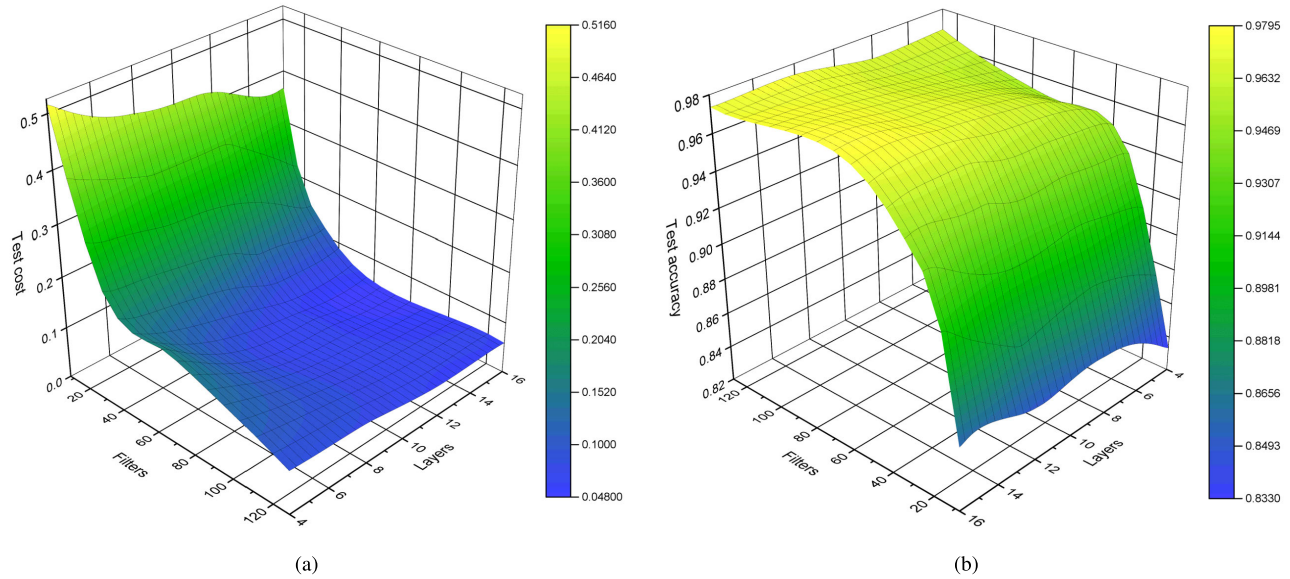
In this section, we analyze how deep-learning-based traffic classifier classifies traffic into services. We conducted experiments by adjusting the two hyper-parameters, λ_1 and λ_2, of the proposed genetic algorithm-based explanatory method and evaluated the performance according to the hyper-parameters. Note that we set the sum of λ_1 and λ_2 as 1 to fairly measure the effects of both variables ρ_1 and ρ_2. Based on the feature selection masks generated by each experiment, we define the dominance rate, which represents the importance of features and analyzes the statistical features of each service.

Figure 5 shows how the fitting score, accuracy, and number of dropped features change throughout the generations depending on the weights λ_1 and λ_2. From each service, it can be observed that, as λ_1 increases, the number of dropped features increases. At the same time, as λ_2 decreases, the average accuracy of the mask pool decreases. The first column is the result in which λ_1 is set to 0.1 and λ_2 is set to 0.9, where the proposed method aims to search for masks with a higher classification accuracy. Because it is generally better to have more data for a deep learning model with respect to classifying the services, a higher accuracy and more robustness in the classification are shown. The second column shows the results in which both λ_1 and λ_2 are set to 0.5, meaning that the approach aims to find masks with both a higher accuracy and a larger number of dropped features. It can be observed that the proposed method tends to show a balanced and complementary behavior between accuracy and the number of dropped features. The third column shows that, when λ_1 is set to 0.9 and λ_2 is set to 0.1, it indicates that the proposed method aims to find masks with a larger number of dropped features, rather than achieving a higher accuracy. Therefore, the result shows that the number of dropped features is much higher than that of the other two results with different weights. However, it shows the lowest



(a) (b)

FIGURE 2. (a) Test cost according to iterations and (b) test accuracy according to number of iterations.



