IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 16 June 2023, accepted 29 June 2023, date of publication 7 July 2023, date of current version 14 July 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3293105

RESEARCH ARTICLE

A Feasible and Explainable Network Traffic Classifier Utilizing DistilBERT

CHANG-YUI SHIN[1], JEE-TAE PARK[2], UI-JUN BAEK[2], AND MYUNG-SUP KIM[2], (Member, IEEE)

[1]C4ISR System Development Quality Team, Defense Agency for Technology and Quality, Daejeon 34327, South Korea
[2]Department of Computer Convergence Software, Major in Network Management, Korea University, Sejong 30019, South Korea

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

This work was supported in part by the Korea University Grant; in part by the Technology Innovation Program Grant funded by the Ministry of Trade, Industry & Energy (MOTIE, South Korea) and the Korea Evaluation Institute of Industrial Technology (KEIT) (Development of SaaS SW Management Platform Based on 5Channel Discovery Technology for IT Cost Saving) under Grant 20008902.

ABSTRACT While user-oriented service industries are rapidly growing, various network devices provide these services through different access paths. Accordingly, the network flow is also increasing explosively. As demand for management related to limited network resources increases, the network traffic classification grows to prominence. Usually, a quick classification task was possible with hundreds of data composed of dozens of features. Afterward, deep learning models have proliferated owing to an outstanding performance that overwhelms existing performance based on hundreds of thousands of features and data. However, the deep learning models showing one of the best performances cannot be free from two facts. One is a lot of time and resource consumption. The other is an uncertain explanation of the process. We solved these problems. Firstly, we used two methods to overcome resource constraints. We modified the DistilBERT applied with knowledge distillation for using a compressed model and securing a remarkable performance. We used a lightweight packet with a header and partial payload for feature reduction. Consequently, our XENTC can process four multi-attribute packets simultaneously and effectively by removing the superfluity of features. And it achieved 97.0∼98.1% F1 scores. The required time to classify a packet using a trained model is 0.0093 seconds. Therefore, it can be one of the feasible solutions. Secondly, to approach human-understandable XAI, we analyzed the relationships between the features by associating them with the packet structure. At the specific point of the model's finished training, it was revealed what the important features of the packet were by counting the Top-5 number of times among the attention values. In addition, we visualized the classification performance of the model using t-SNE to enable intuitive understanding.

INDEX TERMS Encrypted network traffic classification, deep learning, NLP, BERT, knowledge distillation, DistilBERT, XAI.

I. INTRODUCTION

A. MOTIVATION

While tremendous personal devices are connected to the network for access to various services, service industries such as SNS, health care, transportation, and energy management are growing. So, the sharing scope of data and network resources among these service industries is also rapidly increasing. Such services are often established between users and cloud servers. A satisfactory solution to guarantee the network Quality of Service (QoS), intelligent network operation and maintenance, and network security of network-based services is network traffic classification and identification technology, which is continuously researched along with the development of computer network technology [1], [2]. In the early days, port-based approaches tried to classify traffic only with the port number of the transport layer. Using the information registered with Internet Assigned Numbers Authority (IANA) made it possible. However, because application servers gradually began to use unknown

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

ports and to allocate ports dynamically in real-time, using port-based approaches was no longer an acceptable solution.

In consequence, payload-based approaches have been proposed. These used a Deep Packet Inspection (DPI) technique that more closely examines the contents of the received packets. It started with the premise that an application or protocol has a signature, which is a specific pattern. Accordingly, several algorithms capable of detecting the pattern have been developed [3], [4]. However, as personal information security became more critical, encryption technology [2] for network traffic's payload was applied. By encrypting the payload among the header and payload of the packet, the services on the network domain have ensured the security of the contents by using only the necessary header information for packet exchange. This development can be accepted as a growth in the security area. But the DPI paradigm is no longer a favorable solution owing to the unavailability of payload in network traffic classification. So, Encrypted Network Traffic Classification (ENTC) has become a blue-chip challenge.

For this reason, Statistics-based approaches [5] newly emerged. Statistics-based methods have been used in conjunction with port-based methods. Departing from the perspective of DPI techniques, statistics-based methods used statistical characteristics of the network traffic's flows. For example, these are analytic data such as packet size, the interval between packets, and flow continuity. Furthermore, the various methods, which have utilized Machine Learning (ML) algorithms concurrently, were proposed [2], [6], [7], [8].

Over time, the tremendous network traffic of personal devices and applications has exploded. And it was expected to continue this trend consistently. So, advanced research has begun to counteract the requirement for service assurance in the manner of real-time [9], [10], or distributed computing [11].

Moreover, Artificial Intelligence (AI) methods utilization, along with the success of the Convolutional Neural Network (CNN) technique, has increased remarkably. For example, CNN has symbolized a representative of deep neural networks by successful optimal results in Computer Vision (CV). In line with this, the DL method in other fields has performed significantly higher than the existing ML method. As a result, it has become generalized to utilize DL. Moreover, finding unapplied areas is challenging.

In even the field of ENTC, DL-applied approaches [12], [13], [14] are becoming a new trend. Lately, a Transformer model [15], which is superior in sequential data processing to Recurrent Neural Networks (RNN) in the field of Natural Language Process (NLP), has emerged. Because packets are characterized by time series continuity, it was expected that the NLP mechanism could be suitable for network traffic. Accordingly, approaches that apply the NLP mechanism to ENTC problems have been in the limelight [16], [17], [18].

B. PROBLEM STATEMENT AND STRATEGY

The inference process behind conclusions drawn by artificial intelligence models remains a black box. The eXplainable Artificial Intelligence (XAI), which aims to bridge the gap between the quantitative output of AI models and human comprehension by opening an opaque black box, has also become a new task to solve in this field of ENTC continuously [19]. In addition, AI models are demonstrating remarkable performance, but their problem of enormous resource consumption remains. It may be impractical to assume that high-performance devices, sufficient to cover the burdensome computational cost, can be deployed in most places. Therefore, it can be required to reduce the processing burden, including latency, rather than managing the cost of high performances devices with non-infinite resources, which includes the memory footprint and electronic energy consumption. For utilization in various areas, the AI model's feasibility would be needed to secure overcoming computational cost through efficiency, which uses model refinement [20] and fine-grained feature reduction.

In this paper, we aimed to ensure the feasibility of solving the ENTC problem with an acceptable result through fewer computations. Accordingly, we approached two kinds of lightweight to reduce the model's computation amounts. One aspect concerns the input data, and the other concerns the model.

As aforementioned, one aspect of lightweight was to reduce input data. Therefore, we utilized the intuitive perspective in the previous approach [21] as a precept. They researched that the first few packets contain essential information in the application's negotiation phase. In addition, it was believed that the characteristics of each traffic could be found sufficiently with the header, which includes the length information related to the payload and partial payload. For this reason, it has become starting point to provide stark contrast methods not to find patterns in encrypted parts that occupy most of the input data [17], [18] in terms of input data.

