IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 12 December 2024, accepted 24 December 2024, date of publication 31 December 2024, date of current version 10 January 2025.

Digital Object Identifier 10.1109/ACCESS.2024.3525000

RESEARCH ARTICLE

Data Augmentation-Based Enhancement for Efficient Network Traffic Classification

CHANG-YUI SHIN[1], YANG-SEO CHOI[2], AND MYUNG-SUP KIM[3], (Member, IEEE)
[1]C4ISR System Development Quality Team, Defense Agency for Technology and Quality, Daejeon 35409, South Korea
[2]Department of Cyber Security Research Division, Electronics and Telecommunications Research Institute, Daejeon 34129, South Korea
[3]Department of Computer Convergence Software, Korea University, Sejong 30019, South Korea

Corresponding author: Myung-Sup Kim (tmskim@korea.ac.kr)

This work was supported in part by the "Regional Innovation Strategy (RIS)" through the National Research Foundation of Korea (NRF), Ministry of Education (MOE) under Grant 2021RIS-004; and in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP), Development of Security Monitoring Technology Based Network Behavior Against Encrypted Cyber Threats in Information and Communication Technology (ICT) Convergence Environment, Korean Government under Grant 00235509.

ABSTRACT The necessity of Network traffic classification is becoming increasingly significant as users' applications and devices become more diverse and prevalent. As encryption becomes the norm for security reasons, the traffic classification problem is not easily solved. In this work, we provide an inductive counterevidence to the vague belief that deep learning models can perform well and outperform tree-based machine learning models in all aspects across domains, especially in the network traffic classification domain. We address the problem of finding an efficient encrypted traffic classification method in resource-constrained situations in the network traffic classification domain by limiting the scope of our research. Using the first packet, we converted packet headers and encrypted partial payloads into tabular data through a standardized format. We used them as the same inputs for lightweight deep learning and tree-based machine learning models, analyzed their performance, and identified efficient models. Next, we improved the performance of the previously selected efficient traffic classifier through data augmentation methods. Augmentation was performed to a degree that did not significantly damage the original data distribution so that the augmented dataset's class distribution did not interfere with model learning. We applied two fundamentally contrasting methods to augment traffic data, depending on whether the basis of data augmentation is individual data or the entire data. Data augmentation increased the accuracy of the machine learning model by 0.26%, which complemented the machine learning model's performance in network traffic classification and made the machine learning model outperform the lightweight deep learning model.

INDEX TERMS Network traffic classification, data augmentation, generative adversarial networks, tabular, robustness.

I. INTRODUCTION

In recent years, the rise of on-demand services from mobile devices, including wearables and vehicles, has led to an extreme increase in the amount of data being sent and received across online networks worldwide. In this situation, Network Traffic Classification (NTC) technology is receiving more attention due to its growing importance in security, intelligent network management, and stable services. As time goes by, the need for NTC is further highlighted by its potential to improve the quality of experience from a user perspective and to positively impact service quality and automation from a network service management perspective [1].

Chronologically, the earliest NTC at the starting point was developed based on ports, which did not have good results in terms of accuracy and reliability [2], [3]. The subsequent exponential growth of applications using random or non-standard port numbers further reduced the reliability of traffic classification, leading to the emergence of payload-based approaches. However, the Deep Packet Inspection (DPI) method [4] of inspecting the payload soon also became

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

a problem as the payload of network traffic was increasingly encrypted to enhance personal security [5], [6]. As the packet payload that is the subject of NTC is encrypted, it is being redefined as Encrypted Network Traffic Classification [7], [8], [9], [10]. Gradually, the percentage and share of encrypted network traffic have proliferated over the past few years and are expected to continue to do so [11]. The newly entered flow statistics-based methods [12], [13] are to find reasonable statistical characteristics of network traffic based on flow-level measurements (i.e., packet lengths, packet inter-arrival time, flow duration, etc.) empirically.

Feature selection and feature engineering based on researchers' empirical factors appear to be in place to optimize the overall performance metrics of encrypted traffic classification. Empirical features for network traffic classification have been applied in line with the evolution of the model. In such cases, it has been common to apply simple statistical features [14], raw bytes [15], mutual arrival time [16], payload size [17], [18] to traditional machine learning models (C4.5, decision trees, naive Bayes, etc.) or deep learning models like neural networks based on MLP (Multi-Layer Perceptron) [19]. One-dimensional or two-dimensional Convolutional Neural Network (CNN) models [20], [21] have also been added to its scope of application. In fact, from this time, depending on the logical concept of processing network traffic established by the researcher, network traffic was applied to CNN models specialized in image feature classification or Recurrent Neural Network (RNN) models specialized in time series data processing [22], [23]. There have also been useful approaches that blend the strengths of CNN and RNN models [24]. In recent years, Transformer [25] models and Bidirectional Encoder Representations from Transformers (BERT) [26] models have emerged that guarantee significant performance, represented by positional embedding and self-attention mechanisms, which continue to expand into the optional area of multimodal inputs [9], [27], [28], [29].

Despite the massive computational time required, using DL models with good performance is widespread in various fields. However, in the control domain of network management, tasks such as traffic classification and congestion control may be more feasible to make management decisions within milliseconds to ensure quality of experience. So, while automated decision-making is important, efficient and immediate processing times may be a priority. Furthermore, a solution tailored to specific tasks may be essential since resources are often limited in distributed environments. Even if the deep learning model has a slight performance advantage over the machine learning model [30], it can be proper not to use only performance as a criterion to determine whether it is usable in resource-constrained environments.

For a more objective comparative case study of the above subject, we will refer to our past work to our past work [9] to transform the input stage of another lightweight BERT model to the NTC domain to improve its performance and efficiency. In other words, we will apply transfer learning, packet embedding, and Comparative Learning to BERT to compare the basic BERT model and other lightweight models, such as DistilBERT, regarding performance and efficiency. We also designed to extend the adaptation of ALBERT, based on another well-known Transformer model, which shows substantial performance among deep learning models in Sections IV and V.

Deep learning models are a highly superior approach to problem-solving in many areas, including computer vision and natural language processing, where they can outperform machine learning models regarding results. In recent years, deep learning models have achieved remarkable results in the above fields and other fields requiring multimodal inputs, and its effectiveness has dramatically increased the social attention on deep learning. In many cases, deep learning outperforms traditional machine learning approaches. However, for tabular data, tree-based machine learning approaches such as eXtreme Gradient Boosting (XGBoost) [31] and Light Graient-Boosting Machine (LigthGBM) dominate [32], [33].

Although there is a lot of research on encrypted network traffic classification, they can be divided into two categories based on the materials (i.e., inputs into the model) used for classification. They can be categorized into those focusing on first-packet classification [9] and those dealing with post-analysis of complete packet traces, which include flow statistics and time-related features. Most of them focus on classifying flows after observing several initial packets (i.e., flows or several packets) rather than the first packet [34].

As datasets are not immune to the problem of data sufficiency, there are approaches to introducing new synthetic data through data augmentation to help models learn and perform better. The tendency to apply data augmentation methods started in the image domain and has expanded to various fields and, more recently, to the NTC domain. Through the augmented data, the robustness of models increases by broadening their coverage, as it can represent data patterns not present in the original dataset [1], [35], [36], [37]. Regardless of the choice of model, the above means that there is a necessity and possibility for different approaches to the robustness of learning models. For many classification tasks based on supervised learning, data containing relatively insufficient classes due to quantitative differences between classes can have a negative impact on learning. Therefore, it can be a countermeasure to improve learning effectiveness by ensuring adequate data augmentation in relation to the original class distribution [38].

In this paper, we transformed the packet header and some payloads of NTC domain data into a standardized tabular format. We experimented with model robustness by applying data augmentation techniques to NTC using two methods, jittering and GAN, which have contextual differences in augmentation. We generated synthetic data sets that increase the robustness of the model while maintaining the labels

accurately [35]. Then, we tested on the augmented data using one of the efficient models selected through performance comparison experiments with deep learning and machine learning models. We also presented sophisticated analysis on the augmented data based on the experiments.

The proposed data augmentation methods in this paper have three main contributions that are dominant in NTC problems.

- **Data Augmentation-Based Enhancement.**To support realistic usability with limited resources in the encrypted network traffic classification problem, the basic step was to preprocess each packet header and partial payload into a standardized tabular format and use them as inputs for the model. Machine learning models require relatively less input data than deep learning models. Therefore, it is noteworthy that we performed comparative analysis with lightweight deep learning and machine learning models using only the public header information and the encrypted partial payload. Through various models and experiments, we showed that applying the machine learning model is more usable and superior to using time-consuming deep learning in resource-constrained situations. This is a remarkable result compared to studies [39] and [40] that rely heavily on encrypted portions for performance, efficiency, and usability. Data augmentation increased the accuracy of the machine learning model by 0.26%, making it outperform the lightweight deep learning model.
- **Two Contrasting Data Augmentation.**The raw data that a model can learn from often contains imbalanced classes, and this minority class can negatively impact model learning. Therefore, we used data augmentation to alleviate the learning problem caused by unbalanced data. Two opposing data augmentation methods were presented when enhancing model robustness using data augmentation. The robustness and performance of the model were improved by directly transforming the data to augment the data and by augmenting it using a model trained in the entire data. Unlike the combinatorial method [1] or the resampling method that creates a new flow by combining packets extracted from existing flows in the NTC field, new data was created and augmented. Two contrasting data augmentation methods showed different effects on data across traffic types and application types.

The paper is organized as follows. Section II contains related work which is used to solve the problems of encrypted network traffic classification. Section III describes the performed work. Section IV describes the baseline of classification models. Section V describes the obtained results and analyzes them. Section VI provides a discussion. Section VII contains conclusions.

II. REALTED WORKS

As mentioned before, payload encryption of network traffic rendered existing traffic classification and traffic identification methods, including port-based methods and DPI techniques, stale and unusable. Port-based methods classify traffic by utilizing the correspondence between specific ports and protocols, but it became ineffective due to the widespread utilization of proxy port forwarding and dynamic port usage. And DPI became faded techniques to analyze entire IP packets into constituent units to distinguish traffic data. Since signature extraction and further analysis had been limited in the payload of encrypted packets, DPI was also unusable for traffic classification. As encryption of payloads prevails for security purposes, the subsequent contents have included encrypted payloads as a baseline.

The following parts sequentially describe the existing works. Machine learning and deep learning models for network traffic classification are presented sequentially in conjunction with input data. There are studies related to insufficient data for learning with class imbalance and various methodological studies to improve the learning performance of the model.

A. INPUT DATA AND MODELS FOR NTC

Network traffic data that has been preprocessed according to various research perspectives (Raw data or selective parts of it, statistical features, cropped parts of intact packets reshaped into arbitrary image sizes, natural language processing of each byte, etc.) can be input to adequate machine learning or deep learning models. Machine learning models have an inherent limitation in that the amount of input data is limited to produce valid results through model learning. The input data satisfying the limitation of machine learning models have been raw byte values on a single packet or selective statistical values based on the flow [17], [41]. The representative feature selection algorithms are forward selection, backward elimination, and stepwise [42]. However, researchers in flow statistics mostly define empirical feature sets and use those values as inputs to models. Therefore, while most machine learning models focus on the statistical value of flow, deep learning models selectively focus on the whole first packet or flow [17], [43].

It is common to include deep learning models as part of machine learning models in a taxonomic sense. However, this paper focuses on the distinction between tree-based machine learning models, which do not have the structure of a neural network, and deep learning. As aforementioned, there are more notable differences in the amount of input available for each model. Therefore, it will be defined as a separate object in this paper. Deep learning models also accept tabular inputs, and one of their main advantages is that they can be structured to handle multimodal inputs. These multimodal inputs can include images, text, audio, or other types of data [32]. On the other hand, machine learning models based on tree-based methods, compared to deep learning models, are limited to processing tabular data with a limited number of columns and cannot easily incorporate other types of data.

1) MACHINE LEARNING-BASED METHODS

The Authors in [44] presented the results of a study that applied unsupervised learning based on K-means using statistical properties that they judged to be relevant in the flow (e.g., number of packets, duration, packet arrival time deviation, length of the first ten packets, etc.). Their clustering model achieved only 86% accuracy using statistical properties. Furthermore, the following research results [45], [46] should not be underestimated, which show that NTC based on flow statistics also has problems. Flow statistics-based features can vary considerably depending on the structural characteristics of the group-level network. In the case of bidirectional Internet traffic, the tolerance of the characteristics is not general. It can vary over time, as the usage characteristics of the terminal users are variable and dependent on unstable temporary events.

However, there is notable research that attempts a new approach to flow data, moving beyond the statistical properties of flow data [47]. The authors in [48] studied a high-performance online classification method using zero-length packets for TCP data. Based on their findings, zero-length packets have no payload and contain only TCP control information. They constructed information about application-specific fingerprint sequences in application protocol data units based on zero-length packets and classified the traffic using the specific decision tree algorithm. Consequently, they achieved 97% accuracy. Their research, based on application-specific fingerprint sequences, is noteworthy because it can be used to find essential classification grounds even from packets without payloads.

These research results indicate that results can vary greatly depending on the perspective of the researchers who obtain the features input to the machine learning model.

2) DEEP LEARNING-BASED METHODS

As mentioned in the previous part, depending on the direction in which the researcher needs to deal with the data, the data is preprocessed in a form suitable for the input data, and then input to the selected model (i.e., machine learning model, CNN model for image [49], Long Short-Term Memory (LSTM) model time series data [50], etc.). General classification problems of other domains indicate that while domain-specific (i.e., image or language understanding) solution perspectives are important, the NTC domain allows for more flexibility in approach.

The authors in [24] used a convolutional layer to obtain the inherent field characteristics of each packet from a spatial structure and an LSTM layer to treat the previously extracted features from a time series perspective. Therefore, we found that they did not deal with single-packet data from a time series perspective. The authors in [24] simultaneously designed a deep learning model that used spatial construction of packet bytes and time series characteristics of flow using a combination of CNN and RNN. Their focus was on each packet and flow. They used CNN to render network traffic data as an associated image. They designed the structure of the CNN-LSTM connection model by converting packets into images, extracting structural features through CNN, and then connecting the extracted information to an LSTM network so that the time series characteristics of the flow are simultaneously reflected in the model. With the CNN-LSTM linked model, they achieved 96.32% accuracy.

Another noteworthy research implemented each classifier based on three models and compared their performances. The three models are MLP, Stacked AutoEncoder (SAE), and CNN [51]. The authors focused on raw packets. It was not designated whether the raw packet was the first. To use them as inputs for each model, they preprocessed the size of the raw packet of each application to 1,480 bytes. Packets shorter than 1,480 bytes were padded with '0 × 00', and packets longer than that were cut off. Although all models performed well, it was unavoidable that the best-performing SAE model was extremely large, consisting of over 1.3M parameters.