(a) (b)

FIGURE 3. (a) Test cost and (b) test accuracy according to the number of layers and filters.

accuracy result in general. In addition, it can be observed that the result of a "web surfing" service shows a relatively higher accuracy even when the weight λ_1 is higher and λ_2 is lower. That is because a web surfing service is a generic class and therefore has more general characteristics than other specific services, it allows the deep learning model to require much fewer features to classify the "web surfing" than the other services.

Figure 6 shows the fitting score, accuracy, and number of dropped features in the last generation of the proposed method. From each service, it can be observed that the average accuracy and number of dropped features are dependent on the weight values λ_1 and λ_2. The fitting scores tend to be higher when the weight λ_1 is higher because it requires less exploration when finding masks with more dropped features. By contrast, when λ_2 is higher, relatively lower fitting scores are shown because a much more thorough exploration is required to find the proper masks with higher accuracy.

In general, the CNN model operates the classification process by collecting several local features. It is important to discover key features that are used as criteria used for the deep learning model to classify. The XAI is an explanatory method that describes how the deep learning model can classify the characteristics of a certain object by extracting the key features such as eyes, nose, and ear [16]. We found the key features of the flow by designing the dominant feature selection method based on a genetic algorithm as a method of the XAI. We applied the proposed dominant feature selection

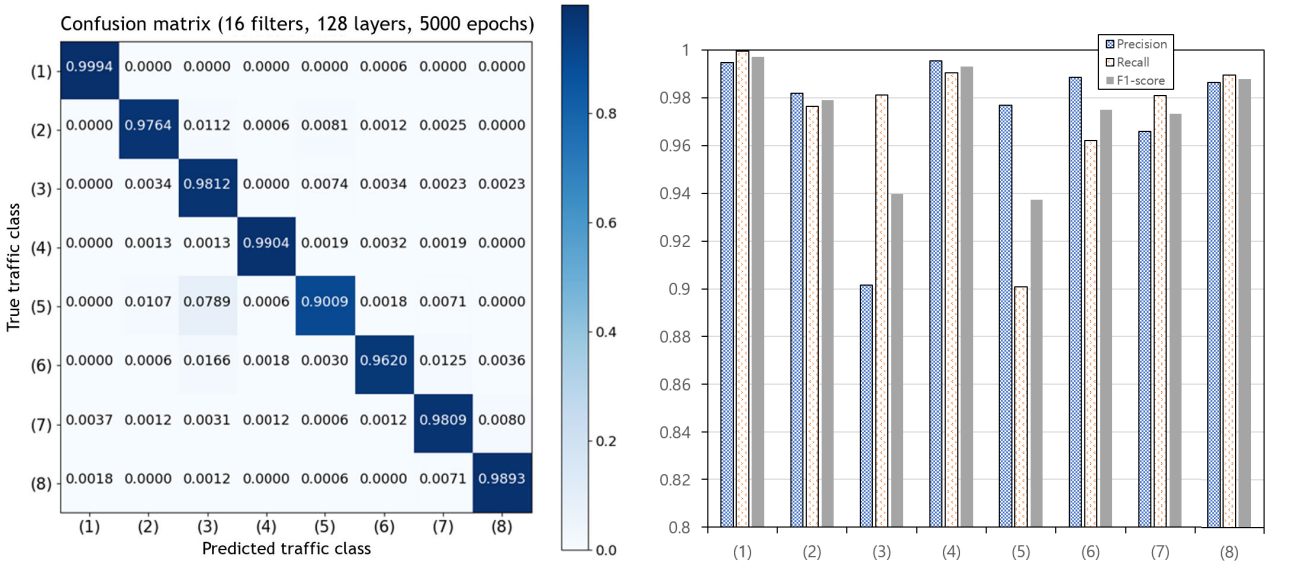


FIGURE 4. (a) Confusion matrix according to traffic classes (the ratio of predicted results to the true traffic class). (b) Precision, recall, and F1-score according to each service. Services 1-8 are as follows: (1) instant messaging, (2) E-mail, (3) file transfer, (4) P2P, (5) remote access, (6) streaming, (7) VoIP, and (8) Web surfing.