As the other aspect of lightweight was removing the superabundant model's layers, we selected the DistilBERT [22] to apply the NLP mechanism in the ENTC field and reduce the model's layer. A tailored model of Bidirectional Encoder Representations from Transformers (BERT) [23], a large-scale model, does not have much meaning in the language domain. However, in the ENTC area, where network resources are limited, the lightweight model showing remarkable performance could have considerable meaning. The ensemble knowledge is obtained from a complicated model and compressed to be suitable for mobile devices, i.e., the superabundant values are distilled into compacted values. The above process indicates Knowledge Distillation (KD) [24]. Furthermore, the DistilBERT model has already been verified in the NLP area for its fast and remarkable performance by miniaturizing computationally expensive BERT. It was expected that applying lightweight data to

such a lightweight NLP model would positively improve the inference speed of the model.

In summary, DL models with excellent performance require both feasibility and interpretability regarding economy and usability. The above perspective, which means that light-weight input data could contribute to reducing the computation, was regarded as also helpful from the aspect of the model simultaneously. In order words, we contemplated that it can make a relation a virtuous circle. In addition to reducing time consumption, we considered that the significantly minimized input data could also support the interpretability of the model. So, we thought the lightweight data could be established as a concurrent baseline of feasibility and interpretability.

Significantly, the method of the inputted packets lightweight assumes below following preferentially. Input data in the NLP field has a different characteristic from input data in the ENTC field. In other words, it can be possible to classify traffic even though all payload of the packets are not used as input data [25].

In addition to the insightful method of applying lightweight data to a lightweight model suitable for time-series continuous data, an intuitively perceivable explanation, which uses corresponding direct relation between inputted byte and packet structure, was presented in Sections III and IV.

C. CONTRIBUTION

In this paper, we proposed a method aliased as an eXplainable Encrypted Network Traffic Classifier (XENTC). The proposed method has three main contributions to the ENTC problem.

- **Feasibility.** We have presented a feasible alternative to limited network resources. Compared to approaches [17], [18], which were proposed to allocate a lot of computation time for inference to find patterns in encrypted parts of network traffic, we have dealt with not the whole data of the packet but header information and partial payload. Our method has made a remarkable performance result using both representing of these data to be recognized each byte unit and modifying the NLP model to fit in well with the ENTC problem. Using traffic type data of 6 classes, the classification result for the inputted packet was a 98.10% F1 score. Using the application type data of 15 classes, the classification result for the inputted packet was a 96.99% F1 score. Regarding time consumption, XENTC can be a distinguished traffic classification model. DistilBERT, a compressed BERT applied with Knowledge Distillation (KD) technique, and lightweight input data were utilized simultaneously in our ENTC field. In applying the NLP model, a tokenizer specialized for network traffic was used, and no separate time was allocated for tokenizer learning and pre-training. There were no problems in the learning process, even though the integers, the converted value from vocabulary in the NLP field, were used as it is. Based on our general HW and the preparations for our experiment, the time needed to train the model for 20 epochs was only 1 hour and 40 minutes, and the time required to test with the trained model was 0.0093 seconds/packet.
- **Comparative Learning.** The training and test process was redesigned for only one packet to improve the model's accuracy. Specifically, our model was specialized to learn and classify the network traffic data of four other packets simultaneously. Classifying several other single packets simultaneously is named Comparative Learning (CL) in the mechanism of our XENTC based on a DistilBERT model. CL, based on a distinction within reciprocal characteristics of input data, has a productive effect, so the model completed training for 20 epochs showed an 0.82% higher F1 score than not.
- **Intuitive Interpretation.** Explaining which factors were used importantly for classification in the model inference process has been applied. The self-attention mechanism, which finds an attention value in the individual element perspective, was improved to recognize elements considered important from an overall point of view. For the explanation, each byte has mapped in a unit of bytes according to a single packet structure for permission of human intuitive perception. So, our explanation method is differentiated from an explanation method applied the CNN. Because CNN provides a mechanical explanation using the arithmetically calculated values passed through the convolution layer as data, it is restricted to enable an intuitive human understanding of the explanation. Similarly, it should be noted that using the LIME [26], [27] can provide a fragmentary superficial interpretation of the inference process rather than overall semantic reinterpretation. From a methodological point of view, we have used the NLP model as the primary frame of our model and did not conduct additional training on the tokenizer. So, each attribute value was maintained in its original state and can be used for analysis. As a result, it enabled an intuitive interpretation of the inference process.

This paper is organized as follows. Section II contains related works which are used to solve the problems of network traffic classification. Section III describes the performed work. Section IV describes the obtained experimental results and analyzes them. Section V provides additional experiments and discussion. And Section IV describes the conclusions.

II. RELATED WORKS

Feature selection by researchers has been performed even before the advent of machine learning [1]. But the number of features applicable to ML models may be limited considerably compared to the DL models. And most of the feature selection was made by the internal standard

of the researchers or by a method devised independently of the model process. Moreover, even though it is an ML model, there have been exceptional cases in which the model even directly intervenes in feature selection based on feature importance [28], [29], [30].

After the appearance of the DL model, more numerous features can be applied, which is a striking contrast to the ML model. And the role of feature selection has been transferred to the DL model, which determines almost autonomously which features are important. However, features not entered into the model after being refined according to the researcher's data engineering plan cannot be included in the model's consideration. In various fields, the utilization availability of the DL model can be clearly distinguished from the previous ones in research extent, which is related to the number of features available. So, its results have worked the same in the field of ENTC. Since there is little limit on the number of features that can be input into the DL model, numerous features have been entered into the DL model at once [14], [31]. Port information, payload (regardless of encryption or not), numerous statistical information that can be extracted as it is sequential data, and various other data have been included.

By the way, it can be that if more than hundreds of thousands of parameters are used, the negative effect on the model's accuracy is not significant, but it is not negligible [32]. Exceptionally, the possibility that the performance result can even differ depending on the model could not be excluded. Back to the point, the features pass through the affine layer and are increased on the layers. These are used as parameters in the model. Therefore, it is unimpeachable that the number of parameters the model handles is much greater than the number of input features. In this regard, we examined it in detail using our model. And it has been presented in Section V, 'Discussion.'

Concerning the viewpoint mentioned above, consideration of the quantitative aspect corresponds to both features and parameters. Therefore, the order of related work is as follows. First, before the DL model, we investigated early approaches (port-based, payload-based, statistics, and ML-based), in which the number of applicable features was relatively limited compared to the DL model. Then, we explored the approaches concerning the DL model that is less constrained by the available number of features. Afterward, an approach to efficiently improving a huge DL model has been researched. Finally, a recent method to insert packets directly into the DL model and an investigation concerning an advanced AI, i.e., XAI, has been executed.

A. METHOD USING RESTRICTIVE FEATURES

Generally, each method has been applied sequentially in the order of port information, payload, statistics, and ML. And approaches using a combination of several methods simultaneously have even appeared. The sublime combination of methods also has resulted in improved performance.

1) PORT-BASED METHOD

The port-based approaches infer network traffic by finding the port number in the SYN packet of the TCP 3-way handshake and looking up the port number registered in IANA. However, applications using ports unregistered with IANA had emerged, and these approaches had been limited.

To overcome this disadvantage, an approach, which classifies network traffic using both the port number and analyzing the variation of continuous packet size [33], was proposed instead of analyzing the payload itself. It used the unique characteristics of maintaining a session through the same port connection with each network traffic. As a result, it achieved 96% accuracy due to utilizing different Packet Size Distribution (PSD) information. This research method has become a guiding light of good results by utilizing summary information on payloads besides the port number.