The authors in [40] proposed ET-BERT, which uses the Masked BURST Model and the Same-origin BURST Prediction structure, referring to the structure of BERT, which is based on the Transformer layer. ET-BERT presented results on packets and flow, respectively. This model was pre-trained with many unlabeled data and fine-tuned with a few task-specific labeled data. The F1 values of the service and application classification results, performed on the single packet basis of the ISCX VPN-nonVPN 2016 dataset (ISCX-VPN2016) [52], reached 98.90% and 99.37%, respectively. However, the ET-BERT model only focused on packet-level features and did not consider byte-level features of encrypted traffic, so it could not escape the disadvantage that it needed to perform the pre-training process of a general large language model and had an enormous computational and resource burden.

Given the excessive number of parameters in deep learning models, the size of stored models, and network device resource constraints, the continuous need to extract significantly relevant features and reduce model complexity to make deep learning-based network solutions suitable for production-grade has been raised [53]. In [9], we adapted DistilBERT, which uses the natural language processing mechanism, to the NTC domain. We used the above-compressed model to classify using reduced features, which consist of lightweight 63-byte packets with headers and partial payloads. We compared the results of the first packet with those of the flows. Furthermore, we applied Comparative Learning to process four multi-attribute packets simultaneously by removing feature redundancy. The F1 score of the service and application classification results, performed on the packet basis of the ISCX-VPN2016 dataset, reached 97.0% and 98.1%, respectively. The trained model classified 128 packets per second. Through model and feature reduction, we provided a starting point for proposing a feasible solution even under resource constraints. The reason for using only 63 bytes as input is that the BERT-based natural

language processing model has one of the highest classification performances using Comparative Learning, which was experimentally shown in our previous study [9]. Further description as to the choice of 63bytes was presented in Sections III-B and IV-A. Based on the above research results, we anticipated that a packet-based approach using a standardized tabular format with only 63 bytes of raw data could provide sufficient performance results even when using machine learning models [32]. So, we have designed a comparison of the results of injecting just 63 bytes of raw data into various tree-based machine learning models suitable for tabular data with the performance of the previous lightweight deep learning models.

In another context, although outside the NTC domain, research that comprehensively compares deep learning and machine learning is noteworthy. The authors in [54] revealed that while deep learning models have achieved tremendous performance on text and image datasets, their performance superiority on tabular data is unclear. They provided a notable reference on tree-based models such as XGBoost, Random Forests, and several deep learning models through many datasets and hyperparameter combination trials. Results showed that tree-based models outperform deep learning models on medium-sized data (around 10,000 samples), aside from the measured superior speed.

The above research result may imply that if the performance of deep learning and machine learning models is almost the same when only tabular data is used as input, pacesetting research can be needed to improve the performance and robustness of deep learning models. This may represent a significant and noteworthy turning point in various research fields. Hence, to persist and extend their considerable achievement in [54], empirical investigations on the different inductive biases through comparison of tree-based machine learning and deep learning models can also be needed in various fields, including the NTC domain. In this paper, we have attempted to approach that subject empirically from the above perspective.

B. INSUFFICIENT AND IMBALANCED DATA

Although not in the NTC domain, there are studies on the effectiveness of data augmentation for insufficient data in the computer vision domain. The authors in [36] demonstrated remarkable performance in traffic sign image classification by leveraging data augmentation in the image domain. Given the limited size of the available training dataset, they utilized more than 20 advanced multi-technique data augmentation methods to improve the accuracy of traffic sign image classification. This multi-technique data augmentation significantly improved the performance of deep learning models by applying controlled transformations to the original dataset, increasing the training data's diversity and robustness. The results of a comparative analysis with the model learned from the original data set were presented. They showed that multi-technique data augmentation improved the accuracy of traffic sign image classification by 2.3%, revealing that data augmentation was effective in improving the model's performance. Their data augmentation methods, which use various technologies, have provided notable implications for a methodologically flexible approach to model improvement in the NTC field.

The researchers tried data augmentation in the NTC domain as a new approach [1]. They created each new flow by selectively extracting individual packets from existing flows of the same class and then combining them into a flow. They assumed that the new flow would behave similarly to the samples used to combine the flows so that they could be labeled accordingly. By transforming combinations of existing data, they changed each flow enough to be considered new data but still retained the same class. Their approach differs from ours in that it generates flows by modifying existing packets directly or through learned models, as their approach is to create new combinations based on existing data.

The authors in [55] analyzed the problem of learning from imbalanced data (i.e., the imbalanced learning problem) from various theoretical perspectives. The imbalanced learning problem has been suggested to be related to the performance of learning algorithms in situations where data is scarce and there is severe hierarchical distributional skew. Therefore, overcoming the imbalanced learning problem suggested the need for new understanding, algorithms, and tools to transform large amounts of raw data into efficient and actionable information and knowledge representations. In their work, representative methods such as sampling methods, cost-sensitive learning methods [56], and others were presented methodologically. Since the sampling method maintains the original data's uniqueness, unlike data augmentation, it only calibrates the imbalance of the data. Cost-sensitive learning methods are functional adjustment techniques in the learning process of the model and are not related to quantitative changes in the data.

Care must be taken when adjusting the data distribution by sampling method, and there are cases where the distribution has been incorrectly adjusted. For example, considering the imbalance of class data in ISCX-VPN2016, authors in [51] adjusted the quantity through random sampling to fit each class with the same amount. The researchers in the paper presented that the model's performance was lower because the model did not learn as well as the original data, even though the adjustment was made for the minority class. The distribution characteristics were too different from those of the original data set. In [57], researchers have experimented with the same quantitative configuration to diminish learning bias in collecting data for each class. As in the previous case, it cannot be ruled out that the model did not learn properly for the major class.

The experimental method [58] that brought about positive effects of oversampling for minority classes differed from the uniform distribution [59]. A notable aspect of their work was that they applied the resampling method while maintaining the imbalanced data distribution ratio and presented meaningful results. Experimental results show that there are

two ways to adjust the ratio of training data to alleviate data distortion. The two methods were uniform sampling, which ideally samples each class in equal amounts, and hierarchical sampling, which samples each class to maintain the same distribution as the original data set. The latter showed better classification performance than the former. The original data ratio determined the distribution characteristics of each data set. The changed proportion of classes with less data through resampling became an essential element in learning. Changes in the data distribution from which the model learns also affected the model's learning results.

The authors in [38] presented empirical results that invalidate the implicit assumption of many studies on classifier research. That false implicit assumption is that the class distribution of the training data should match the original ''natural'' distribution. So, they changed the class distribution of the training set so that the distribution between the minority and majority classes changed. As a result, their study showed that the naturally occurring original class distribution is often not the best fit for learning and that using a different class distribution, which is a state where the distribution of a particular class is not extreme, can lead to much better performance. However, a caveat did not apply when the class distribution of the altered training set deviated significantly from the original class distribution. Contrariwise, they also showed that for natural distributions with extremely small minority class distributions, the modified distributions perform slightly better than the original distribution. One important thing to note is that the performance is good when the training data does not deviate too much from the original natural distribution and at the same time is similar. Based on the research results of [38], we applied data augmentation by selecting a minority class to make the class distribution natural, but we were careful that the performance may not improve if the class distribution of the altered training set deviates significantly from the original natural class distribution.

C. ROBUST LEARNING

Research on increasing model robustness should begin with a comprehensive perspective that the overall direction of model learning comes from data. Among related studies, four representative ones are multi-modal learning, multi-task learning, out-of-distribution sample, and few-shot learning.

1) MULTI-MODAL LEARNING

Researchers have systematically studied the relationship between deep learning and multi-modal learning in [60] and [61]. With the remarkable success of deep learning in the computer vision domain, the direction of deep learning research is expanding to highlight the advantages related to more complex multimodal data. These multimodal data sets are composed of data from various perspectives observing a single phenomenon. Multimodal data complement other data at the model training step to support effectively learn from complex tasks. It is well presented in their works that the more significant advantage of deep learning is that it can automatically learn through hierarchical representation for each modality by manually designing modality-specific features.

Authors in [62] had analyzed the flow features of encrypted traffic (e.g., length sequences, message types, statistical features, etc.). Although they could achieve good results in some cases, they concluded that these handcraft-extracted features could not apply to each different task individually, and important information was lost, which could affect the classification accuracy. To address these issues, they designed a multimodal-based deep learning framework, which used raw bytes and length sequences as multimodal inputs. They used a self-attention mechanism in the model to learn deep relationships between network packets in bi-flows and applied unsupervised pre-learning in the learning process. The effectiveness of their multi-modal framework was demonstrated in 19 applications, showing an accuracy of 99.22%. It was remarkable that robust learning of the model was achieved based on the multimodality of the inputs forwarding to the model.

2) MULTI-TASK LEARNING

A primary motivation in the introductive stages of multi-task learning [63] was to alleviate the problem of having limited data for each task. In the computer vision and natural language processing domains, the powerful resources provided by ''big data'' have shown that deep multi-task learning models can outperform single-task models. The reason why multi-task learning is effective is because it utilizes more data across different learning tasks. This contrasts with single-task models. Multi-task learning becomes more robust as it uses more data across multiple tasks. Since it can learn universal representations, it improves knowledge sharing across functions, improving each task's performance and resulting in less overfitting on each task.

There is a paper [29] that presents the benefits of extending from a single-task baseline to a multi-task, multi-modal architecture in the NTC domain. The authors proposed a multi-modal, multi-task deep learning approach. By leveraging data heterogeneity, they presented results that surpass the performance limitations of conventional single-task-based classification. At the same time, it solved various traffic classification problems with different modalities. The results showed a 5-8% performance improvement over single-task, single-modality approaches. This indicates that multi-task learning and multi-modal learning act as levers to increase the learning robustness of the model within an independent model. The previous study [64] also found that multi-task learning significantly impacts deep learning models. However, the magnitude of the change is different compared to the multi-modal, multi-task deep learning model in [29].

3) OUT-OF-DISTRIBUTION SAMPLE

The uncertainty of a learning model on out-of-distribution samples occurs both when the model is uncertain about which

class to choose in label assignment and when the data does not belong to a pre-learned class. Thus the classification can be uncertain [11]. Experimentally, the above studies have shown that the performance of models trained on general data distribution can be improved by designing them to undergo an additional step to learn quickly on out-of-distribution samples. That is, they performed out-of-distribution sample detection based on score evaluation, and then retrained the model to improve the robustness of the model. While they collected out-of-distribution samples through time-consuming manual work, we applied data augmentation to solve the related problem in about an hour.

4) FEW-SHOT LEARNING

The ability of an AI model to generalize characteristics of data by learning from a small amount of data is known as Few-Shot learning [65], [66], [67]. A generative data augmentation method based on Few-Shot learning theory, which can effectively capture and reproduce synthetic data with similar characteristics through models trained on natural data.

TABLE 1. Data type by task.

Task	Classes
Task 1 (6 Traffic Type)	VoIP, Email, Chat, Streaming, FTP, P2P
Task 2 (15 Application)	Skype, Spotify, Vimeo, VoipBuster, YouTube, AIM Chat, Email, Facebook, FTPS Gmail, Hangouts, ICQ, Netflix, SCP, SFTP

Few-Shot learning and data augmentation stand for background and implementation methods, respectively, and can be seen as starting with the same motivation other than a theoretical role difference. Research results have had significance in practicality and suggested substantial advantages in two aspects. First, collecting thousands of labeled data to achieve reasonable performance on a new task is unnecessary. Easing the data collection effort and reducing the computational cost and time spent on preprocessing can ease the effort. Second, in many domains, data collection is often difficult or impossible for reasons such as privacy and security. This may be an alternative to these cases.

III. METHODOLOGY

In the NTC domain, determining the data units to be input into a classification model can be important, as it involves finding a form that fits the characteristics inherent in the classification model. Session data, which consists of all bidirectional transmission and reception packets, and flow data, which means only unidirectional packets, do not target single packets. Both types are sequential arrangements of packets rather than individual features, so they can have the characteristics of time-series data.

In other words, a single packet is a collection of values that occurred at the same time and does not have the properties of time-series data. We reflected the research results [40], [68], [69] that experimentally showed that the entire features or subsets (i.e., partial packets) of a single packet that occurred simultaneously in this time unit are also suitable for traffic classification. The above partial packets have been transformed into standardized tabular data, allowing for the simultaneous application of various classification models, such as machine learning or deep learning models. Based on the previous perspective, we built our research and applied it to an efficient machine learning model suitable for tabular format data. We also compared and analyzed it with a lightweight deep learning model regarding performance and efficiency. Furthermore, we have extended it by applying data augmentation techniques to increase the learning robustness of the model.

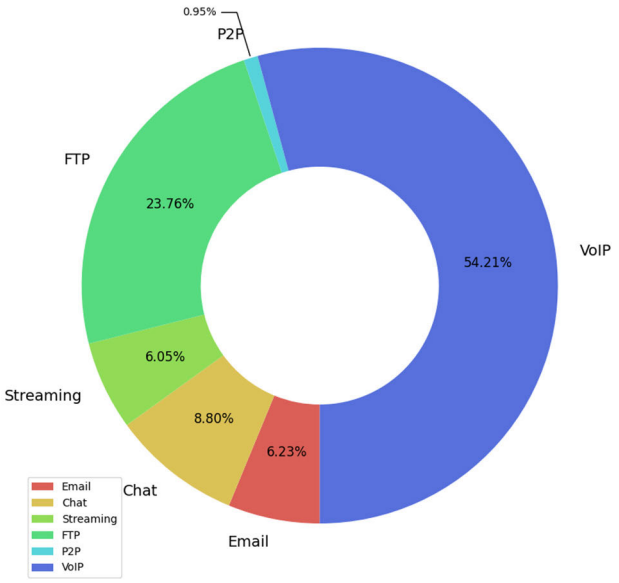

FIGURE 1. Donut chart showing the distribution of network traffic type data (task 1).

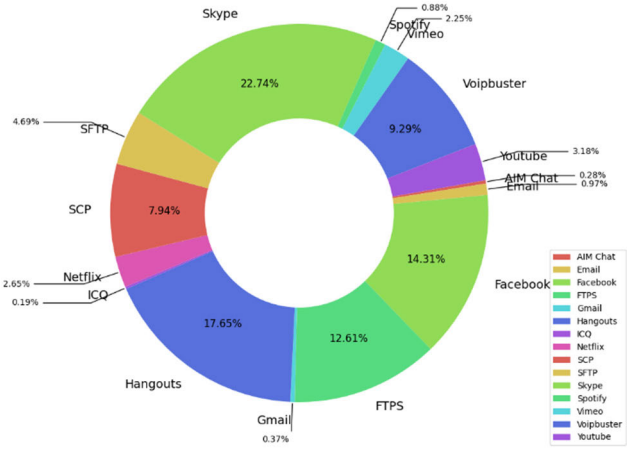

FIGURE 2. Donut chart showing the distribution of internet application data (task 2).