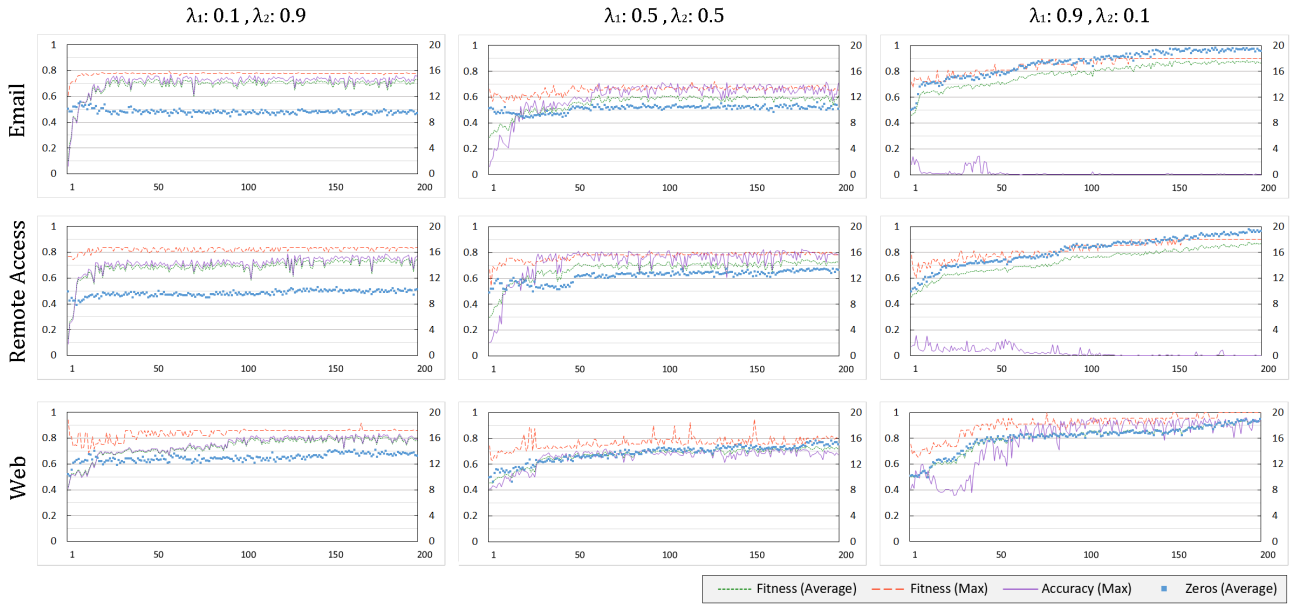


FIGURE 5. Fitting score, accuracy, and number of zeros per generation for 3 services: email, remote access, and web surfing. The X-axis indicates the generations. The Y-axis on the left indicates the fitting score and accuracy. The Y-axis on the right indicates the number of zeros, which equals to the number of dropped features.

method by changing the hyper-parameters such as the weight of the dropped features λ_1 and the accuracy λ_2. We defined the dominance rate to illustrate the extent to which each feature is dominant in classifying the traffic. The dominance rate is defined according to the proportion of the number of selected key features to the number of features in the entire experiment as follows:

$$I = \frac{K}{N} \times 100, \ 0 \leq K \leq N,$$

where I is the dominance rate of each feature, and K represents the number of times a key feature is selected in each experiment. The experiment was conducted starting from λ_1 at 0.1, and λ_1 was gradually increased by 0.1, until reaching 0.9. Then, λ_2 is determined depending on λ_1, starting from 0.9 to 0.1, making $\lambda_1 + \lambda_2 = 1$. Consequently, the entire experiments were conducted 10 times and N was set to 10. Figure 7 shows the dominance rate of each statistical flow feature for each service.