2) PAYLOAD-BASED METHOD

Payload-based approaches are usually signature-based methodologies grounded on whether the presence of a known string exists in a packet's unencrypted payload. Reference [34] presented a method to improve accuracy by classifying according to the following three steps. As a first step, the port number was checked. As the next step, the payload of the first packet was checked to find the known signature. As the last step, the payload of the entire flow was inspected. The performance result of each step showed an accuracy of 69%, 79%, and almost 99%, respectively. It was a prominent part that increased the performance by coping with the computational burden step by step according to the performance standard. The computational burden required for high performance is a weakness.

Additionally, a limitation of the research then was that sharing ground-truth public datasets was restricted [5] due to personal security issues. Because the payload had not been encrypted, it was using their collection of ground-truth data that researchers classified network traffic for their research [35]. Therefore, since the types of collected network traffic were diverse, it took discrete work to compare each result objectively.

3) STATISTICS-BASED AND ML-BASED METHODS

Starting with payload encryption [2], [36], network traffic classification using the payload-based method has been limited. Accordingly, a technique has been proposed using the sequence signature of payload size as additional data [37]. They generated a sequence signature using the order, direction, and payload size of the first N packets in each network traffic.

Afterward, abundant approaches using statistical features from network flow's characteristics were proposed beside header information. However, those approaches did not access the internal data of the payload. Because the time-series features were associated with exterior information of the payload, such as length, it was applicable regardless of

whether it was encrypted, so it was used with ML algorithms such as Random Forest and K-Nearest Neighbor [38]. Reference [39] used various ML algorithms such as Naïve Bayes, C4.5 Decision Tree, Bayesian Network, and Naïve Bayes Tree. And the authors used 22 features obtained from flows and compared each performance. All four algorithms achieved over 95% accuracy. The various heuristic features were applied equivalently to various ML algorithms to compare their superiority. The research reflected the trend of the times when ML algorithms were prospering.

In this period, features were selected according to the researcher's design or an algorithm separate from the ML model or an algorithm applied inside the ML model. In most cases, dozens of features were selectively used. However, the allowable number of features compared to the DL model was relatively limited.

B. METHOD USING LESS RESTRICTIVE FEATURES

1) REVIVAL OF THE DL MODEL

As mentioned before, the allowable number of features is less restrictive, and feature selection is performed by a model in DL [40]. The feature selection process is executed by weighting the parameters related to important features entered into the model to obtain the optimal solution. With the activation of backpropagation [41] and the development of memory technology, computational power using GPU has substantiated DL merits. And its progress is still ongoing. Hence, adjusting weight values relevant to finding the optimal solution through training the DL model can be progressively accelerated.

Reference [13] proposed a method of applying both CNN and RNN, the representative deep learning techniques in CV and NLP, to the ENTC problem. First of all, automatic feature representation of network flows was also applied. Six features (source port, destination port, payload size, TCP window size, packet arrival time, and packet direction) were selected by mixing four header information and two statistical data. Those were chosen from a flow composed of 20 consecutive packets. The six features were rendered as 2D-image shapes, and then CNN was utilized to extract synthetic features. After extracting features through CNN, the structure of the model was designed to connect to the LSTM network. As a result of inference, the performance value achieved 96.32% accuracy. Their deep learning model used both a spatial property of feature vectors and the time-series property of feature vectors. Combining the CNN and the RNN is a strategic method that utilizes the above two properties. It can be the noticeable achievement of strengthening the strength and making up for the weakness.

The self-attention mechanism of the Transformer model [15], which has been proven to be dominant in time-series data processing, has begun to be applied to the ENTC problem [17], [25] from the field of NLP. Subsequently, approaches [18] that apply the BERT model [23] using the Transformer structure to find features in encrypted payload also appeared. As the name suggests, BERT uses Transformer encoder layers to recognize input data from a bi-directional perspective. BERT utilizes the three input vectors simultaneously. The first is an embedding vector in which each sentence is divided into token units. The second one is a position vector for context. The last one is a segmentation vector for which sentence each token belongs to. It has a pre-learning mechanism using the three input vectors in two ways. One of the two techniques is to select 15% of the inputted data. Then it generates a [MASK] token and learns to predict the original word. Another technique is to understand the relationship between two sentences using comparison. And the other one is predicting whether the following sentence belongs to the category of the identical attribute of the former one.

In [18], the authors used five consecutive packets and a single packet as input data for the pre-training and fine-tuning. It utilized this BERT model by using 512 bytes among the entire 1,500 bytes of a packet. It achieved 85.19% and 99.62% accuracy on the classification for a session and a single packet, respectively. Their study provided a distinguished point to announce that it is possible to classify network traffic by including packet headers and the partial payload using the NLP model. However, the model used the one-third packet during the training process. The resources and time required for the computation were not disclosed in their study for the pre-training process, but it could be expected to be significant. Actually, the training time measured using BERT with 512 bytes as input data was time-consuming, as presented in Table 3 of this paper.

Accordingly, in the ENTC area where network resources are limited. Even if a model shows significant performance, it may be reasonable to question whether it can be a viable alternative if it is heavy and consumes considerable time and resources in the network traffic classification. In this regard, we examined it in detail through comparison. And it has been described in Section IV-D.

2) IMPROVING THE FEASIBILITY OF DL MODELS

Research on reducing the DL models has been explored to prevent the enormous computation and processing time of Deep Neural Networks (DNN). Even though it cannot avoid a trade-off relationship between accuracy and the quantity of computation, research is underway to overcome the adverse effect, especially in restricted circumstances.

The reduction of the DL model can be classified into two types. One of the two types is a method of using a lightweight algorithm from the beginning [42], [43], and the other is a method of lightening the trained model's layer [24], [44]. Compared to the former, the latter may take considerable time to train the model but may have the advantage of obtaining good performance. The Knowledge Distillation (KD) belonging to the latter is a mechanism that decreases training and inference time by reducing the number of layers of the student model to layers of the teacher model. Applying

temperature to the soft-max function is used for it. As the KD mechanism is a significant part of the baseline model used in our method, we allocated a specific part for an explanation in Section III, 'Methodology.'

3) PUTTING PACKET INTO MODEL

References [25] and [27] selected several types of models and demonstrated that the model's performance could differ depending on the packet's various input lengths, even though they were configured using similar models. So, from a broad perspective, [25], [27] can be noticeable research on the fact that different data can be used depending on the researcher's intent, even if the same model is used. In other words, the model's performance may vary depending on the selection of the input length. The model can be determined by the purpose of the researcher's intended goal. Accuracy and efficiency or both may be goals. But it should also have been concerned that having a lot of data input to the model can also degrade performance [32].

Based on the above lesson, the two aspects pre-considering the final result of the model in the planning stage can be organized as follows. Firstly, it needs to be focused on the fact that increasing input features lead to increase processing time, and the accuracy of the model even may not be improved. On the contrary, the performance may even decrease [32]. Secondly, as the lightweight model can contribute to reducing the massive computation and processing time of DNN, it should also be focused on the fact that reducing the inputted data can contribute to the model in the same aspect. In this paper, focusing on the above two aspects, we planned feature and model reduction for the model's feasibility within a $1 \sim 2\%$ lower performance. We regarded that the degradation of the model's performance can be a manageable and acceptable problem under the condition that the selected decision leads to reducing processing time or resources.