A. DATASET

The ISCX VPN-nonVPN 2016 dataset (ISCX-VPN2016) [52] is an open dataset of the University of New Brunswick and is widely used by other researchers in the NTC field [11], [20], [21], [40], [41], [56], [69].

We also choose the ISCX-VPN2016 dataset for the openness of our implementation approach to our research process. The open ISCX-VPN2016 dataset contains both general traffic and VPN traffic. There are six types of network traffic collected from various applications. They can be categorized as chat, email, file transfer, P2P, streaming, and VoIP. There are 15 types of internet applications, such as Facebook and YouTube. The original ISCX-VPN2016 dataset is seriously unbalanced. It reflects the characteristics of actual data as they are, ensuring an open research scope. This paper aims to study one of the meaningful solutions for utilizing data in this imbalance. Details of the ISCX-VPN2016 dataset are available in Table 1 and Table 6. The two types of task-specific data obtained through the preprocessing step in Table 1 consist of 27,814 and 134,563.

The class distribution of the two kinds of task-specific data is shown in Figures 1 and 2. In the following work, all processes were performed after maintaining the split ratio of training and test data as 80:20 for independent distribution of augmentation data. In the following Section V-D, these two types of training data have been separated from test data and augmented.

In [70], they also mentioned the class imbalance problem. The class imbalance problem of a data set is an ongoing issue and a significant problem for the training of classification models. The mistake of ignoring or misclassifying a minority sample while focusing only on the classification of the primary sample can be made unconsciously. In this work, we applied two different data augmentation methods with different methodological frameworks from the perspective of data augmentation, with a focus on increasing the minority class. How we dealt with it is described in Section IV-D.

B. DATA REPRESENTATION

1) PREPROCESSING FOR STANDARDIZED FORMAT

We considered it desirable and adapted the method used by the authors of Deep Packet [21] to reduce the influence of the length of individual packets and preprocess packets into a structured form, such as tabular format data. This standardization locates the related values of their attributes in its own fixed place. The Ethernet header had been removed. The UDP header padded the insufficient with zeros to make the UDP header 20 bytes long, the same size as the TCP header. IP addresses were masked in the IP header, as specific IPs can add bias to model training. It removed irrelevant packets, such as packets with no payload or DNS packets.

Then, the raw packets were converted into byte vectors. To prevent the learning bias of the model due to packet size, the excess parts of byte vectors larger than 1,500 bytes were truncated, and the missing parts of byte vectors smaller than 1,500 bytes were padded with zeros. The byte vectors were normalized by dividing each byte by 255. Therefore, each byte value has been normalized to be between 0 and 255.

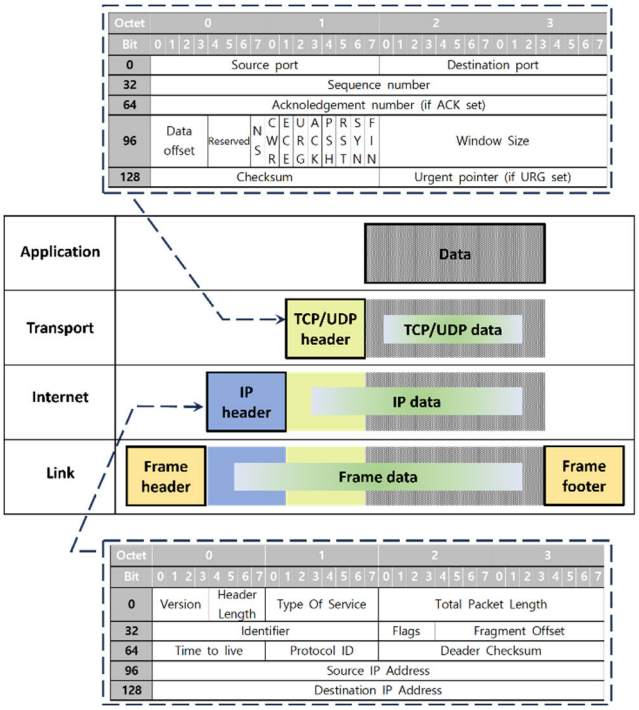

FIGURE 3. The hierarchical structures and the detailed header according to the TCP/IP protocol.

2) UNIFYING THE EXTENT OF INPUT FOR ALL MODELS

In the NTC domain, since the Transformer layer-based BERT series deep learning models can achieve optimal performance when using Comparative Learning [9]. Comparative learning is a structure that trains 4 packets, each consisting of 128 bytes, simultaneously. The structural design that applies to the Natural Language Processing (NLP) model within the input range of the BERT series, which is generally limited to 512 bytes, is described in Section IV-A. Section IV-A describes the characterization and Comparative Learning of the ALBERT model in the natural language processing domain to apply NTC for packet classification. This characterized structure can be suitable for performance comparison in terms of experimental results, as it can be applied to all BERT series models. According to the above, we used 63 bytes of input for all comparison models, excluding 65 bytes of structural data out of 128 bytes that are used to adapt the NLP model. Hereafter, the 63 bytes, the partial packet, are the first 63 bytes of the first packet of the flow. Therefore, the partial packet consists of the IP header (20 bytes), the TCP/UDP header (20 bytes), and the partial of the encrypted payload (23 bytes), as shown in Figure 3. This partial packet is the basic unit for data augmentation in our study and is the basic structure of the data input to all classification models.

It was thoroughly considered that there would not be a problem with the sensitivity difference when using input

data of more than 63 bytes in the classification model. The experimental basis revealed is described in the Discussion section of this paper for machine learning models, and it was confirmed in our previous research [9] for NLP-based deep learning models.

C. TWO CONTRASTING DATA AUGMENTATION

In terms of data augmentation methodology, flipping, resizing, cropping, rotating, etc., can be used for image data. can be used. However, designing similar rules for data in other domains relies heavily on domain knowledge [71] and requires a lot of validation effort. Furthermore, augmentation rules are specific to a particular dataset and may be difficult to apply to other datasets. Consequentially, it is realistically very difficult for humans to design and implement all possible changes.

Because there have not been many approaches to data augmentation in the NTC domain, there has not been much development in terms of taxonomy. Therefore, among various methods in the Taxonomy of the time series domain [37], the following two were selected. We designed two opposite augmentation methods to ensure that the classifier is suitable for directly using the 63-byte tabular format data generated through augmentation. The former is based on the random transformation of each data, which can be called the method of specific sample-driven augmentation. The latter is based on learning of generative models for overall data, which can be called the method of entire sample-driven augmentation.

Figure 4 shows the overall flow of improving the robustness of the model through data augmentation. The training data and test data were separated before the data augmentation process. After that, we chose one of the two afore-mentioned methods as the data augmentation method. The combined data consisted of training data and augmented data generated based on the original training data. That is, after acquiring augmented data using a generator from training data, combined data can be obtained by assembling the augmented data with the original training data. The empirical verification process should be repeated for the augmented data by comparing the performance of the combined data with the original training data. When performing the data augmentation step in the data augmentation process, we considered that the model performance could vary significantly depending on the data augmentation degree of the minority class, as mentioned in Section II-B. So, to prevent the changes in the minority class from being too large, we designed it so that a moderate amount of augmentation is performed mainly in the lower minority class.

The overall adjusted ratios through augmentation are detailed in Table 6.

1) ADDING RANDOM NOISE: JITTERING

Specific Sample-Driven Augmentation (SSDA) has been materialized by adding random noise. Jittering was chosen for that. Jittering, a classic but very simple and effective method of transformation-based data augmentation [72], [73], was applied to increase the robustness of individual data values by varying the amount of variation in the range of each value. p^1 denotes the first single packet which is extracted from interactive session data. v_1 means the actual value of packet data in bytes.

$$p^1 = (v_1 + \varepsilon_1) + \ldots + (v_n + \varepsilon_n) \tag{1}$$

In Equation (1), ε_n is the noise vector is designed to be optionally added. The optional ε_n can be zero, or a specific value that is obtained probabilistically. The vertical selection range of values for the optional vector, ε_n, is a random variable based on the mean and standard deviation, which are

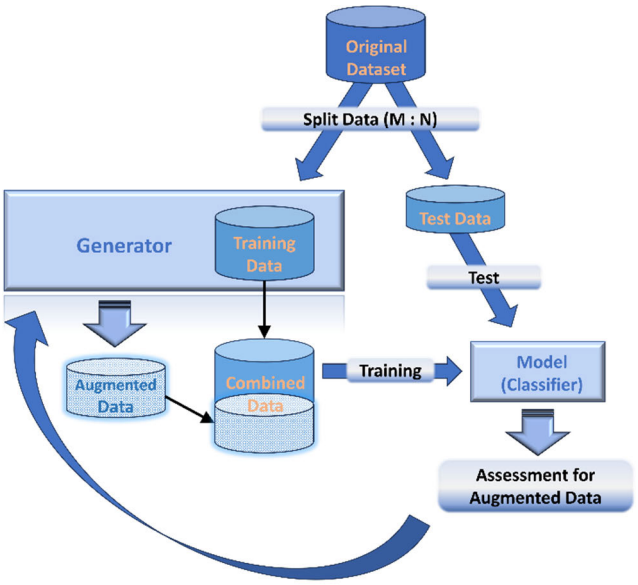

FIGURE 4. Diagram showing the cyclical structure of original data partitioning, augmentation, model learning, and assessment to increase model robustness by utilizing data augmentation.

Algorithm 1 Jittering

Input: First_63_Bytes, MAX_LEN = 63, NOISE_RATIO = 0.1
Output: First_63_Bytes added with noises
1: **function**Values_diff(array)
2: Calculate the i-th discrete difference # out[i] = a[i] - a[i-1]
3: Calculates the mean μ and standard deviation σ of array
4: **return** elements of array within the range $\mu - \sigma \leq$ array $\leq \mu + \sigma$
5: **end function**
6: DIF $\leftarrow$ Values_diff(First_63_Bytes)
7: $\mu_{dif} \leftarrow$ Compute the mean μ of DIF
8: $\sigma_{dif} \leftarrow$ Compute the standard deviation σ of DIF
9: POS $\leftarrow$ Select a random subset of indices, POS, of size MAX_LEN $\times$ NOISE_RATIO
10: Generate random noise from a normal distribution with mean μ_{dif} and standard deviation σ_{dif}, and assign noise to the positions indicated by POS.
11: Inject the generated noise into First_63_Bytes at the indices specified by POS.
12: **return** The modified First_63_Bytes # selectively added noises to each byte

the basic statistics of the range of variation in the selected data. In addition, the horizontal selection range for each byte is such that only a small fraction of all bytes is randomly selected. As the NOISE ratio increases, the proportion of important features that the model should focus on changes significantly, which can have a negative impact on learning, so we set it to 10%. We used the first 63 bytes from the packet as input to the jittering module to add noise directly to the data. Afterward, the 63 bytes of generated data are then retained for use as a tabular format data type in the classifier. At this time, the label of the generated synthetic data was specified as the label of the original data entered the jittering module.

2) UTILZING GENERATIVE MODEL: CGAN

Entire Sample-Driven Augmentation (ESDA) has been materialized by the generative model. Conditional Generative adversarial Network (CGAN) was chosen for that. CGAN is a GAN model that allows researchers to produce data with the desired labels by adding condition variables during the GAN training process [74]. It is a unique structure in which two 'adversarial' separate models are configured to learn together. These two consist of a generative model G that captures the properties of the original data, learns them, and then creates data, and a model D that discriminates whether the samples created by model G are real or synthetic data.

Since it has been trained under a condition, denoted here as y, it is possible to obtain the desired data conditional on y. Therefore, we used label as the y value identically for making synthetic data by each label thereafter.

$$\min_G \max_D V(D, G) = \mathbb{E}_{x\sim\mathcal{P}_{data}(x)}[logD(x|y)] + \mathbb{E}_{z\sim\mathcal{P}_z(z)}[log\,(1\text{-}D\,(G\,(z\,|\,y))] \quad (2)$$

In Equation (2), $\mathbb{E}$ means the general characteristics of stochastic as a distribution of small values. $\mathcal{P}_{data}(x)$ denotes the distribution of training data and $\mathcal{P}_z(z)$ denotes the randomly added noise distribution. The generator learns $\mathcal{P}_{data}(x)$ and builds its function based on the combination of noise distribution and the input conditions, denoted as y.

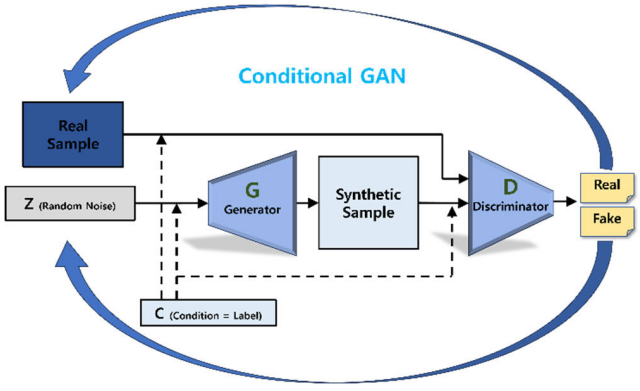

FIGURE 5. CGAN, which represents a Generator and Discriminator that perform competitive circular learning depending on conditions.

TABLE 2. Network architecture and hyper parameters of CGAN.

	Layer Type	Shape	Activation
Generator	Linear1	(NOISE + Condition, 256)	Relu
	Linear2	(256, 512)	Relu
	Linear3	(512, 1024)	Relu
	Linear4	(1024, Partial Packet)	Relu
	Tanh	-	-
Discriminator	Linear1	(Partial Packet + Condition, 1024)	LeakyRelu
	Linear2	(1024, 512)	LeakyRelu
	Linear3	(512, 256)	LeakyRelu
	Linear4	(256, 1)	LeakyRelu
	Sigmoid	-	-

The discriminator builds a probability function that represents $\mathcal{P}_{data}(x)$ as a single scalar value. In this case, x also should be the data with the combined condition. A value function V (G, D) can be thought of as a learning process that achieves the simultaneous goal of minimizing the value of G while maximizing the value of D. Figure 6 illustrates the general structure of CGAN, which uses labels as conditional values to generate data according to the labels. In terms of learning structure, this network is based on the basic structure of training the generator first and the discriminator afterward. To elaborate, the generator creates a synthetic sample that is input with a noise vector and a condition vector. By distinguishing between this synthetic sample and the original sample with the condition vector added, a structure is formed in which the discriminator learns while the generator also learns at the same time. During a unit epoch, the generator and the discriminator learn simultaneously for all data through this competitive structure.