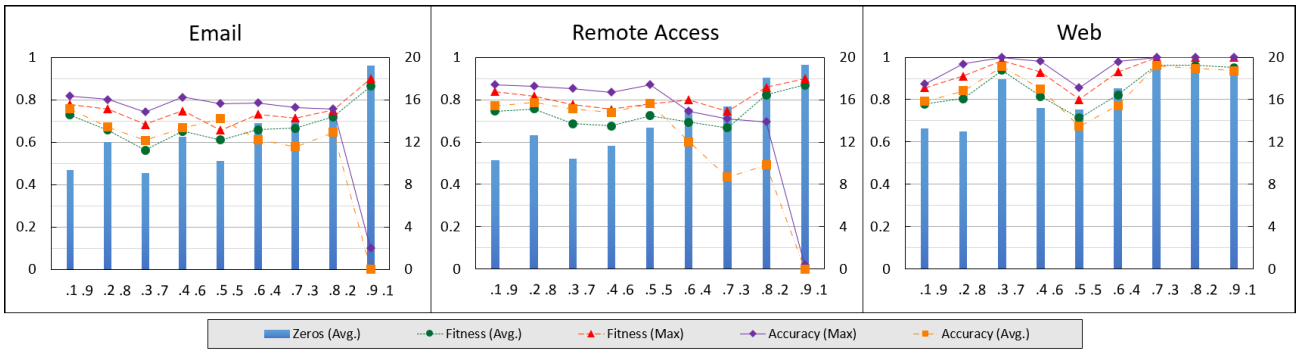


FIGURE 6. Number of zeros, fitting score, and accuracy depending on different weight settings. The X-axis indicates the weights λ_1 and λ_2. The Y-axis on the left indicates the fitting score and accuracy. The Y-axis on the right indicates the number of zeros, which is equal to the number of dropped features.

Service \ Feature	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)	(14)	(15)	(16)	(17)	(18)	(19)	(20)
Chat	22%	11%	39%	37%	33%	44%	69%	11%	24%	23%	66%	24%	14%	27%	41%	37%	34%	68%	12%	53%
Email	54%	30%	8%	52%	26%	22%	0%	90%	56%	38%	56%	36%	11%	3%	87%	54%	2%	78%	38%	0%
File Transfer	57%	12%	68%	76%	34%	90%	12%	12%	56%	0%	6%	22%	4%	41%	22%	72%	14%	41%	19%	46%
P2P	7%	38%	49%	22%	48%	73%	3%	0%	80%	13%	26%	38%	9%	0%	57%	0%	2%	56%	17%	0%
Remote Access	27%	27%	13%	61%	36%	0%	0%	98%	1%	13%	37%	46%	22%	53%	8%	0%	24%	48%	61%	71%
Streaming	16%	24%	50%	0%	19%	14%	22%	22%	0%	11%	28%	23%	0%	88%	3%	0%	2%	0%	90%	12%
VoIP	12%	18%	26%	11%	14%	6%	34%	21%	0%	0%	14%	39%	22%	62%	44%	22%	26%	56%	0%	67%
Web	1%	1%	22%	22%	1%	11%	11%	0%	0%	0%	7%	11%	27%	21%	39%	9%	41%	3%	9%	19%

FIGURE 7. Importance of features for each service. Note that (1) is the number of packets, (2) is the amount of bytes, (3), (4), (5), and (6) are the minimum, maximum, average, standard deviation of inter-arrival time, respectively, and (7), (8), (9), and (10) are the minimum, maximum, average, standard deviation of packet size, and (11), (12), (13), (14), (15), (16), (17), (18), (19), and (20) is the features of reverse direction, respectively.