Reference [31] dealt with a multimodal technique that selects and applies multiple data simultaneously in terms of input data. And [14] used both statistical features chosen by researchers and features selected by deep learning models simultaneously. In [21], over 80% accuracy was achieved using the first five packets of each TCP flow. From a chronological point of view, the approach [21], which intuitively pointed out that the first few packets are important in traffic classification because they contain essential information in the connection stage of the application, can be worth paying attention to.

So, in this paper, we designated the first four packets as input data in bi-directional flow (i.e., session). The number of input packets is related to the size of bytes available for input into our model, described in Section III-D.

In [25], Attempts to achieve real-time traffic classification, which can be inevitable in network traffic classification, have emerged at a new level in the viewpoint of input data. The proposed model did not deal with flow-level data connected with multiple continuous packets as input data. It is only presented to use the specific part of a packet as input data by a single packet unit. The particular part of the payload was used in addition to the header information of a single packet unit. A referential experiment was presented to find that optimal value positively affects the classification performance among the first 40, 50, and 60 bytes of the L4 payload. It achieved an accuracy of 90.33% by selecting only 50 bytes of payload as the optimal input data value. What is noteworthy in their research is that they tried to find the optimal value under the premise that good results can be obtained even if only a small part of the payload is included in addition to the header. In a different view, by applying Transformer's self-attention mechanism, attention weights of input data were calculated to find the ranks of bytes important for classification. However, as the CNN was used right before the model's classifier layer, finding the corresponding bytes among the input data connectionally was impossible.

We analyzed the above result and evaluated that not all data in the packet need to be entered into the model. So, we considered an acceptable level of accuracy can also be obtained using a partial packet payload beside header information. So, in this paper, we designated the first 63 bytes of a packet as input data into our model. And content concerning the optimal selection of the payload size inputted to our model is described in Section V-A.

Furthermore, other research finding on the critical input data that influences the classification results of the model are continued in the right next Section.

4) XAI IN ENTC PROBLEM

In various scientific research fields, DL models based on ANNs (Artificial Neural Networks), which include hidden layers, have been becoming relatively recommended solutions by surpassing the performance of existing ML models. By the way, the higher the performance of both ML and DL models, the higher the black box-like opacity of the result's derivation process. Therefore, to increase reliability using explanation, the necessity of research on both transparent design and post-hoc explanation for AI was raised [45], [46].

To progress further from the classification result of the DL model, finding how to interpret and utilize the final meaning of the features weighted by the model is becoming the norm even for interpretation in the ENTC area.

Reference [47] suggested the interpretation possibility of ENTC with a model designed using a genetic algorithm and the 1D-CNN. A dominant feature selection was preceded among the statistical flow features by adjusting the hyper-parameters of the genetic algorithm. Confirming the classification result using the 1D-CNN to determine the dominance of the weighted features was performed later. By the way, their approach needs to be discussed in the following two aspects. Firstly, although the method of attempting to interpret the model was unusual, it was not an interpretation of a pure deep learning model. Because the genetic algorithm was used in adjusting the weights of the features based on the pre-trained model, this is an

interpretation of the feature selected by not the DL model but the genetic algorithm. There was a difference from the interpretability of the general DL model because their DL model adjusted the previously selected features again within its limited domain range. A general DL model conducts feature selection and weighting features simultaneously. Secondly, since the features applied to the model are statistical values, additional utilization from the user's point of view may be optional. While the original packet field has meaning, selecting the post-interpretive statistical values has a heuristic characteristic for each researcher. Therefore, the range of statistical features set in advance may change at any time heuristically. In this regard, we used both the utilization of DL's feature selection and the original value of the packet field as it is in our research.

Reference [25] that applied both the self-attention mechanism of the Transformer model [15] and the 1D-CNN mechanism to model simultaneously was presented for the possibility of interpreting the model for the ENTC problem. They used a first packet as input data for online network traffic classification and divided a single packet by a byte unit. Each byte had gone through the Transformer layer. After that, the clustered bytes composed of an adjacent byte were represented using the 1D-CNN before a classifier layer [40]. But the attentive weight of the model has been separated from the intrinsic value after passing through the CNN layer. Even though the above process of going through the CNN layer comprised partially, only the possibility of superficial interpretation related to byte, which played an important role, could be presented. That is because the interpretation using representative values, not such as a part of a picture that people can see at a glance, is the ambiguous context beneath human perception in other areas not related to the perceivable image.

Generally, each feature recognized by the model can be increased and represented by passing the affine layer, and the model optimizes the related weights. The CNN mechanism in the CV can suggest an explanation by finding that the features of the represented middle layer refer to a specific part of the original image [46]. But the byte value represented by the subset of packet header fields may be an unknown computed value in the ENTC area.

To wrap it up, feature interpretation concerning the model using a convolution layer cannot be appropriate for human comprehension according to the applied area. That is because the input data's features represented by group units were at a level humans could not perceive directly, at least in the ENTC area. Accordingly, adopting the XAI technique can be a research area that needs to be carefully planned. It is reasonable to be developed by simultaneously considering both the baseline of the DL model and the characteristics of the field to which the model is applied.

In this paper, we proposed a XENTC model that can be understood and utilized by finding the importance of each byte unit not by the unknown computed value but by the connection with the packet structure.

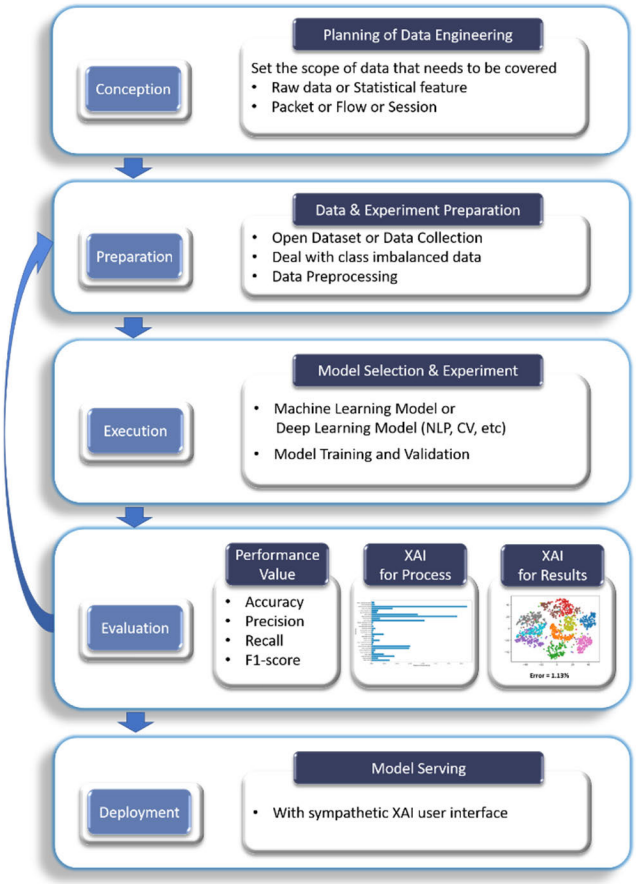

FIGURE 1. The overall framework for model design and implementation.

III. METHODOLOGY

We have determined outline that the problem of classifying sentences by processing data sequentially in the NLP field can be applied to the issue of the ENTC field. Therefore, we defined the following associative relation. A network packet can correspond to a single sentence, and a session composed of bi-directional flows can compare to a collection of several conversational sentences.