The two samples that form the basis of learning are fake samples generated by the generator and real samples with a condition vector added. To keep the size of the NOISE vector smaller than the size of the partial packets used as input to train the generator model, which is 63 bytes, we set the NOISE vector to a much smaller size of 6 bytes. When specifying the NOISE vector, it should be considered equivalent to the NOISE ratio of jittering algorithm.

TABLE 3. Size comparison of bert series models.

	BERT	DistilBERT	ALBERT
Size of Model (bytes)	440 M	267 M	49 M
Parameter of Model (quantity)	109 M	66 M	11 M

As the proportion of input noise increases, the proportion of important features that the model should focus on changes significantly, which can have a negative impact on learning. The condition value input to the CGAN is the same as the label value so that the generation model can generate the packet data with the required label later. So here we have

applied 6 types of labels for Task 1 and 15 types of labels for Task 2 as the condition. As revealed in Table 2, we kept the neural network structure of the generator and discriminator simple, with four linear layers symmetrically.

D. PREPARING EXPERIMENTS

Since the two data augmentation methods we chose are different, the preparation process is different to enable the data augmentation to be evaluated recursively according to Figure 5. In the case of CGAN, after the training is completed, the packet generation model G is saved and used as the generation network of the data augmentation module in Section V. We can simply generate synthetic packets according to the labels of the classes we need to generate. On the other hand, Jittering needs to create a synthetic packet by inserting random noise based on the training data, so we divided the training data by class and configured it beforehand for use in the Jittering module.

IV. BASELINE OF CLASSIFICATION MODELS

This paper also places intermediate significance on comparing the dominance of computational performance metrics between machine learning models, including deep learning models. Therefore, the following describes the theoretical characteristics of baseline models used in this work. It includes three BERT-based deep learning models adapted to NTC and three machine models such as Random Forest, Extra Tree, and LightGBM.

A. DEEP LEARNING MODELS FOR CLASSIFICATION

1) THREE BERT SERIES MODELS

In a DistillBERT, which was a Transformer-based model lightened by knowledge distillation [75]. However, the teacher model itself, which consists of many parameters, cannot be excluded from the learning process due to the learning structure between the large teacher model and the student model, so the learning process cannot be diminished anymore. Therefore, the ALBERT [76] model based on the Cross-layer Parameter Sharing Structure of Universal Transformer [77], which was started from the perspective of moving away from the extensive parameters of the teacher model of such knowledge distillation, is much smaller than the parameter size of the model based on distillation that we studied before, about one-fifth of the size. The output of each Transformer layer is fed back into its input, called Cross-layer Parameter Sharing. It uses a recursively Transformer layer. ALBERT is a structure that shares all layer parameters, so the number of parameters does not increase even if the number of layers increases by sharing all parameters between layers, such as the Attention layer, feed-forward network layer, etc.

Furthermore, after using Universal Transformer's Cross Layer parameter sharing to eliminate the redundancy of the same layer in the model, ALBERT applied the Factorized Embedding Parameterization technique to reduce the layers' size again. In ALBERT, the authors factorized the embedding parameter into two smaller matrices.

In Equation (3), instead of projecting the one-hot vector V (Vocabulary size) directly to the H (Hidden state dimension) vector, they projected it through the low-dimensional E (word Embedding dimension) vector, which is the embedding size.

$$O(V \times H) \Rightarrow O(V \times E + E \times H) \tag{3}$$

In Table 3, DistilBERT [78] reduces the model and parameter size by half compared to the original BERT by using knowledge distillation. Using the above two significant weight reduction methods, the ALBERT model is about one-fifth smaller in model and parameter size than DistilBERT.

2) ADAPTATION TO NTC DOMAIN

To adapt the ALBERT model based on the natural language processing mechanism to the NTC domain, we have introduced packet-specialized tokenization and Comparative Learning from our previous work [9] in the front and back stages of the model, as shown in Figure 1.

As a first step for packet-specialized tokenization, each byte of the packet corresponds to a word in the natural language processing domain, so we put a 'space' between each byte value for separation to apply the embedding while maintaining the value of the individual byte. The next step was to use the subword tokenizer of BERT for tokenizing, inserting special tokens CLS and SEP, which typically separate the start and midpoint of the input data. Furthermore, we also inserted the STARTER, a special token designed to separate each packet as a preliminary operation that considers Comparative Learning.

After packet-specialized tokenization for the first 63 bytes of the packet, the 128-byte size is composed of 3 bytes of three special tokens STARTER, CLS, and SEP, 63 bytes of the input value, and 62 bytes converted to the 'comma' string values between the input values. Comparative learning is a structure that trains 4 packets, each consisting of 128 bytes, simultaneously.

The structural design applies to the Natural Language Processing (NLP) model within the input range of the BERT series, which is generally limited to 512 bytes. The four 128-byte represented packets were concatenated, simultaneously embedded, and passed through the ALBERT model, and the resulting values were trained and classified using Comparative Learning at once.

The above three Transformer-based deep learning models, which are overtly provided online, were used to be directly fine-tuned through transfer learning. After that, the three-step structure of packet-specialized tokenization, each model's internal processing (BERT, DistilBERT, ALBERT), and Comparative Learning were designed to be identically applied to the above three models. An experimental comparison of learning performance for models reflecting the same mechanisms we designed is presented in Section V.

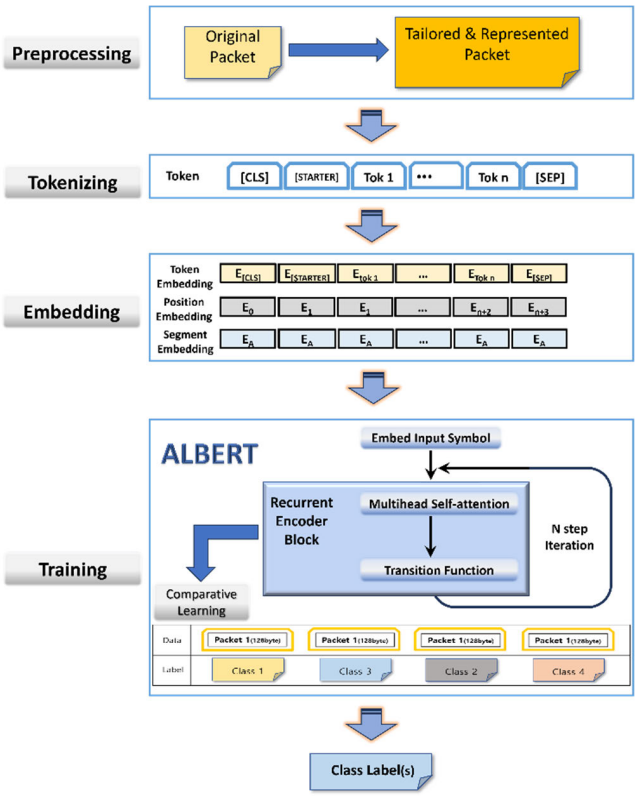

FIGURE 6. An example of ALBERT, a structure that adapts the NLP mechanism of the BERT series model to the NTC domain. The sequential three-stage structure (packet-specialized tokenization, model internal processing, and comparative learning).

B. MACHINE LEARINIG MODELS FOR CLASSIFICATION

1) RANDOM FOREST

Random Forest [79], as the name suggests, is a method that extracts results by ensemble and prevents the creation of duplicate decision trees through 'random' as much as possible.$D_i = \{e_1, e_2, \cdots, e_{i-1}, e_i$ means the data have i number of elements. The e_i denotes the i-th element of data. When the data has n finite elements, bootstrap is executed to create new datasets, D_n, by selecting e_i. At that time, e_i is extracted as often as n, with replacement sampling (allowing for duplication). When D_n are generated using n number of Datasets respectively, DT_n denotes n number of decision trees. The DT_n has a n-type of shape depending on the different n number of data sets. The n predicted values are aggregated through DT_n, and classification is performed through voting. To sum up, an important part of RF is randomly extracting e_i through restoration sampling to create new datasets, D_n. Through this process, Random Forest has reduced bias and variance.

2) EXTRA TREE

Extremely Randomized Trees (Extra Tree) [80] are characterized by adding randomness when splitting. To explain in comparison, the Random Forest finds the best part among the randomly selected features and continuously splits them. When focusing on features, while Random Forest only considers randomly selected features, Extra Tree added randomness when splitting. As an ensuing process, finding the point with the highest information gain and splitting is the same as Random Forest. Reflecting the random characteristic in feature selection, it reduces bias by configuring the tree shape differently and forming an ensemble. It also reduces variation by randomly selecting features and splitting them.

3) LightGBM

LightGBM [81] has achieved the effect of reducing the size of the dataset and the effect of feature engineering through new algorithms called Gradient-based One-Side Sampling and Exclusive Feature Bundling and has greatly increased the execution speed. As the name suggests, Gradient Boosting Decision Tree utilizes Gradient Boosting in the process of creating a Decision Tree. LightGBM is one of several solutions proposed to efficiently construct a Gradient Boosting Decision Tree.

The boosting technique is a method that intensively learns the parts that were not learned well in the previous stage at each stage. In the process of increasing the branches of the tree, the gradient is calculated, and the leaves are split at the part where the loss function can be reduced the most. In addition, at each stage, a tree is created that intensively learns data that was not learned well in the previous stage. The part that takes the most time and resources in the process of creating a Gradient Boosting Decision Tree is the part of finding the split point. This is because all values of the corresponding feature must be sorted, and which part of all possible split points best classifies the data.

Since checking all the split points in this way is inefficient, to efficiently construct Gradient Boosting Decision Tree, LightGBM uses two algorithms named Gradient-based One-Side Sampling and Exclusive Feature Bundling. Gradient-based One-Side Sampling is an algorithm that reduces the number of instances, which are samples in the dataset. The basic assumption of Gradient-based One-Side Sampling is that the larger the gradient of an instance, the greater the influence and the less learned the instance is. A large gradient of an instance means that changes in the values (elements) that make up the instance can bring about a large change in the loss function.

They consider such instances as instances that the model has not learned enough, and therefore we reduced the number of samples in the dataset by including such instances. When learning a tree, it calculates all the information gains corresponding to each split point and splits the point with the largest information gain. They proved that there is an upper bound on the difference in information gain between the reduced dataset and the original dataset.

Exclusive Feature Bundling is an algorithm that reduces the number of features in the dataset. Exclusive Feature Bundling is a method to reduce the number of features by grouping mutually exclusive variables into a single bucket

under the assumption that high-dimensional data is sparse. Here, "mutually exclusive variables" refer to variables that do not have non-zero values at the same time. Since there are few completely mutually exclusive variables, a condition that allows a certain amount of conflict (the number of instances that are not mutually exclusive) was added. This is called random polluting, and they proved that there is an upper bound on the difference in results when random polluting is applied, so that even if a certain level of conflict is allowed, there is no significant degradation in performance.

V. EXPERIMENT RESULTS AND ANALYSIS

A. RESOURCES AND METRICS

The implementation details have been described in Table 4. BERT, DistilBERT, and ALBERT classification models are based on the Transformer layers. Including the above deep learning model, the MLP-based CGAN, used for data augmentation, utilized GPUs. The three tree-based classification models (LightGBM, Random Forest, and Extra Tree) and the Jittering module, the generative model used for data augmentation, utilized the CPU. At the beginning of the experiment, we used the Pycaret library to perform basic performance tests on the machine learning models.

After that, we experimented with the Scikit-learn library to tune the hyperparameters of each machine learning model to elevate their performance.

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

$$Recall = \frac{TP}{(TP + FN)} \tag{5}$$

$$Precision = \frac{TP}{(TP + FP)} \tag{6}$$

$$F1\ score = \frac{(2 \times Recall \times Precision)}{(Recall + Precision)} \tag{7}$$

TABLE 4. Configuration of experimental resources.

Hardware / Software		Description
Hardware	CPU	12 Core AMD Ryzen 9 5900X @ 3.70 GHz
	GPU	NVIDIA GeForce RTX 3080 Ti (12GB Memory)
Software	CUDA	11.6
	Python	3.7.13
	Pytorch	1.12.1
	Pycaret	3.1.0
	Scikit-learn	1.3.0

In Equation (4) ∼ (7), 'T' stands for true, 'F' stands for false, 'P' stands for positive, and 'N' stands for negative, i.e., 'TP' and 'TN' stand for the part of the prediction that matches the actual value, and 'FP' and 'FN' stand for the part of the prediction that differs from the actual value. TP represents the amount of data where the actual value is in the positive class and the predicted value is in the positive class. FP represents the amount of data where the actual value is in the negative class and the predicted value is in the positive class. FN represents the amount of data where the actual value is in the positive class and the predicted value is in the negative class. TN represents the amount of data where the actual value is in the negative class, and the predicted value is in the negative class. Accuracy is the percentage of total values that the model correctly classified. Recall is the proportion of things the model predicted to be true out of actually true things. Precision is the percentage of what the model classifies as True that is actually True. The F1 score is the harmonic mean of precision and recall.

TABLE 5. Performance and TestTime comparison of each model for task 1.

Model (Training Time)	Performance (%)				Test Time
	Accuracy	Precision	Recall	F1-score	Packet / s
BERT (224min)	97.77	97.96	97.96	97.96	98
DistilBERT (97min)	97.72	98.30	97.09	98.10	106
ALBERT (210min)	97.24	98.04	97.56	97.80	129
LightGBM (9.75sec)	**97.72**	**97.72**	**97.72**	**97.72**	**23,672**
Random Forest (24.0sec)	97.41	97.02	97.41	97.40	8,241
Extra Tree (17.7sec)	97.27	97.28	97.27	97.26	6,236

B. PERFORMANCE COMPARISON BETWEEN MODELS

As shown in Table 5, the deep learning model's learning performance was reflected by the number of epochs until the best performance was achieved. In the case of BERT, since it took 449 seconds for 1 epoch, the total training time was 3 hours and 44 minutes by reflecting the time of 30 epochs.

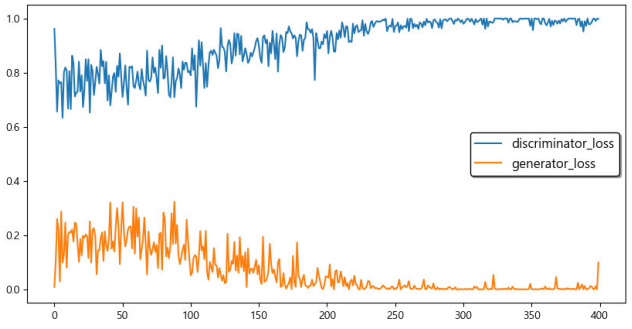

FIGURE 7. Graphs showing the changes in the loss values of the generator and discriminator during competitive circular learning over 400 epochs.