It can be seen that the deep-learning-based traffic classifier does not classify traffic into a service using all features, but classifies it using only specific features. We conducted nine experiments by changing the λ_1 and λ_2 and averaged the results of the experiments. When the proposed method completes a sufficient number of iterations, it generates 200 feature selection masks of each service for one experiment and picks the top-10 masks, wihch achieve the highest accuracy. Figure 7 shows the dominance rate for each statistical feature that affects the accuracy. If the dominance rate is high, it can help increase the classification accuracy or fitting score, namely, it can be a candidate for the key feature. Otherwise, these features have less influence on the traffic classification, and thus they can be candidates of unnecessary features. A dominance rate of 100% implies that the classifier always uses the features to classify the traffic into the service, namely, the feature is used as the core feature of the service. Because the 0% dominance rate implies that the feature does not affect the classification, it indicates that the feature is irrelevant for classification.

To measure how much each feature contributes to the classification accuracy, we define the elimination threshold and measure the accuracy of each elimination threshold. If the dominance rate of a feature is lower than the elimination threshold, the feature is removed. Figure 8 shows the accuracy according to the elimination threshold. As the elimination threshold increases, the number of features removed increases, and thus the overall classification accuracy tends to decrease. Because features with a low dominance rate are simply removed through the elimination threshold, the correlations between features are not considered. Consequently, when removing a feature that is related to other features, a fluctuation that temporarily decreases the accuracy may occur. In the case of the "instant messaging" class, the relationship between the number of removed features and accuracy has a monotonous decrease, which means that there is little correlation for each feature. In the case of the "web surfing" class, there may be slight fluctuations in accuracy, although the overall accuracy does not decrease significantly as the number of removed features increases because most of

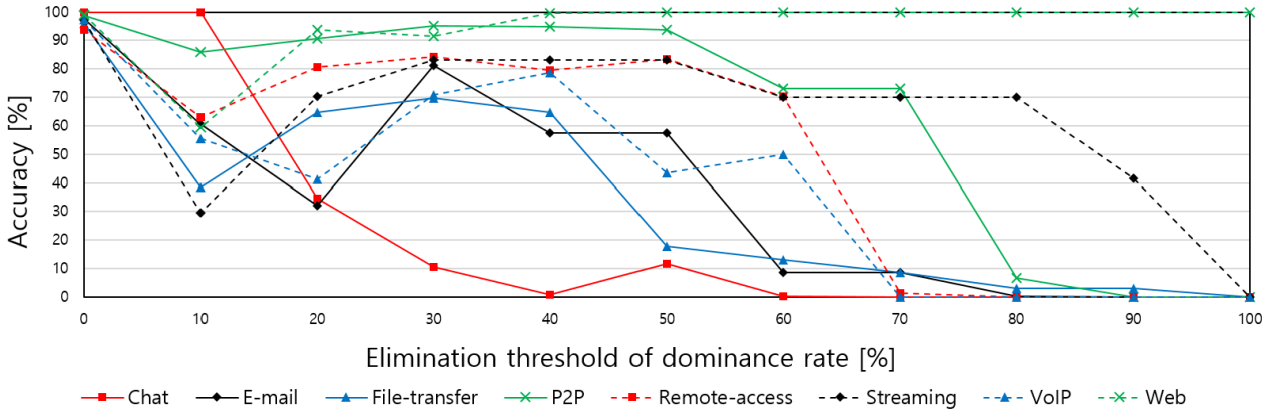


FIGURE 8. Test accuracy according to the elimination threshold of the dominance rate. The X-axis indicates the elimination threshold. The Y-axis indicates the accuracy in the measurements using features having a dominance rate of a certain threshold or higher.

the features have a low dominance rate. Most of the services have several key features, although the "web surfing" class has few candidates. In other words, the deep learning model of our method tends to classify traffic that it cannot classify into the "web surfing" class. This can be explained by the fact that the traffic of "web surfing" services tends to show a universal property that represents the general behaviors of Internet services. For example, most Internet services are based on the hyper-text transfer protocol (HTTP), and thus the traffic of a "web surfing" service based on HTTP can show various behaviors in the network. Therefore, all of the other traffic that does not show the specific behaviors of the other services can be classified into the "web surfing" class, which is regarded as a generic class. By contrast, there are many correlations between features, and thus a fluctuation in the accuracy can frequently occur.