In detail, the semantic difference between session data and flow data, which were used in this paper, is as follows. Session data consists of packets sent and received in both directions. Flow data, contrary to session data, consisted of one-way packets. As mentioned earlier in Section II, we designated the first packet or first four sequential packets in session as our primary data [21], [25].

In constructing the model, we referred to the general framework [38] that presented concisely and practically the sequential process. So, we selected the parts that were useful to our plan and modified the parts that needed to be developed and specialized. And we expanded the methodological framework for the DL model by designing the other additional process.

Therefore, we have also referred to a sequentially organized process from the data engineering planning to the XAI

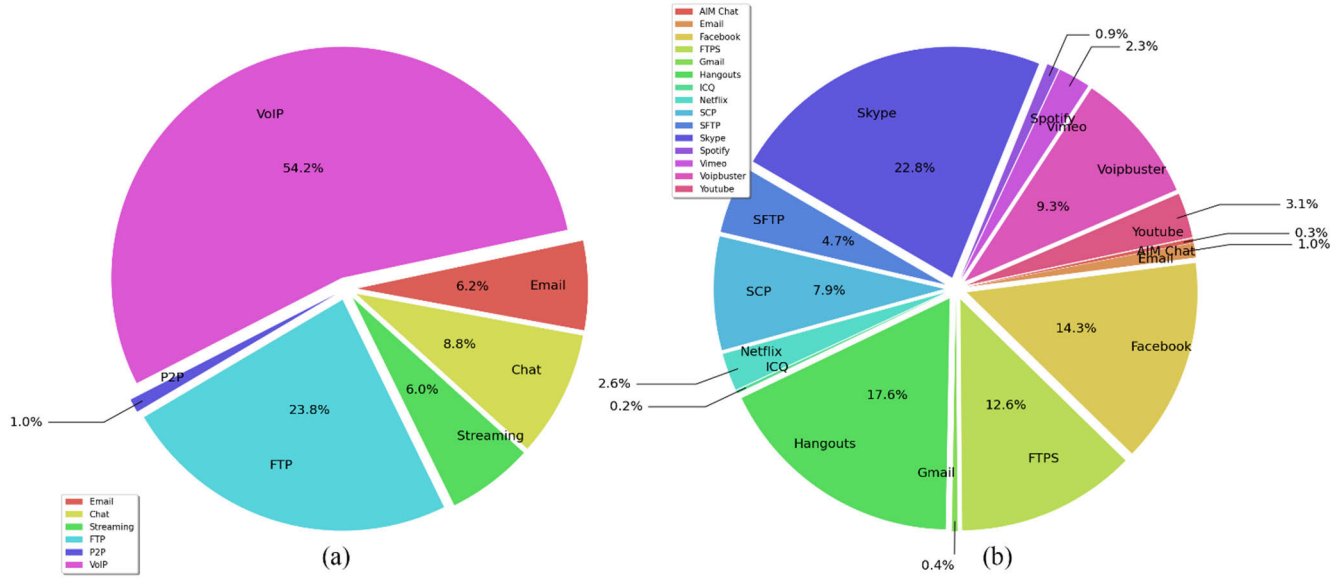

FIGURE 2. Distribution of bi-directional flow data. (a) A pie chart showing the distribution of traffic type, which is composed of 6 classes. (b) A pie chart showing the distribution of applications, which is composed of 15 classes.

TABLE 1. Class composition of each property.

Category	Classes
Traffic Type (6)	VoIP, File Transfer, Streaming, Chat, Email
Application (15)	Skype, Spotify, Vimeo, VoipBuster, Youtube, AIM Chat, Email, Facebook, FTPS, Gmail, Hangouts, ICQ, Netflix, SCP, SFTP

for the ENTC problem [48]. Our method has been established this way, and the overall framework applied to this Section III is illustrated in Figure 1.

A. TARGETED DATASET

The emergence of payload encryption has brought a new phase to the network traffic classification problem. Previously, data sharing had been restricted due to personal security issues. Payload encryption has solved this problem related to network traffic research by enabling data sharing. As a result, it has become possible to use public data to compare each performance objectively. So we have chosen the VPN-nonVPN dataset (ISCXVPN2016) [49] that was captured from real-world traffic, and it was also commonly used by other researchers. Therefore, a comparison with different experimental results can be presented, and it was expected to have a practical relative meaning.

The composition of both Traffic and applications corresponding to the ISCXVPN2016 are shown in Table 1. The two datasets comprise 6 and 15 classes, and the distribution is presented in Figure 2 by each class. The 15 classes' data consisted of 134,563 bi-directional flows. And the six classes' data consisted of 27,814 bi-directional flows, which were obtained in smaller quantities by random sampling for time reduction of various comparable experiments. Those were randomly selected by maintaining an imbalance of the distribution ratio of the public dataset. Because it was considered that making a similar environment to observe the objective experimental results in the actual use of the model can be an essential factor. As illustrated in Figure 2, the distribution of specific network traffic data reflects the biased phenomenon in the environment of actual data austerely. By each quantity, the two kinds of dataset's split ratio for training and testing were set to 9:1 in the case of 15 classes dataset and 8:2 in the case of 6 classes dataset.

However, since the imbalanced class data can cause either overfitting or underfitting of the model in the training process, pre-processing the imbalanced class data may be necessary before the model's training. Already, [50] dealt with this class imbalance problem. They revealed that an unintended situation could significantly hinder the model's training using such a class-imbalanced data set. For example, while focusing only on the classification of the main sample, a crucial mistake of ignoring or misclassifying a minority sample can be made unexpectedly.

Accordingly, the necessity of pretreatment on the imbalanced class data was decided for training in our research. To solve this class-imbalanced problem, we have chosen an uncomplicated method to bring the effect of balanced training. In this work, relative class weights have been applied to the loss function for an imbalanced distribution of classes. It describes how we dealt with the class imbalance problem in this Section III-E, 'Experiment Preparation.'

B. DATA PREPROCESSING

As aforementioned, [32] revealed that the number of features entering the model could differ depending on the researcher's data engineering plan. And subsequent results can differ according to it [51]. So, to find a lightweight unit suitable

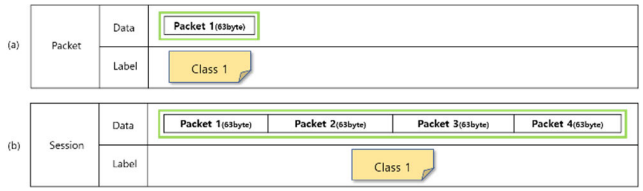

FIGURE 3. (a) The first 63 bytes of a single packet are extracted. (b) The first 63 bytes of each packet in a session are extracted and concatenated.