DistilBERT took 290 seconds for 1 epoch, so the total training time was 1 hour and 37 minutes by reflecting 20 epochs. ALBERT had taken 269 seconds for 1 epoch, so the total

TABLE 6. Configuration of original training data and augmented data.

	Data	Original Training Data	Augmented Data	Combined Data
Task 1 (6 Traffic Type)	**Total**	22,248	12,746	34,994
	VoIP	1,386	4,158	5,544
	Email	1,957	3,914	5,871
	Chat	1,346	4,038	5,384
	Streaming	5,287	0	5,287
	FTP	212	636	848
	P2P	12,060	0	12,060
Task 2 (15 Application)	**Total**	107,650	38,265	145,915
	Skype	305	5,185	5,490
	Spotify	1,048	5,240	6,288
	Vimeo	15,401	0	15,401
	VoipBuster	13,570	0	13,570
	YouTube	403	5,239	5,642
	AIM Chat	19,002	0	19,002
	Email	209	5,225	5,434
	Facebook	2,853	0	2,853
	FTPS	8,546	0	8,546
	Gmail	5,046	0	5,046
	Hangouts	24,480	0	24,480
	ICQ	950	5,700	6,650
	Netflix	2,425	4,850	7,275
	SCP	9,999	0	9,999
	SFTP	3,413	6,826	10,239

training time was 3 hours and 30 minutes by reflecting 47 epochs. On the other hand, LightGBM, Random Forest, and Extra Tree only took 9.75 seconds, 24.0 seconds, and 17.7 seconds for the whole training process, respectively. Among the above machine learning models, LightGBM, which revealed one of the highest accuracies, is slightly lower than BERT by about 0.05%. Still, the number of classifiable packets per second is 23,672, which is a significant number that can overcome the performance disadvantage. Furthermore, while BERT using GPU can classify 98 packets per second, LightGBM can be evaluated as significantly efficient in that it shows a speed performance 230 times better than BERT while only using CPU resources.

Considering efficiency based on hardware resources and packet classification rate per second as the classification model, LightGBM revealed significant performances, as shown in Table 5. So, hereafter, we have conducted experiments and analyses on augmented data using only LightGBM, which has demonstrated superior performance in terms of efficient time utilization.

C. VISUALIZATION OF CGAN LEARNING PROCESS

In terms of the model's learning structure, the learning losses of the generator and the discriminator of the CGAN model have been revealed to be symmetrical graphs, as shown in Figure 7. As described in Section III, the discriminator continuously learns a series of two samples within a batch, which are composed of a fake sample created by the generator with a noise vector and a condition vector as input, and a real sample with a condition vector, based on the batch size unit.

The generator and discriminator were trained in over 400 iterations, building gradual stability. Over the epoch, while the loss of the generator has approached zero, the loss of the discriminator has approached 1. It can be evaluated that the generator and discriminator successfully completed symmetric learning as planned through the learning process.

D. MODEL ROUBUSTNESS USING AUGMENTED DATA

The original training data of 22,248 and 107,650 for each task in Table 6 correspond to 80% of the 27,814 and 134,563 obtained in the previous preprocessing phase. Based on the original training data, the Jittering and CGAN were applied to proceed with the augmentation of tabular format data [82], and SSDA data and ESDA data were obtained. As explained in Section III, the NOISE ratio is commonly applied to both data augmentation methods, and the ratio was kept at 10% of the data input. As the noise ratio increased, the range of variation in the augmented data and the number of empirical verifications increased.

However, the performance did not necessarily increase proportionally. Combined data was obtained by assembling the augmented data with the original training data.

In the data augmentation process, the class selection for augmentation and the amount of augmentation should be essential consideration factors affecting performance. In detail, we considered that when the class distribution of the training data deviates significantly from the original class distribution, the performance improvement through the class distribution change suggested in [38] could no longer be applicable. Therefore, we selected the number of classes and applied data augmentation not to deviate from the original class distribution. For example, when implementing data augmentation in Task 1, the FTP class was maintained so that its proportion did not deviate significantly from the original overall class distribution.

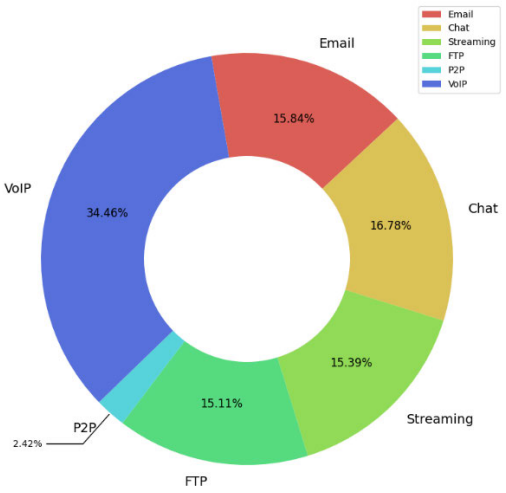

FIGURE 8. Donut chart showing the distribution of combined data for task 1 in Table 6.

TABLE 7. Performance comparison using LightGBM by 3 types of data and class weight.

Data			Class Weight	Performance			
				Accuracy	Precision	Recall	F1-score
Task 1 (6 Traffic Type)	Original Data		Class unweighted	97.72	97.72	97.72	97.72
			Class weighted	97.68	97.68	97.68	97.68
	Combined Data	**SSDA-based Data**	Class unweighted	**97.91 (+0.19)**	**97.91 (+0.19)**	**97.91 (+0.19)**	**97.91 (+0.19)**
		ESDA-based Data	Class unweighted	97.80	97.80	97.81	97.80
Task 2 (15 Application)	Original Data		Class unweighted	96.75	96.78	96.75	96.74
			Class weighted	96.84	96.87	96.84	96.83
	Combined Data	SSDA-based Data	Class unweighted	96.81	96.85	96.81	96.80
		ESDA-based Data	Class unweighted	**97.01 (+0.26)**	**97.05 (+0.27)**	**97.01 (+0.26)**	**97.01 (+0.27)**

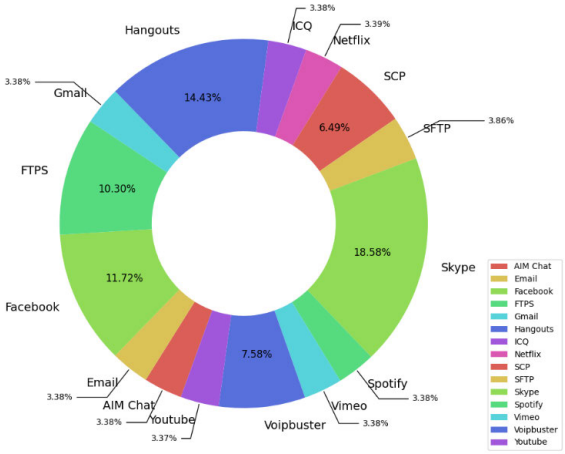

FIGURE 9. Donut chart showing the distribution of combined data for task 2 in Table 6.

The distribution of each data before and after data augmentation can be confirmed visually by Figures 1 and 2 of the previous Section III-A to Figures 8 and 9 of the current section. For reference, the augmented data was generated in approximately 30 minutes. The time it took for ESDA's generator model to train using CGAN was less than 30 minutes, and the total time to create the augmented data for Task 2 was 3.34 seconds and 6.79 seconds for SSDA and ESDA, respectively. Since data distribution adjustment through data augmentation is an empirical factor, comparing performance test results with different distributions can be an essential means to obtain better results.

As shown in Table 7, the three types of data are the original, SSDA, and ESDA data. It also contains results of measuring the performance of two types of tasks. Furthermore, when a model learns imbalanced data, it can have a negative impact on the model's performance due to biased learning. So, we added experimental results to simultaneously compare the performance of models learned by reflecting class weights.

In Task 1, the model trained with SSDA data revealed 0.19% higher results in all performance metrics than the model trained with the original data.

The model that reflected the class weights during the learning process showed 0.04% lower performance and 0.12% lower than the model trained with augmented data using ESDA. In Task 2, the model trained with ESDA data showed 0.26 ∼ 0.27 higher results in all performance metrics than the model trained with the original data. The model that reflected the class weights in the learning process showed a performance of about 0.09% higher than the model trained with the original data and 0.02∼0.03% higher than the model trained with the augmented data using SSDA. However, it showed a performance of 0.07∼0.08% lower than the data augmented with ESDA. The model's performance reflected class weight in the learning process showed lower performance than the superior model, which is among the models learned through data augmentation (either SSDA or ESDA).

Although not shown in Table 7, models trained by applying additional class weights to the combined data through data augmentation showed lower performance. Therefore, the performance of models using combined data in Table 7 did not reflect class weights.

The results of applying oversampling to each task for another contrast experiment to compare to class distribution adjustment are shown in Table 8. For appropriate augmentation for each task during oversampling, SSDA was applied to task 1, and ESDA was applied to task 2. Comparing Tables 7 and 8, the accuracy of the oversampling result was 0.9% lower than that of the result that maintained the appropriate ratio in Task 1 and 0.8% lower in Task 2. Therefore, it was confirmed that a better result can be obtained through an appropriate distribution that does not significantly deviate from the original data distribution than a distribution due to oversampling.

The data that constitutes the traffic type of Task 1 is a class that groups similar services of different applications into one kind and defines them with the same class label.

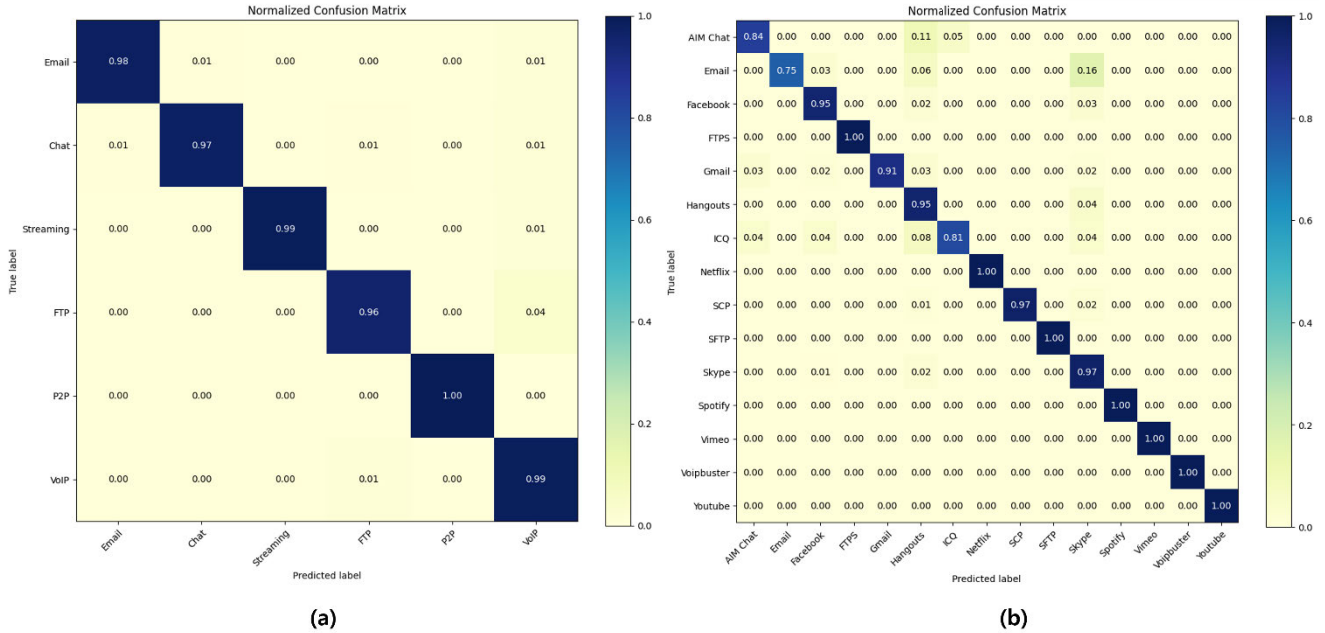

FIGURE 10. (a) Confusion matrix by applying the test data to the LightGBM model, which had been trained with the SSDA-based combined data for task 1. (b) Confusion matrix by applying the test data to the LightGBM model, which had been trained with the ESDA-based combined data for task 2.

TABLE 8. Performance comparison by oversampling using LightGBM.

Data		Performance			
		Accuracy	Precision	Recall	F1-score
Task 1 (6 Traffic Type)	Original Data	97.72	97.72	97.72	97.72
	Over-Sampling (by SSDA)	97.82 (+0.10)	97.82 (+0.10)	97.82 (+0.10)	97.82 (+0.10)
Task 2 (15 Applic ation)	Original Data	96.75	96.78	96.75	96.74
	Over-Sampling (by ESDA)	96.93 (+0.18)	96.98 (+0.20)	96.94 (+0.19)	96.93 (+0.19)

Synthetic data generated by SSDA can have variable ranges based on the properties of individual traffic data. Therefore, since the data generated by the SSDA method can reflect the characteristics of the original application, it can be analyzed that the properties of individual applications do not disappear because they do not reflect mechanisms directly related to the application categories generated in taxonomic terms.

However, synthetic data generated using ESDA has variable ranges within the group category where the generative model learns the characteristics of the unified group category type of each application traffic to determine the boundary. The reason for this may be that each class of the synthetic data of Task 1 generated through CGAN was learned by grouping them into a unified type according to the condition value (i.e., label) input when the generator model G of CGAN was trained. Due to the fundamental difference in how these two synthetic data are generated, SSDA seems to yield better results for task 1, data augmentation by traffic type. The unified network traffic type characteristics of the synthetic data generated by ESDA have been analyzed to fail to reproduce the characteristics of individual traffic. The visualization analysis of the synthetic data generated by SSDA and ESDA has been presented in Section V-E.

Figure 10 (a) demonstrates the best performance metrics of LightGBM trained on SSDA-based Combined data for Task 1.

Figure 10 (b) demonstrates the best performance metrics that LightGBM, trained on ESDA-based Combined data for Task 2. In Figure 10 (a) and (b), each value on the left diagonal presents the accuracy for each class as a confusion matrix. Other values on the parallel lines indicate misclassifications into different classes.