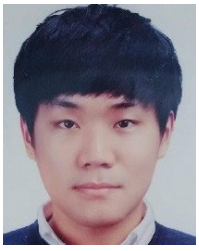
V. CONCLUSION

In this study, we proposed a explanatory method of the deep-learning-based traffic classifier based on a genetic algorithm. Further, we implemented the deep-learning-based traffic classifier based on the ResNet model for demonstrating the proposed explanatory method. We designed the dominant feature selection method as a explanatory method based on a genetic algorithm to generate an optimal feature selection mask. The proposed explanatory method generates the optimal feature selection masks by grafting the deep-learning-based traffic classifier's result onto the evaluation of the chromosome in a genetic algorithm. The feature selection masks are used to extract the key feature subset from the entire feature set by considering the trade-off between the classifier's accuracy and the number of unnecessary features. We conducted several experiments for reflecting the stochastic property of a genetic algorithm and computed the importance rate through the feature selection masks. Through the importance rate, we explained the mechanism of the deep-learning-based traffic classifier by investigating the key features of each Internet service. In the future, we plan to design a key feature selection algorithm for finer-grained application-specific traffic classifiers. In addition, we will improve the convergence speed of the genetic algorithm to enable real-time key feature selection.

REFERENCES

[1] H. Kim and N. Feamster, "Improving network management with software defined networking," *IEEE Commun. Mag.*, vol. 51, no. 2, pp. 114–119, Feb. 2013.
[2] M. Karakus and A. Durresi, "Quality of service (QoS) in software defined networking (SDN): A survey," *J. Netw. Comput. Appl.*, vol. 80, pp. 200–218, Feb. 2017.
[3] A. Adadi and M. Berrada, "Peeking inside the black-box: A survey on explainable artificial intelligence (XAI)," *IEEE Access*, vol. 6, pp. 52138–52160, 2018.
[4] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, "ZOO: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in *Proc. 10th ACM Workshop Artif. Intell. Secur.*, Nov. 2017, pp. 15–26, doi: 10.1145/3128572.3140448.
[5] M. Usama, A. Qayyum, J. Qadir, and A. Al-Fuqaha, "Black-box adversarial machine learning attack on network traffic classification," in *Proc. 15th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Jun. 2019, pp. 84–89.
[6] D. Gunning, "Explainable artificial intelligence (XAI)," in *Defense Advanced Research Projects Agency (DARPA), nd Web*. Arlington, VA, USA: Defense Advanced Research Projects Agency (DARPA), 2017.
[7] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, and F. Herrera, "Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI," *Inf. Fusion*, vol. 58, pp. 82–115, Jun. 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1566253519308103
[8] K. Zhou, W. Wang, C. Wu, and T. Hu, "Practical evaluation of encrypted traffic classification based on a combined method of entropy estimation and neural networks," *ETRI J.*, vol. 42, no. 3, pp. 311–323, Jun. 2020.
[9] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware traffic classification using convolutional neural network for representation learning," in *Proc. Int. Conf. Inf. Netw. (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, pp. 18042–18050, 2017.
[11] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescape, "Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges," *IEEE Trans. Netw. Service Manage.*, vol. 16, no. 2, pp. 445–458, Jun. 2019.
[12] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapè, "MIMETIC: Mobile encrypted traffic classification using multimodal deep learning," *Comput. Netw.*, vol. 165, Dec. 2019, Art. no. 106944. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128619304669