Algorithm 1 Preprocessing

```
Input : PATH = Path(dir path of PCAP files), MAX_BYTES = 63, MAX_ROW = 4
Output : packets = concatenated packets
1:  packets = []
2:  for packet in each_session(read_pcap(PATH)) :
3:      if packet_number(packet) > MAX_ROW : go to next session
4:      if discard_specific_packet(packet) : continue          # no-payload-TCP or DNS
5:      packet = remove_ethernet_header(packet)
6:      if udp = = categorize_packet(packet)          # distinguish protocol
7:      packet = pad_udp(packet)          # pad with zero until 20 bytes
8:      packet = mask_ip(packet)          # convert sre & dst address to 0.0 .0 .0
9:      packet = pad_cut_func(packet)
10:         # fix the size into MAX_BYTES using zero_padding or cutting
11:     packets.append(packet)
12:     save_data(packets)      # save the preprocessed packets
```

for the XENTC model structure, we set the size of the packet to the first 63 bytes, 126 bytes, and 189 bytes as alternatives. Through experiments, it was concluded that the suitable unit of input data is the first 63 bytes of the packet. Relevant details are described in Section V, 'Discussion.'

Accordingly, we light-weighted packets and used them as the model's input data. Figure 3 (a) demonstrates that only the first 63 bytes of a single packet are extracted. Figure 3 (b) illustrates that only the first 63 bytes of each packet in a session have been chosen and concatenated. Depending on the purpose of the model, the above light-weighted single packet or session data can be selectively inputted into the model.

In [52] and [53], research results presented that it was helpful for model optimization if local minima were generated as internally distributed representations through high-level abstraction before input into the model. Therefore, it means that pre-processing and converting the data representing local minima play an essential role in deep learning. So, we applied the method used in [54] and eliminated the ethernet header, DNS packets, and packets without payload. Furthermore, the overall preprocessing mechanism, including padding and the distributed representation, is presented in Algorithm 1.

Subsection C, which follows, introduces the important theoretical background of DistilBERT [22]. In DistilBERT, the superabundant layer is distilled into the concise layer. The above method indicates Knowledge Distillation (KD) [24]. We utilized DistilBERT for ENTC because we needed a lightweight model of BERT, which shows a remarkable performance but has excessive Transformer layers. Model lightweight is a fundamental concept because it is part of the foundation in which many Transformer layers used for model learning can be reduced. Using the KD method, DistilBERT

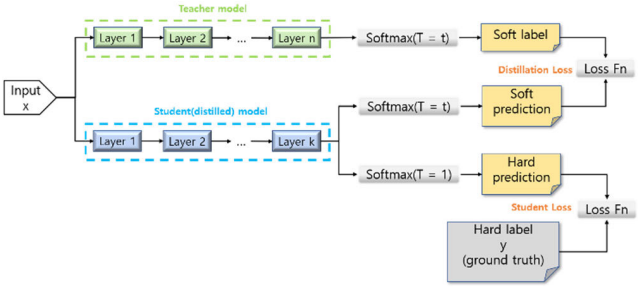

FIGURE 4. The learning process of the student model acquires dark knowledge from the teacher model by applying temperature to the soft-max function.

lessens many Transformer layers and progresses the learning process of its model simultaneously.

This pre-trained model of the DistilBERT has been designed to learn based on language data. So, in subsection D, we describe the modifications we made to make this DistilBERT able to fine-tune network traffic data straight away without any additional pre-training using it.

C. MODEL BASELINE

In ENTC, various methods such as CNN, RNN, and Attention mechanisms can be used. Among them, we focused on BERT's process and performance, which makes bi-directional access to sequential data based on the transformer's layer. Furthermore, we finally selected DistilBERT as our model baseline according to the model's weight reduction necessity. In the NLP field, it has already been verified with a 97% performance ratio compared to BERT, even though it is 40% smaller than BERT, which uses 12 Transformer blocks for model configuration. DistilBERT uses only 6 Transformer blocks.

The small student model of DistilBERT makes its small layer by utilizing the KD method, which is used to acquire knowledge from the big teacher model. Figure 4 illustrates the KD method applied to DistilBERT. The general 'softmax' converts the logit, z_i, into q_i through probabilistic computation. Unlike the soft-max output, the probability value q_i for the i-th class in Equation (1) is based on obtaining dark knowledge by entering the value T, which means temperature, as the denominator. When the temperature is lowered, the label becomes hard. When the temperature is increased, the label becomes soft. So, the distillation process can be established by the control of the temperature. The control is executed in the learning process automatically. The soft target, the knowledge of the teacher model, is taught to the student model through the loss function, as shown in Equation (2). Below, L is the loss function, S is a student model, and T is a teacher model. In addition, (x, y) is each input data and its label, respectively.

The value L_{KD}, which means Distillation Loss, is also used together with L_{CE}, which means Cross Entropy Loss while maintaining the same temperature when it compares the soft label of the teacher model and the soft prediction

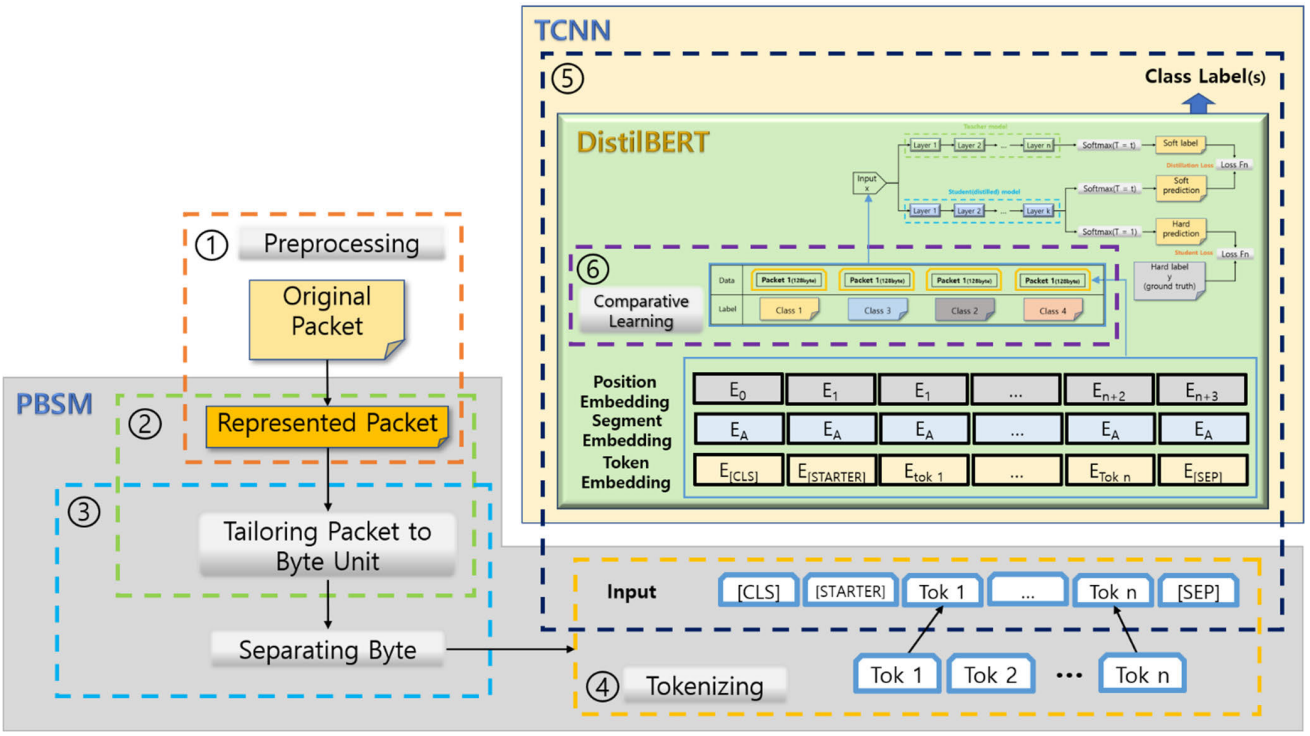

FIGURE 5. The overall configuration of our XENTC.