Figure 10 (b) A detailed analysis of the confusion metrics reveals that AIM Chat, Email, and ICQ have relatively low classification performance. The AIM Chat application was difficult to distinguish from the Hangout application, the Email application was difficult to differentiate from the Skype application, and the ICQ application was difficult to distinguish from the Hangout application. When analyzing the results of the confusion metrics from the perspective of investigating the cause, it appeared that the Skype application had a negative effect on the classification of Email applications. In addition, the Hangout application had a negative effect on the classification of ICQ and AIM Chat applications. These similar results were shown in other papers that tested the performance of classifiers made with CNN [21] and

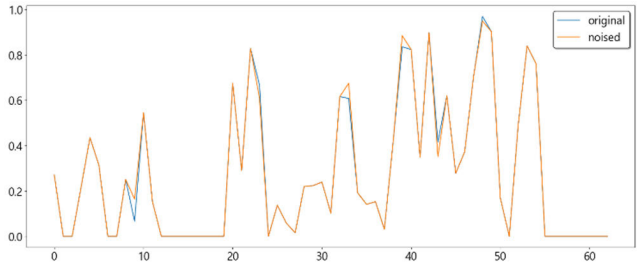

FIGURE 11. A graph comparing the feature phase changes of synthetic data obtained by applying SSDA using one sample packet.

attention mechanisms [27] targeting the ISCX-VPN2016 dataset, in that the three applications had relatively low classification performance metrics.

The classification result of the model trained on data that applied ESDA, which adds random noise to the first packet of 15 applications containing various images, audio, and text data, is shown in Figure 10 (b), and the result of applying SSDA is similar. These classification performance results of each application are identical to those derived from other researchers' papers mentioned above. Therefore, we can see that even when our proposed data augmentation is applied to the encrypted partial payload in 15 applications with various modalities, the characteristics of each data did not deviate significantly. The in-depth analysis of the above part is reflected in Section V-E. The extent to which the encrypted partial payload affects the model's classification results was reflected in Section VI, Discussion.

E. ANALYSING AUGMENTED DATA

To apply ESDA, the generator obtained by learning CGAN reflects the application type under the same conditions as the application label of the original data. Therefore, it was possible to analyze the similarity by visualizing the grouping of application distributions using UMAP. On the other hand, the synthetic data applied with SSDA is obtained by adding random noise to the individual feature values of the original packet according to Algorithm 1. Therefore, in addition to the distribution grouping analysis using UMAP, it was additionally possible to compare the similarity for individual data by analyzing the cosine similarity.

1) ANALYSING AUGMENTED DATA USING SSDA

Cosine similarity [83], [84] can be used to measure the similarity between two vectors in the inner product space. By measuring the cosine of the angle between the two vectors, we can determine approximately whether they are pointing in the same direction. In Equation (8), conceptually, $\|x\|$ is the length of the vector. It can be defined as $\sqrt{x_1^2+x_2^2+\cdots+x_n^2}$ and means the Euclidean norm of vector $x = (x_1, x_2, \cdots, x_n)$. Similarly, $\|y\|$ is the Euclidean norm of vector y. Because the *similarity*(x, y), the cosine of the angle between vectors x and y, can be obtained.

$$similarity\,(x, y) = \frac{x \cdot y}{\|x\| \cdot \|y\|} \tag{8}$$

So, the packets before and after the noise were added by the jittering Algorithm 1 were equally entered as x and y vectors, respectively.

The resulting values approximated the original data by about $99.70 \pm 0.05\%$. Furthermore, Figure 11 illustrates a graphical representation of phase changes that reveal how similar the synthetic data generated by each jittering is to the original partial packet data. As intended in Algorithm 1, it appears that limiting the range of noise distribution to the arithmetic deviation range within a feature unit of 63 bytes and limiting the number of features injected with noise to within 10% have been applied successfully.

2) ANALYSING AUGMENTED DATA USING ESDA

Uniform Manifold Approximation and Projection (UMAP) [85] is an algorithm that enables a unique implementation of the manifold learning technique for dimension reduction. It was created from a theoretical framework based on Riemannian geometry and algebraic topology. It also has practical characteristics in terms of usability because it can be quickly applied to actual data. When t-distributed Stochastic Neighbor Embedding (t-SNE) [86], another dimensionality reduction algorithm embeds dimensions, the waiting time can be a disadvantage due to the increased computational amount according to the dimension level. On the other hand, UMAP is a dimension reduction algorithm with significant usability for machine learning and deep learning data because it can be analyzed that there is no computational limitation due to the increased dimension. So, considering convenience, we used UMAP to visualize high-dimensional data.

Table 6 presents the original training data for each task, which was visualized in Figure 12 using UMAP.

Two types of synthetic data created through SSDA and ESDA with the same amount of original training data were visualized by applying them to UMAP in Figures 13 and 14, respectively. In all three Figures 12 $\sim$ 14, the left side is the visualized data for Task 1 and the right side is the visualized data for Task 2.

For Task 1, comparing Figures 12 (a) and 13 (a), the sporadic nature of the data tends to be alleviated in SSDA-based data compared to the original training data. Comparing Figures 13 (a) and 14 (a), the ESDA-based data has a distribution form with a single characteristic, the sporadic nature of the SSDA-based data by class has disappeared. In connection with the performance analysis in Table 7 above, since the data by each class in Task 1 are different applications but are bundles of the same type, it can be visually confirmed that the characteristics of other applications have disappeared through ESDA feature generalization. In terms of maintaining the original characteristics of data in a bundle of similar applications, data augmentation using jittering may be suitable for Task 1.

For Task 2, comparing Figures 12 (b) and 13 (b), the sporadic nature of the data has tended to be alleviated in SSDA-based data compared to the original training data.

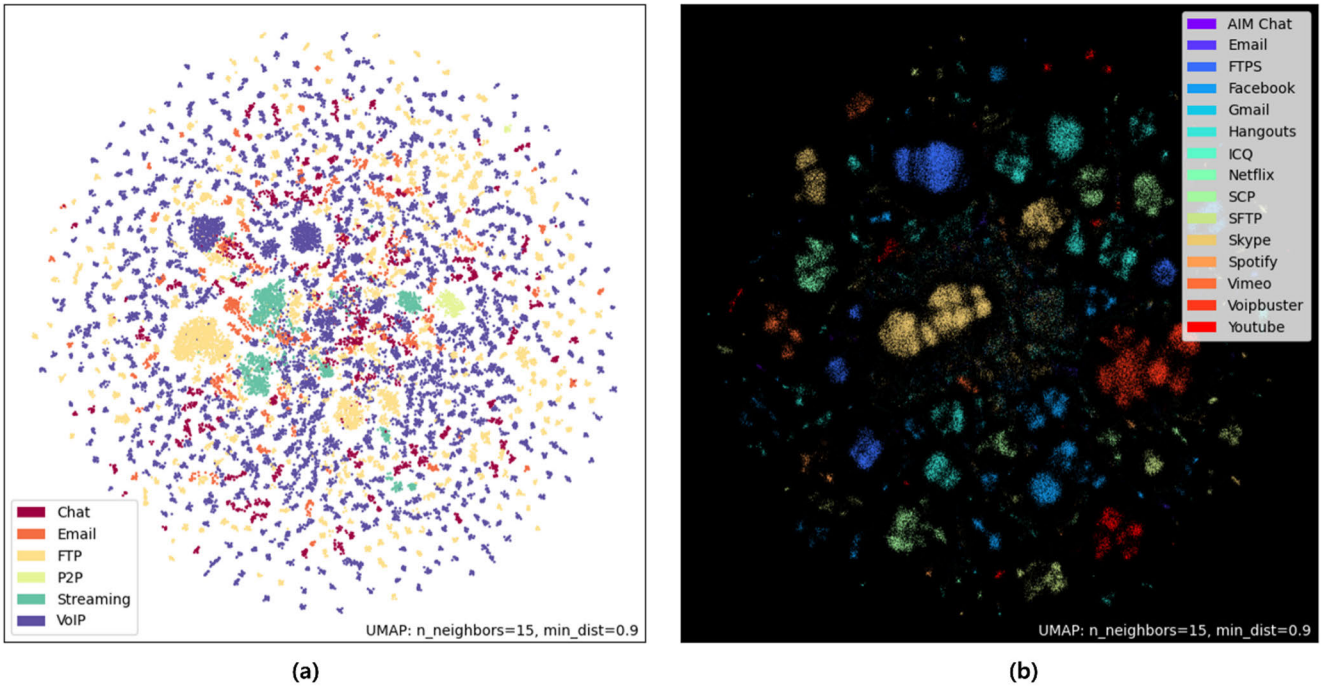

FIGURE 12. (a) The original data distribution of task 1 on a two-dimensional plane using UMAP. (b) The original data distribution of task 2 on a two-dimensional plane using UMAP.

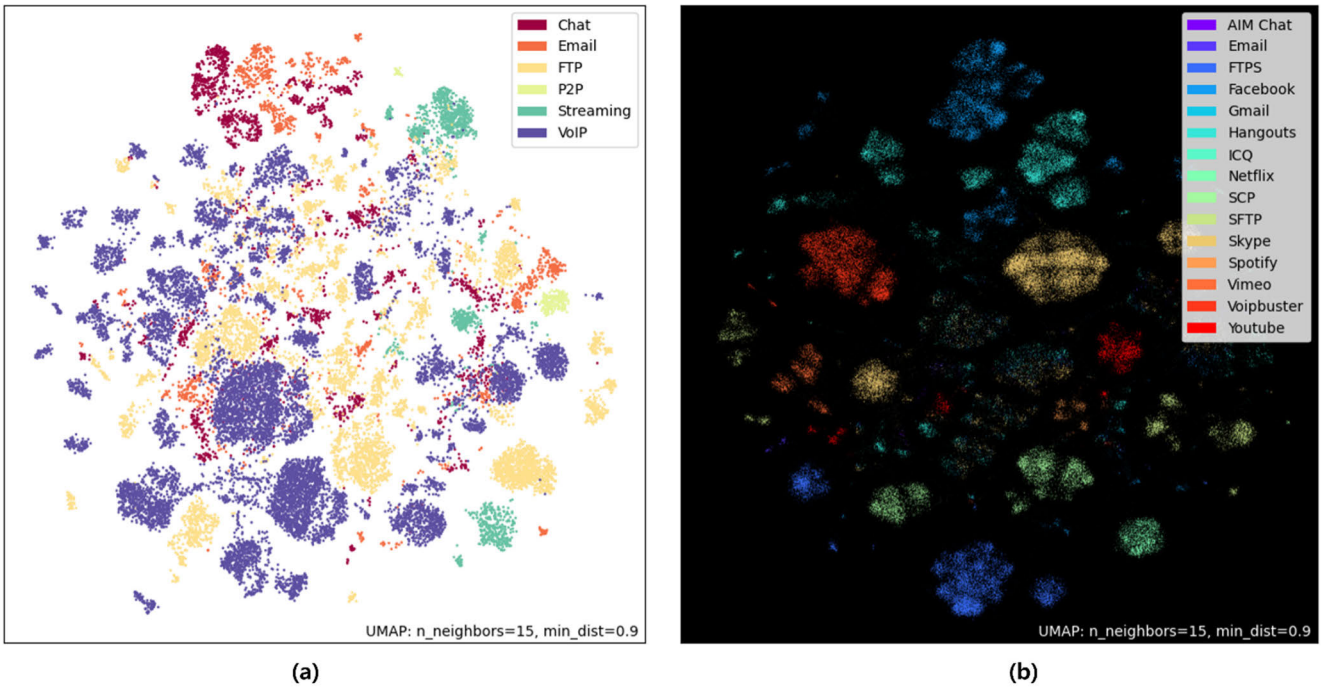

FIGURE 13. (a) The SSDA-based data distribution of task 1 on a two-dimensional plane using UMAP. (b) The SSDA-based data distribution of task 2 on a two-dimensional plane using UMAP.

Comparing Figures 13 (b) and 14 (b), the ESDA-based data showed that the sporadic nature of the data for each of the 15 classes has disappeared and that each of the 15 classes has its distribution pattern. In connection with the performance analysis in Table 7 above, it seems that the data for each class in Task 2 are different applications and that the characteristics

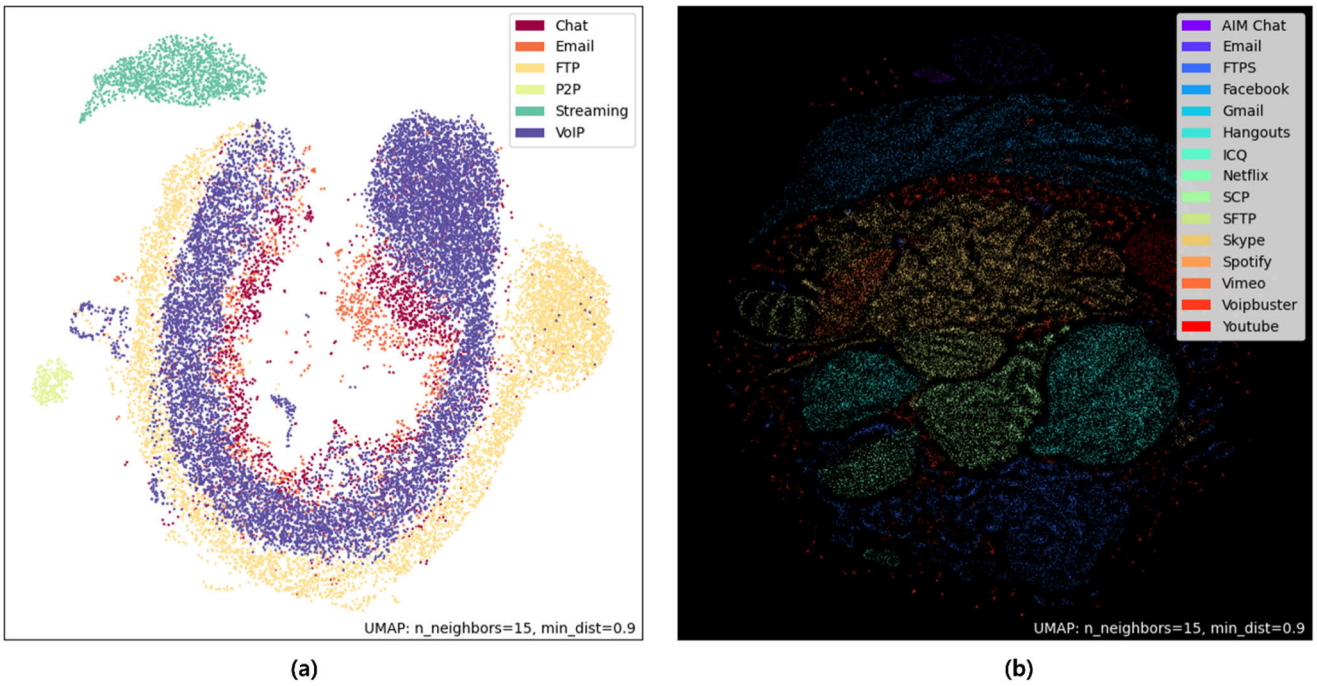

FIGURE 14. (a) The ESDA-based data distribution of task 1 on a two-dimensional plane using UMAP. (b) The ESDA-based data distribution of task 2 on a two-dimensional plane using UMAP.

of each of the 15 classes are effectively distinguished and extracted through ESDA. Applying data augmentation using a generative model that learns the characteristics of each application can be relatively suitable for Task 2.