[13] A. Montieri, D. Ciuonzo, G. Bovenzi, V. Persico, and A. Pescape, "A dive into the dark Web: Hierarchical traffic classification of anonymity tools," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 3, pp. 1043–1054, Jul. 2020.
[14] L. Grimaudo, M. Mellia, and E. Baralis, "Hierarchical learning for fine grained Internet traffic classification," in *Proc. 8th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Aug. 2012, pp. 463–468.
[15] W. Zheng, C. Gou, L. Yan, and S. Mo, "Learning to classify: A flow-based relation network for encrypted traffic classification," in *Proc. Web Conf.*, Apr. 2020, pp. 13–22, doi: 10.1145/3366423.3380090.
[16] W. Samek, T. Wiegand, and K.-R. Müller, "Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models," 2017, *arXiv:1708.08296*. [Online]. Available: http://arxiv.org/abs/1708.08296
[17] A. Morichetta, P. Casas, and M. Mellia, "EXPLAIN-IT: Towards explainable AI for unsupervised network traffic analysis," in *Proc. 3rd ACM CoNEXT Workshop Big DAta, Mach. Learn. Artif. Intell. Data Commun. Netw. Big-DAMA*, 2019, pp. 22–28, doi: 10.1145/3359992.3366639.
[18] R. C. Fong and A. Vedaldi, "Interpretable explanations of black boxes by meaningful perturbation," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 3429–3437.
[19] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 2048–2057.
[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[21] J. Zhang, X. Chen, Y. Xiang, W. Zhou, and J. Wu, "Robust network traffic classification," *IEEE/ACM Trans. Netw.*, vol. 23, no. 3, pp. 1257–1270, Aug. 2015.
[22] P. Agrawal, R. Girshick, and J. Malik, "Analyzing the performance of multilayer neural networks for object recognition," in *Computer Vision—ECCV* (Lecture Notes in Computer Science), vol. 8695, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham, Switzerland: Springer, Sep. 2014, doi: 10.1007/978-3-319-10584-0_22.



SEYOUNG AHN received the B.E. degree in computer science and engineering from Hanyang University, South Korea, in 2018, where he is currently pursuing the M.S.-leading-to-Ph.D. degree in computer science and engineering. Since 2018, he has been with the Computer Science and Engineering, Hanyang University of Engineering, South Korea. His research interests include network security and network management with machine learning techniques.

JEEHYEONG KIM received the B.E. and Ph.D. degrees in computer science and engineering from Hanyang University, South Korea, in 2015 and 2020, respectively. He is currently the Postdoctoral Visiting Scholar with the Information and Intelligent Security Laboratory, Kennesaw State University, Marietta, GA, USA. His research interests include interference management for cellular communications, military communications using satellites, the IoT security, and applied deep learning to wireless communications.

SOO YOUNG PARK received the M.D. degree from the School of Medicine, Kyungpook National University, South Korea, in 1999, and the Ph.D. degree from the Postgraduate School of Internal Medicine, Kyungpook National University, in 2006. He is currently an Associate Professor with the Department of Gastroenterology and Hepatology, Kyungpook National University Hospital.

SUNGHYUN CHO (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in computer science and engineering from Hanyang University, South Korea, in 1995, 1997, and 2001, respectively. From 2001 to 2006, he was with the Samsung Advanced Institute of Technology, and the Telecommunication Research and Development Center, Samsung Electronics Company Ltd., where he has been involved in the design and standardization of MAC and network layers of WiBro/WiMAX and 4G-LTE systems. From 2006 to 2008, he was a Postdoctoral Visiting Scholar with the Department of Electrical Engineering, Stanford University. He is currently a Professor with the Department of Computer Science and Engineering, Hanyang University. His research interests include 5th generation mobile communications, software defined networks, and vehicular communication systems. He is also a member of the board of directors of the Institute of Electronics and Information Engineers (IEIE) and the Korean Institute of Communication Sciences (KICS).