of the student model. The value θ is the learning parameter of the model. The value τ is the temperature. The value λ is a balancing parameter. If λ is large, it means that L_{CE} is treated more importantly in the learning process. The dark knowledge, $\mathbb{D}$, that can be obtained from (x, y) composes the loss.

$$q_i = \frac{exp\left(\frac{z_i}{T}\right)}{\sum_j exp\left(\frac{z_j}{T}\right)} \tag{1}$$

$$L = \sum_{(x,y)\in\mathbb{D}} L_{KD}\left(S\left(x, \theta_S, \tau\right), T\left(x, \theta_\tau, \tau\right)\right) + \lambda L_{CE}\left(\hat{y}_S, y\right) \tag{2}$$

KD method, which learns through this dark knowledge, was used as a great foundation to create a small compressed model that is not far behind the performance of a burdensome large model. Using the above KD method, DistilBERT lessens many layers and progresses the learning process of its own model simultaneously. We have adjusted DistilBERT, corresponding to our model baseline, to use 5 Transformer layers for the teacher and 1 Transformer layer for student models, respectively. The following processes were devised to adapt DistilBERT to the ENTC field. The model in the NLP domain can be adapted to the model in the network traffic domain through the process of the following right next section.

D. DISTILBERT ADAPTED TO ENTC FIELD

To prevent overfitting or underfitting of the model, the unglazed data acquired from preprocessing are passed to the PBSM phase in Figure 5. The adaptation process to network traffic consisted of the following two phases. The first phase is composed of Packet Byte Separation Module (PBSM). This module converts the inputted network traffic data to a particular form suitable for the following TCNN phase. To recapitulate, it makes data a distinct unit that can be interpreted to articulate with the packet structure. The next phase is composed of Traffic Classification Neural Network (TCNN). And it is designed so that even the sequences of multiple property packets can be used simultaneously according to the researcher's selection. Accordingly, our eXplainable Encrypted Network Traffic Classifier (XENTC) comprises PBSM and TCNN. The above two phases are as follows in detail.

1) PBSM

The operation of the PBSM phase is demonstrated in Figure 5. The primary role of the PBSM phase is twofold. One is to convert the range of each field's value in terms of the packet structure to 256 dimensions for the intrinsic processing byte units. Since the data has gone through the preprocessing and PBSM phases, values are only the integers between 0 and 255. And the other is to enable maintaining the independence of each field so that the TCNN in the next phase recognizes it as an individual unit. First, the sequences are divided by 'space' so that these are distinct by individual byte units while maintaining the order of packet structure. And next, these are followed by converting the value into a character type. Each byte unit can correspond to a separate word or phrase in the field of NLP. In a semantic context, each byte unit

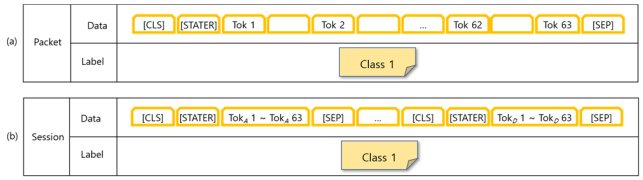

FIGURE 6. (a) Single packet. (b) Session. Each consists of special tokens (CLS, STARTER, SEP), packet information, and a space separator.

can correspond to a separate word or phrase in the field of NLP. Through distinction, i.e., 'space', the numerical values of each byte can be recognized as a non-merging word or phrase in the next TCNN.

Afterward, we used the Word Piece Tokenizer of BERT as a sub-word tokenizer. And STARTER is devised as a special token at the beginning of the packet header. As presented in Figure 6 (b), it ensures that the model recognizes the beginning of each packet when connecting packets for session structure. In addition, CLS and SEP, which are special tokens that indicate the front and the middle of sequential data, were applied without modification. When the tokenizer is used for the sequences of character type, the inputted spaces of the previous process are converted to a comma. Finally, MAXLEN, one of the model's various hyperparameters, is set to 128. These 128 bytes are designed to be structured as follows. It comprises 63 bytes of input packet information, 62 bytes of the 'comma' which is a blank separator between packet information in the previous step, and 3 bytes of the special token (CLS, STARTER, SEP).

In addition, 63 bytes comprise packet headers (40 bytes) and partial payloads (23 bytes). Above packet header is consisted of an IP header and a TCP header. Each value in a byte unit constitutes features, and a description of each byte is presented in Figure 12. However, partial payloads (23 bytes) are not separately specified because they are only encrypted contents. And the examples of 128 and 512 bytes generated through PBSM are illustrated in Figures 6 (a) and (b), respectively.

To keep the tokens generated by separating each byte of the packet, an additional training step is not given to the tokenizer. The non-processing of tokenizer training can be essential to protect each byte unit because each byte is used as an explanatory factor for the inference of the model from the XAI point of view. The related content to XAI applied to XENTC is described in Section V-G.

2) TCNN AND COMPARATIVE LEARNING

Typically, the training phase of the BERT model [23] is divided into pre-training and fine-tuning. Two kinds of loss functions can be used to pretraining the model. The one is acquired from Masked Language Model (MLM). It randomly selects 15% from the inputted sentences. After that, the sentences are changed according to the ratio of {8:1:1 = mask:replace: preserved} and are used as training data. And the other is acquired from Next Sentence Prediction (NSP), which guesses whether the two sequential sentences are consecutive.

Algorithm 2 Test Algorithm for Comparative Learning Using Batching

```
Input : testing_loader, NUMBER_OF_INPUT
Output : test_loss
1: for idx1, data in enumerate(testing_loader, 0) :
2:     ids = data['ids']
3:     mask = data['mask']
4:     token_type_ids = data['token_type_ids']
5:     targets = data['targets']
6:     outputs = model(ids, mask, token_type_ids)
7:     for idx2 in range(NUMBER_OF_INPUT) :
8:         test_loss += loss_fn(outputs[idx1], targets[idx1]) item()
9: return test_loss
```

Significantly, XENTC does not include an additional pre-training step [18] using network traffic data in the TCNN phase. Because the values applied in the previous phase are only integers between 0 and 255, the DistilBERT [22] model provided overtly as the 'distilbert-base-uncased' can be directly fine-tuned. So, the last classifier layer of the model is changed according to the number of classes. After that, the affine layers and classifier layer parameters are trained instantly using the data. It is one of the simple methods for transfer learning [55] using the replacement of the classifier layer while leaving the feature extraction layers as it is.

Furthermore, besides faster training time for an advantage of distilling knowledge [24], we improved the model to process multi-sequence inputs to maximize the training effect in terms of performance. The characteristic of the model that learns by comparing multiple attributes of four packets is called Comparative Learning (CL) in this paper. A model specialized for this learning structure can be trained more effectively. It corresponds to an inference task of NLP that receives a pair of sentence inputs in BERT. Explicitly, the CL applies DistilBERT's Sentence Pair Classification (SPC) task model to XENTC so that four multiple attribute sequences can be input into the model. In other words, TCNN is the extended SPC to proceed with the symmetric learning between four other packets by utilizing the relationship learning process from two sentences. So, it enables learning the logical relationship between multiple sequences of four packets.