After a numerical analysis of the model's experimental result, we extended this to visualize and analyze the data in a verifiable manner. By visualizing the data augmented in two opposing ways and the original data, we analyzed the correlation between the augmentation method and the model's performance. The above Correlation Analysis with Data Visualization provided a more reasonable justification.

VI. DISCUSSION

The approach to the NTC problem can be extended in various ways depending on the range of input data. In this paper, the basis of input to the model is packet headers and encrypted payloads. Also, as aforementioned, the number of features entering the deep learning model must be the same to be valuable in comparative experiments with machine learning, so the input was limited to 63 bytes of low packets. Figure 15 shows the accuracy variation according to the length of the encrypted payload used as input to various models. It is an Initial result obtained without manually adjusting hyperparameters using the Pycaret. 50 to 100 bytes, which correspond to a portion of the first packet, significantly impact performance. Inductive experiments that support meaningful and diverse results in the NTC domain based on more empirical factors are needed.

Because the packet header, which acts as structured data in raw packets, can vary depending on the researcher's

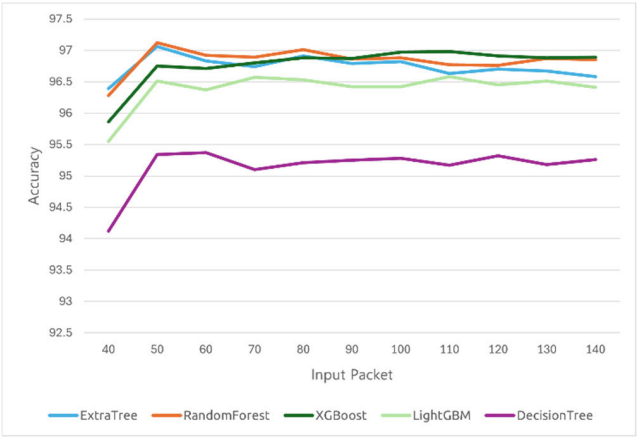

FIGURE 15. Accuracy variation according to the raw packet length.

perspective. It can be used as it is or differently through representation. Encrypted payloads may be less valuable than the representation effort due to being encrypted. In addition, the results can vary depending on the model's characteristics, and the scope of research can expand further when it can be included in the flow perspective and statistical features.

A. DATA AUGMENTATION AND OUT-OF-DISTRIBUTION SAMPLES

Table 7 also presents the performance comparison of class weights. The model's performance, which reflected the class weights during the learning process, was lower than the superior model among the models learned through data augmentation (SSDA or ESDA). As mentioned before, even

if the learning process includes augmented data, improving performance is difficult if the class distribution of the data is significantly different from the original class distribution [38]. Therefore, it means that we performed data augmentation to a degree that did not significantly deviate from the original data distribution so that the class distribution of the augmented data did not interfere with learning. Based on the details of our experiment, the distribution of the augmented data did not distort the distribution of the original data, as can be seen by comparing Figures 8 and 9 with Figures 1 and 2. Applying proportional class weights to handle imbalanced data may distort the original class distribution during the learning process, resulting in inferior performance. Therefore, data augmentation can be a distinguishing point because it allows for fine-tuning of the data distribution, unlike the cost-sensitive learning perspective [85], which uniformly reflects class weights during the model learning process.

From the perspective of increasing model robustness, researchers can also relate their approach to the problem of out-of-distribution samples [87] when trying to improve model learning through data augmentation. The most confusing part when trying to augment data is adjusting the distribution of each class from the perspective of the approach to the class imbalance problem. A raising adjustment also increases the range of random noise distributions in the augmented data, causing the model to overtrain beyond the appropriate out-of-distribution data. This adjustment may cause a similar effect to randomly deviating the training data from the original data distribution, which sometimes resulted in poor performance [51], and we experienced the same results in our experiments.

Researchers may overlook that the distribution of the test data, which has the same distribution range as the original data, has not changed. This oversight may lead researchers to augment the training data with more out-of-distribution samples than necessary. Research into the extent to which augmented data has appropriate out-of-distribution can be an ongoing subject of research expansion.

B. LIMITATION AND RANGE OF EXTENSION

Applying data augmentation to increase the robustness of the model through a new distribution of data is quite encouraging. Still, it also presents a new challenge to solve. Although machine learning models generally have the disadvantage that they cannot be trained jointly by augmenting features or connecting with deep learning blocks, the data can be evaluated within seconds when the above data augmentation is applied to machine learning models. Based on the details of our experiment, data augmentation applying ESDA used the deep learning mechanism to create synthetic data, but the classifier's performance improvement using the deep learning model was within 0.05%. In addition, deep learning models had a constant time delay corresponding to the training time. The above can be an empirical case corresponding to the research result [54] that neural networks have difficulty learning irregular patterns of target functions and that a rotational invariant learning procedure degrades performance. The same phenomenon can occur when deep learning models process more non-informative features in tabular format data. Further in-depth research is needed to approach data and models from a broader perspective.

VII. CONCLUSION

A. SUMMARY

In this paper, the packet header and some encrypted payloads (63 bytes) were converted into a standardized tabular form in the first step. Then, BERT and its lightweight models (ALBERT, DistilBERT) were adapted to the natural language processing mechanism in the NTC domain. And Comparative Learning was applied to test the performance of the deep learning models with the best performance to obtain excellent results. Next, the same input was used for tree-based machine learning models (Random Forest, Extra Tree, LightGBM) and compared with each other in terms of performance and effectiveness of the deep learning models. Although the LightGBM using the original data was about 0.05% less accurate than BERT, it was possible to classify 230 times more packets per second. The critical fundamental of the above is to convert the first packet into a standardized format, i.e., a tabular format, regardless of the packet type. It played an important role in classifying network traffic with a substantial performance using not a deep learning model but ''efficient'' machine learning.

To solve the data imbalance problem that can affect the training performance of these machine learning models, two opposing data augmentation methodologies (SSDA and ESDA) were applied to increase data production. Among them, the former applied jittering that randomly adds noise to individual data, and the latter applied a generative model of CGAN that learned the entire data and generated data based on labels. To positively utilize the factor that data from a minority class affects the learning of the model, it should not deviate significantly from the distribution level of the original data. So, we increased the training data empirically through a circulating evaluation process.

Since SSDA had created synthetic data based on individual data of each class, it was effective in augmenting the network traffic type data of Task 1. As a result, the performance increased by up to 0.19% compared to the performance of the model learned with the original training data. Since ESDA has created synthetic data based on the model that learned the feature distribution of each class as it is, it was effective in augmenting the application data of Task 2. As a result, the performance increased by up to 0.26% compared to the performance of the model learned with the original training data. These analysis results were visualized using UMAP, and a convincing verification of the analysis was possible.

B. FUTURE WORK

In the future, the research on how tree-based models outperform deep learning on tabular data in various domains [54]

seems to be extended to the NTC domain. Therefore, we will continue to conduct more inductive empirical studies and evaluations on deep learning using tabular data research in the NTC domain. Next, we will study how tree-based machine learning models can detect and internally interpret different biases in the same data compared to deep learning models due to differences in their internal algorithms. Therefore, we will construct related modules from a post-hoc explanation perspective or apply explanation mechanisms from other studies to provide a comprehensive understanding of the model and data from a user perspective. Finally, we will secure a more diverse open data set to test the general usability of the other method of network traffic classification and data augmentation we implement.

REFERENCES

[1] Y. Zion, P. Aharon, R. Dubin, A. Dvir, and C. Hajaj, "Enhancing encrypted internet traffic classification through advanced data augmentation techniques," 2024, *arXiv:2407.16539*.

[2] Y.-D. Lin, C.-N. Lu, Y.-C. Lai, W.-H. Peng, and P.-C. Lin, "Application classification using packet size distribution and port association," *J. Netw. Comput. Appl.*, vol. 32, no. 5, pp. 1023–1030, Sep. 2009, doi: 10.1016/j.jnca.2009.03.001.

[3] A. W. Moore and K. Papagiannaki, "Toward the accurate identification of network applications," in *Passive and Active Network Measurement* (Lecture Notes in Computer Science), vol. 3431, C. Dovrolis, Ed., Berlin, Germany: Springer, 2005, pp. 41–54, doi: 10.1007/978-3-540-31966-5_4.

[4] S. Fernandes, R. Antonello, T. Lacerda, A. Santos, D. Sadok, and T. Westholm, "Slimming down deep packet inspection systems," in *Proc. IEEE INFOCOM Workshops*, Apr. 2009, pp. 1–6, doi: 10.1109/INFCOMW.2009.5072188.

[5] A. Finamore, M. Mellia, M. Meo, and D. Rossi, "KISS: Stochastic packet inspection classifier for UDP traffic," *IEEE/ACM Trans. Netw.*, vol. 18, no. 5, pp. 1505–1515, Oct. 2010, doi: 10.1109/TNET.2010.2044046.

[6] S. Sen, O. Spatscheck, and D. Wang, "Accurate, scalable in-network identification of p2p traffic using application signatures," in *Proc. 13th Int. Conf. World Wide Web*. New York, NY, USA: Association for Computing Machinery, May 2004, pp. 512–521, doi: 10.1145/988672.988742.

[7] M. Abbasi, A. Shahraki, and A. Taherkordi, "Deep learning for network traffic monitoring and analysis (NTMA): A survey," *Comput. Commun.*, vol. 170, pp. 19–41, Mar. 2021, doi: 10.1016/j.comcom.2021.01.021.

[8] T. Shapira and Y. Shavitt, "FlowPic: Encrypted internet traffic classification is as easy as image recognition," in *Proc. IEEE INFOCOM Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Apr. 2019, pp. 680–687, doi: 10.1109/INFCOMW.2019.8845315.

[9] C.-Y. Shin, J.-T. Park, U.-J. Baek, and M.-S. Kim, "A feasible and explainable network traffic classifier utilizing DistilBERT," *IEEE Access*, vol. 11, pp. 70216–70237, 2023, doi: 10.1109/ACCESS.2023.3293105.

[10] P. Zhang, F. Chen, and H. Yue, "Detection and utilization of new-type encrypted network traffic in distributed scenarios," *Eng. Appl. Artif. Intell.*, vol. 127, Jan. 2024, Art. no. 107196, doi: 10.1016/j.engappai.2023.107196.

[11] S. Jorgensen, J. Holodnak, J. Dempsey, K. D. Souza, A. Raghunath, V. Rivet, N. DeMoes, A. Alejos, and A. Wollaber, "Extensible machine learning for encrypted network traffic application labeling via uncertainty quantification," *IEEE Trans. Artif. Intell.*, vol. 5, no. 1, pp. 420–433, Jan. 2024, doi: 10.1109/TAI.2023.3244168.

[12] V. Paxson, "Empirically derived analytic models of wide-area TCP connections," *IEEE/ACM Trans. Netw.*, vol. 2, no. 4, pp. 316–336, Aug. 1994, doi: 10.1109/90.330413.

[13] C. Dewes, A. Wichmann, and A. Feldmann, "An analysis of internet chat systems," in *Proc. ACM SIGCOMM Conf. Internet Meas.*, Miami Beach, FL, USA, 2003, p. 51, doi: 10.1145/948205.948214.

[14] T. Nguyen and G. Armitage, "Training on multiple sub-flows to optimise the use of machine learning classifiers in real-world IP networks," in *Proc. 31st IEEE Conf. Local Comput. Netw.*, Nov. 2006, pp. 369–376, doi: 10.1109/LCN.2006.322122.

[15] R. Alshammari and A. N. Zincir-Heywood, "Can encrypted traffic be identified without port numbers, IP addresses and payload inspection?" *Comput. Netw.*, vol. 55, no. 6, pp. 1326–1350, Apr. 2011, doi: 10.1016/j.comnet.2010.12.002.

[16] T. Auld, A. W. Moore, and S. F. Gull, "Bayesian neural networks for internet traffic classification," *IEEE Trans. Neural Netw.*, vol. 18, no. 1, pp. 223–239, Jan. 2007, doi: 10.1109/TNN.2006.883010.

[17] T. T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE Commun. Surveys Tuts.*, vol. 10, no. 4, pp. 56–76, 2008, doi: 10.1109/SURV.2008.080406.

[18] K. Shim, J. Ham, B. D. Sija, and M. Kim, "Application traffic classification using payload size sequence signature," *Int. J. Netw. Manage.*, vol. 27, no. 5, p. 1981, Sep. 2017, doi: 10.1002/nem.1981.

[19] A. Y. Nikravesh, S. A. Ajila, C.-H. Lung, and W. Ding, "Mobile network traffic prediction using MLP, MLPWD, and SVM," in *Proc. IEEE Int. Congr. Big Data*, Jun. 2016, pp. 402–409, doi: 10.1109/BIGDATACONGRESS.2016.63.

[20] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," in *Proc. IEEE Int. Conf. Intell. Secur. Informat. (ISI)*, Jul. 2017, pp. 43–48, doi: 10.1109/ISI.2017.8004872.

[21] M. Lotfollahi, M. J. Siavoshani, R. S. H. Zade, and M. Saberian, "Deep packet: A novel approach for encrypted traffic classification using deep learning," *Soft Comput.*, vol. 24, no. 3, pp. 1999–2012, Feb. 2020, doi: 10.1007/s00500-019-04030-2.

[22] V. Tong, H. A. Tran, S. Souihi, and A. Mellouk, "A novel QUIC traffic classifier based on convolutional neural networks," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Abu Dhabi, United Arab Emirates, Dec. 2018, pp. 1–6, doi: 10.1109/GLOCOM.2018.8647128.

[23] X. Ren, H. Gu, and W. Wei, "Tree-RNN: Tree structural recurrent neural network for network traffic classification," *Exp. Syst. Appl.*, vol. 167, Apr. 2021, Art. no. 114363, doi: 10.1016/j.eswa.2020.114363.

[24] 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, doi: 10.1109/ACCESS.2017.2747560.

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.* Red Hook, NY, USA: Curran Associates Inc., Dec. 2017, pp. 6000–6010.

[26] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," 2018, *arXiv:1810.04805*.

[27] W. Zheng, J. Zhong, Q. Zhang, and G. Zhao, "MTT: An efficient model for encrypted network traffic classification using multi-task transformer," *Appl. Intell.*, vol. 52, no. 9, pp. 10741–10756, Jan. 2022, doi: 10.1007/s10489-021-03032-8.

[28] G. Xie, Q. Li, Y. Jiang, T. Dai, G. Shen, R. Li, R. Sinnott, and S. Xia, "SAM: Self-attention based deep learning method for online traffic classification," in *Proc. Workshop Netw. Meets AI ML*, Aug. 2020, pp. 14–20, doi: 10.1145/3405671.3405811.

[29] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "DISTILLER: Encrypted traffic classification via multimodal multitask deep learning," *J. Netw. Comput. Appl.*, vols. 183–184, Jun. 2021, Art. no. 102985, doi: 10.1016/j.jnca.2021.102985.

[30] S. O. Arik and T. Pfister, "TabNet: Attentive interpretable tabular learning," 2019, *arXiv:1908.07442*.

[31] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*. New York, NY, USA: Association for Computing Machinery, Aug. 2016, pp. 785–794, doi: 10.1145/2939672.2939785.

[32] S. Onishi and S. Meguro, "Rethinking data augmentation for tabular data in deep learning," 2023, *arXiv:2305.10308*.

[33] Y. Zoabi, O. Kehat, D. Lahav, A. Weiss-Meilik, A. Adler, and N. Shomron, "Predicting bloodstream infection outcome using machine learning," *Sci. Rep.*, vol. 11, no. 1, p. 20101, Oct. 2021, doi: 10.1038/s41598-021-99105-2.

[34] P. Jurkiewicz, B. Kadziołka, M. Kantor, J. Domżał, and R. Wójcik, "Machine learning-based elephant flow classification on the first packet," *IEEE Access*, vol. 12, pp. 105744–105760, 2024, doi: 10.1109/ACCESS.2024.3436056.

[35] Q. Wen, L. Sun, F. Yang, X. Song, J. Gao, X. Wang, and H. Xu, "Time series data augmentation for deep learning: A survey," in *Proc. 13th Int. Joint Conf. Artif. Intell.*, Aug. 2021, pp. 4653–4660, doi: 10.24963/ijcai.2021/631.

[36] J. Dsouza, V. B. Shukla, V. Bhatia, and S. K. Pandey, "Enhancing classification of traffic sign using multi-technique data augmentation," in *Proc. IEEE 7th Conf. Inf. Commun. Technol. (CICT)*, Dec. 2023, pp. 1–8, doi: 10.1109/cict59886.2023.10455608.

[37] B. K. Iwana and S. Uchida, "An empirical survey of data augmentation for time series classification with neural networks," *PLoS ONE*, vol. 16, no. 7, Jul. 2021, Art. no. e0254841, doi: 10.1371/journal.pone.0254841.

[38] G. M. Weiss and F. Provost, "The effect of class distribution on classifier learning: An empirical study," Dept. Comput. Sci., Rutgers Univ., Tech. Rep. ML-TR-44, p. 26, Aug. 2001. [Online]. Available: https://scholarship.libraries.rutgers.edu/esploro/outputs/technicalDocumentation/The-effect-of-class-distribution-on/99103155024404646#details

[39] H. Y. He, Z. Guo Yang, and X. N. Chen, "PERT: Payload encoding representation from transformer for encrypted traffic classification," in *Proc. ITU Kaleidoscope, Industry-Driven Digit. Transformation (ITU K)*, Dec. 2020, pp. 1–8, doi: 10.23919/ITUK50268.2020.9303204.

[40] X. Lin, G. Xiong, G. Gou, Z. Li, J. Shi, and J. Yu, "ET-BERT: A contextualized datagram representation with pre-training transformers for encrypted traffic classification," 2022, *arXiv:2202.06335*.

[41] Y. Xu, J. Cao, K. Song, Q. Xiang, and G. Cheng, "FastTraffic: A lightweight method for encrypted traffic fast classification," *Comput. Netw.*, vol. 235, Nov. 2023, Art. no. 109965, doi: 10.1016/j.comnet.2023.109965.

[42] F. Sohil, M. U. Sohali, and J. Shabbir, "An introduction to statistical learning with applications in R: By Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani, New York, Springer Science and Bus. Media, 2013, $41.98,' *Stat. Theory Relat. Fields*, vol. 6, no. 1, p. 87, Jan. 2022, doi: 10.1080/24754269.2021.1980261.

[43] S. Rezaei and X. Liu, "Deep learning for encrypted traffic classification: An overview," *IEEE Commun. Mag.*, vol. 57, no. 5, pp. 76–81, May 2019, doi: 10.1109/MCOM.2019.1800819.

[44] A. Vlăduţu, D. Comăneci, and C. Dobre, "Internet traffic classification based on flows' statistical properties with machine learning," *Int. J. Netw. Manage.*, vol. 27, no. 3, p. 1929, May 2017, doi: 10.1002/nem.1929.

[45] A. C. Gilbert, W. Willinger, and A. Feldmann, "Scaling analysis of conservative cascades, with applications to network traffic," *IEEE Trans. Inf. Theory*, vol. 45, no. 3, pp. 971–991, Apr. 1999. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/761336

[46] R. H. Riedi and W. Willinger, "Toward an improved understanding of network traffic dynamics," in *Self-Similar Network Traffic and Performance Evaluation*, 1st ed., K. Park and W. Willinger, Eds., Hoboken, NJ, USA: Wiley, 2000, pp. 507–530, doi: 10.1002/047120644X.ch20.

[47] J. Zhao, X. Jing, Z. Yan, and W. Pedrycz, "Network traffic classification for data fusion: A survey," *Inf. Fusion*, vol. 72, pp. 22–47, Aug. 2021, doi: 10.1016/j.inffus.2021.02.009.

[48] J. Kampeas, A. Cohen, and O. Gurewitz, "Traffic classification based on zero-length packets," *IEEE Trans. Netw. Service Manage.*, vol. 15, no. 3, pp. 1049–1062, Sep. 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8335764

[49] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *J. Big Data*, vol. 6, no. 1, p. 60, Jul. 2019, doi: 10.1186/s40537-019-0197-0.

[50] J. Zhao and X. He, "NTAM-LSTM models of network traffic prediction," in *Proc. MATEC Web Conf.*, vol. 355, 2022, p. 02007, doi: 10.1051/matecconf/202235502007.

[51] P. Wang, F. Ye, X. Chen, and Y. Qian, "Datanet: Deep learning based encrypted network traffic classification in SDN home gateway," *IEEE Access*, vol. 6, pp. 55380–55391, 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8473682

[52] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of encrypted and VPN traffic using time-related features," in *Proc. 2nd Int. Conf. Inf. Syst. Security Privacy*, Rome, Italy, 2016, pp. 407–414, doi: 10.5220/0005740704070414.

[53] T. Zhang, H. Qiu, M. Mellia, Y. Li, H. Li, and K. Xu, "Interpreting AI for networking: Where we are and where we are going," *IEEE Commun. Mag.*, vol. 60, no. 2, pp. 25–31, Feb. 2022, doi: 10.1109/MCOM.001.2100736.

[54] L. Grinsztajn, E. Oyallon, and G. Varoquaux, "Why do tree-based models still outperform deep learning on tabular data?" 2022, *arXiv:2207.08815*.

[55] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 9, pp. 1263–1284, Sep. 2009, doi: 10.1109/TKDE.2008.239.

[56] H. Yao, C. Liu, P. Zhang, S. Wu, C. Jiang, and S. Yu, "Identification of encrypted traffic through attention mechanism based long short term memory," *IEEE Trans. Big Data*, vol. 8, no. 1, pp. 241–252, Feb. 2022, doi: 10.1109/TBDATA.2019.2940675.

[57] S. Ahn, J. Kim, S. Y. Park, and S. Cho, "Explaining deep learning-based traffic classification using a genetic algorithm," *IEEE Access*, vol. 9, pp. 4738–4751, 2021, doi: 10.1109/ACCESS.2020.3048348.

[58] W. Ruoyu, L. Zhen, and Z. Ling, "A new re-sampling method for network traffic classification using SML," in *Proc. 2nd Int. Conf. Inf. Sci. Eng.*, Dec. 2010, pp. 1735–1738, doi: 10.1109/ICISE.2010.5688893.

[59] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, Jun. 2002, doi: 10.1613/jair.953.

[60] D. Ramachandram and G. W. Taylor, "Deep multimodal learning: A survey on recent advances and trends," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 96–108, Nov. 2017, doi: 10.1109/MSP.2017.2738401.

[61] A. Nascita, A. Montieri, G. Aceto, D. Ciuonzo, V. Persico, and A. Pescapé, "XAI meets mobile traffic classification: Understanding and improving multimodal deep learning architectures," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 4, pp. 4225–4246, Dec. 2021, doi: 10.1109/TNSM.2021.3098157.

[62] P. Lin, K. Ye, Y. Hu, Y. Lin, and C.-Z. Xu, "A novel multimodal deep learning framework for encrypted traffic classification," *IEEE/ACM Trans. Netw.*, vol. 31, no. 3, pp. 1369–1384, Jun. 2023, doi: 10.1109/TNET.2022.3215507.

[63] Y. Zhang and Q. Yang, "A survey on multi-task learning," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 12, pp. 5586–5609, Dec. 2022, doi: 10.1109/TKDE.2021.3070203.

[64] J.-T. Park, C.-Y. Shin, U.-J. Baek, and M.-S. Kim, "Fast and accurate multi-task learning for encrypted network traffic classification," *Appl. Sci.*, vol. 14, no. 7, p. 3073, Apr. 2024, doi: 10.3390/app14073073.

[65] Y. Wang, Q. Yao, J. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," 2024, *arXiv:1904.05046*.

[66] A. Parnami and M. Lee, "Learning from few examples: A summary of approaches to few-shot learning," 2022, *arXiv:2203.04291*.

[67] Y.-X. Wang, R. Girshick, M. Hebert, and B. Hariharan, "Low-shot learning from imaginary data," 2018, *arXiv:1801.05401*.

[68] L. Bernaille, R. Teixeira, I. Akodkenou, A. Soule, and K. Salamatian, "Traffic classification on the fly," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 2, pp. 23–26, Apr. 2006, doi: 10.1145/1129582.1129589.

[69] G. Xie, Q. Li, and Y. Jiang, "Self-attentive deep learning method for online traffic classification and its interpretability," *Comput. Netw.*, vol. 196, Sep. 2021, Art. no. 108267, doi: 10.1016/j.comnet.2021.108267.

[70] R. Longadge and S. Dongre, "Class imbalance problem in data mining review," 2013, *arXiv:1305.1707*.

[71] B. Sathianarayanan, Y. C. Singh Samant, P. S. Conjeepuram Guruprasad, V. B. Hariharan, and N. D. Manickam, "Feature-based augmentation and classification for tabular data," *CAAI Trans. Intell. Technol.*, vol. 7, no. 3, pp. 481–491, Sep. 2022, doi: 10.1049/cit2.12123.

[72] T. T. Um, F. M. J. Pfister, D. Pichler, S. Endo, M. Lang, S. Hirche, U. Fietzek, and D. Kulić, "Data augmentation of wearable sensor data for Parkinson's disease monitoring using convolutional neural networks," in *Proc. 19th ACM Int. Conf. Multimodal Interact.*, Glasgow, U.K., Nov. 2017, pp. 216–220, doi: 10.1145/3136755.3136817.

[73] K. Rashid and J. Louis, "Time-warping: A time series data augmentation of IMU data for construction equipment activity identification," 2019, doi: 10.22260/ISARC2019/0087.

[74] M. Mirza and S. Osindero, "Conditional generative adversarial nets," 2014, *arXiv:1411.1784*.

[75] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*.

[76] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut, "ALBERT: A lite BERT for self-supervised learning of language representations," 2019, *arXiv:1909.11942*.

[77] M. Dehghani, S. Gouws, O. Vinyals, J. Uszkoreit, and Ł. Kaiser, "Universal transformers," 2018, *arXiv:1807.03819*.

[78] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter," 2019, *arXiv:1910.01108*.

[79] *Wayback Machine*. Accessed: Aug. 21, 2024. [Online]. Available: https://web.archive.org/web/20160417030218/http://ect.bell-labs.com/who/tkh/publications/papers/odt.pdf
[80] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Mach. Learn.*, vol. 63, no. 1, pp. 3–42, Apr. 2006, doi: 10.1007/s10994-006-6226-1.
[81] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T. Y. Liu, "LightGBM: A highly efficient gradient boosting decision tree," in *Proc. 31st Conf. Neural Inf. Process. Syst. (NIPS)*. Long Beach, CA, USA: Curran Associates, 2017. Accessed: Jan. 7, 2025. [Online]. Available: https://proceedings.neurips.cc/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html
[82] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, "Modeling tabular data using conditional GAN," 2019, *arXiv:1907.00503*.
[83] A. R. Lahitani, A. E. Permanasari, and N. A. Setiawan, "Cosine similarity to determine similarity measure: Study case in online essay assessment," in *Proc. 4th Int. Conf. Cyber IT Service Manage.*, Apr. 2016, pp. 1–6, doi: 10.1109/CITSM.2016.7577578.
[84] H. Steck, C. Ekanadham, and N. Kallus, "Is cosine-similarity of embeddings really about similarity?" 2024, *arXiv:2403.05440*.
[85] L. McInnes, J. Healy, and J. Melville, "UMAP: Uniform manifold approximation and projection for dimension reduction," 2020, *arXiv:1802.03426*.
[86] L. van der Maaten and G. Hinton, "Viualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.
[87] S. Liang, Y. Li, and R. Srikant, "Enhancing the reliability of out-of-distribution image detection in neural networks," 2017, *arXiv:1706.02690*.

CHANG-YUI SHIN received the B.S. degree in operating analysis from Korea Military Academy, Seoul, in 2003, and the M.S. degree in electronic computer engineering from Korea University, Seoul, in 2007. Since being commissioned as an Army Officer, in 2003, his service department has been Information Communication with Korean Army. At the time, he researched the field of mobile ad-hoc networks. After that, he became interested in practicalization while working in related diverse organizations such as weapon system planning, interoperability, development quality management, and actual operation. From 2022 to 2024, his doctoral course, he researched internet traffic classification, network management, and AI at the Laboratory of Korea University, Sejong Campus.

YANG-SEO CHOI received the B.S. degree in computer science from Kangwon National University, Republic of Korea, in 1996, the M.S. degree in computer engineering from Sogang University, Republic of Korea, in 2000, and the Ph.D. degree in computer engineering from Chungnam National University, Republic of Korea, in 2011. Since 2000, he has been with Electronics and Telecommunications Research Institute (ETRI), Daejeon, Republic of Korea, where he is currently working as a Principal Member with the Department of Cyber Security Research Division. His recent research interests include machine learning based network traffic analysis, vulnerability analysis, and intelligent cyber/network security.

MYUNG-SUP KIM (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in computer science and engineering from POSTECH, South Korea, in 1998, 2000, and 2004, respectively. From September 2004 to August 2006, he was a Postdoctoral Fellow at the Department of Electrical and Computer Engineering, University of Toronto, Canada. He joined Korea University, South Korea, in 2006, where he is currently working as a Full Professor with the Department of Computer Convergence Software. His research interests include internet traffic monitoring and analysis, service and network management, the future internet, and internet security.