Consequently, the method for automatic adaption testing according to the number of the inputted attribute is presented in Algorithm 2. It is designed to take sequences of four attributes as input to make the most of the characteristic of the BERT series model called batching.

The CL process is intended to increase the traffic classification performance's the chain effect by applying a lightweight packet to a lightweight model. Because XENTC can deal with several lightweight packets, both packet and session, as illustrated in Figure 7, are possible as input data, respectively. In connection with this, Figure 7 (b)

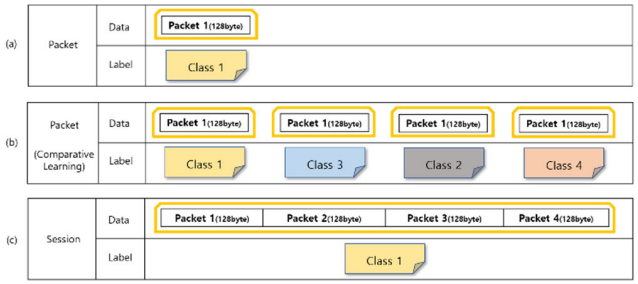

FIGURE 7. Three kinds of represented data can be inputted into the model. (a) A single packet and a single label. (b) Four different packets and four different labels. (c) A Session with four bi-directional packets, which were concatenated, and a single label.

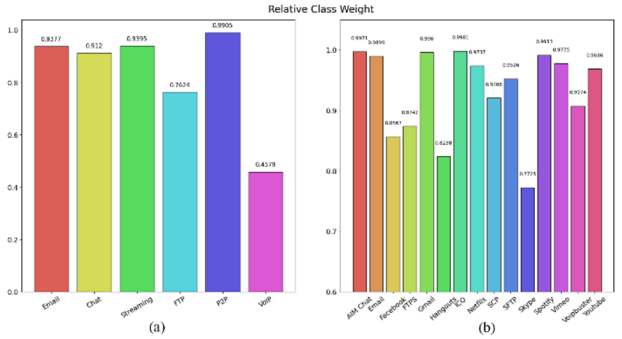

FIGURE 8. Relative Class Weight for improving imbalanced data problems. (a) Comparison of weight values of 6 classes. (b) Comparison of weight values of 15 classes.

demonstrates the four distinct packets, represented by 128 bytes, and four corresponding labels. And Figure 7 (c) reflects a session concatenated the four packets. Among them, the CL only applies to the four packets illustrated in Figure 7 (b).

That is because the previously reformed data can satisfy both the max sequence length (512) range limit and truncation mechanism characteristic of the BERT model, tasks using lightweight packets are possible. As demonstrated in Figure 7 (b), the sequences of multiple attributes are the first packet of each of the four different applications. Therefore, CL is to learn and evaluate them simultaneously by comparison. In other words, the particular learning effect of TCNN can be obtained through this relation learning using multiple attribute sequences.

To wrap it up, The CL mechanism using batching is an advantage of XENTC. A packet has been reformed for lightweight and reconstructed to be appropriate for batching because the CL utilizes the batching skill affiliated with the BERT series. Consequently, it uses a bundle of multi-attribute packets to learn more effectively than a single packet. Experimental results comparing both single-attribute packets and sequential structures of the multi-attribute packet in terms of learning and classification effects are described in Section IV, 'Experiment, Result and Analysis.'

E. EXPERIMENT PREPARATION

1) HANDLING IMBALANCED DATA

As aforementioned in Section III-A, to improve the data imbalance problem mentioned in [50] and [56], we used a simple value, Relative Class Weight (RCW).

$$W_i = 1 - \frac{Q_i}{\sum_i Q_i} \tag{3}$$

The value Q_i is the total quantity for the i-th class data. The adapted weight value of the i-th class data, W_i, can be obtained utilizing the total amount of all class data as the denominator in the weight calculation formula (3). The adapted weight value for each class, W_i, obtained by applying the formula (3) is shown in Figure 8. These adapted weight values are RCWs, and they were applied to the cross-entropy loss function for the learning process.

So, the imbalanced property between each class has been tuned relatively using W_i. A comparative experiment to uncover the effect of RCW on the learning process of the model is presented in Section IV, 'Experiment, Result and Analysis.'

2) OPTIMIZER SELECTION AND SETTING FOR LEARNING OPTIMIZATION

Generally, two influential factors can determine primally the degree of learning optimization of a model. Therefore, pre-consideration for two factors should be a base. One is to select an optimizer, and the other is to set the batch size and running rate suitable for the chosen optimizer.

Firstly, we considered the ability to reduce the unstable learning process at the beginning of learning as the optimizer adoption criterion. Therefore, in that regard, the verified RAdam [57] was selected as the optimizer. Even Adam [58] can support the excellent learning performance of the model. However, the inherent bad local optima problem was expected to affect the learning stability of our model adversely. RAdam is an optimizer that corrects the variance of Adam's adaptive learning rate term to solve the bad local optima problem and improves learning stability and accuracy.

$$\begin{aligned} &Var\left[\psi\left(.\right)\right] \\ &\quad = Var\left[\sqrt{x}\right] = \mathbb{E}\left[x\right] - \mathbb{E}\left[\sqrt{x}\right]^2 \\ &\quad = \tau^2\left(\frac{\rho}{\rho-2}\frac{\rho 2^{2\rho-5}}{\pi}B\left(\frac{\rho-1}{2},\frac{\rho-1}{2}\right)^2\right) \end{aligned} \tag{4}$$

$$\begin{aligned} &Var\left[r_t\psi\left(g_1,\ldots,g_t\right)\right] \\ &\quad = C_{var}, where\ r_t = \sqrt{\frac{C_{var}}{Var[r_t\psi(g_1,\ldots,g_2)]}} \end{aligned} \tag{5}$$

$$\begin{aligned} &r_t \\ &\quad = \sqrt{\frac{\left(\rho_t-4\right)\left(\rho_t-2\right)\rho_\infty}{\left(\rho_\infty-4\right)\left(\rho_\infty-2\right)\rho_t}} \end{aligned} \tag{6}$$

RAdam [57], which improved Adam's shortcomings, became a good alternative. Equation (4) was presented in RAdam to prove the initial learning state's instability using

the fact that the gradient follows a normal distribution and the adaptive learning rate term follows a scaled inverse chi-square distribution. Here, $Var\,[\psi\,(.)]$ is the square root variance, B is the beta function, and ρ is the degree of freedom. The ρ_t is the value estimated using the step size, t. And g_t means gradient. The degree of freedom can be seen as the progress of learning. When the obtained variation is differentiated by the degrees of freedom, ρ, it was revealed to be monotonically decreasing. Consequently, it has been proved the smaller the degree of freedom, the greater the variance. As aforementioned, it means that the property of incompleteness in the initial stage of learning has been proved. Using the variance equation (5) of the adaptive learning rate term inversely, equation (6), a rectification term that can make the variance consistent, can be obtained. Consequently, RAdam is characterized by obtaining the initial learning state's stability by multiplying this equation (6) by Adam. This means explicitly a heuristic warmup process has been reduced by RAdam.

Secondly, we contemplated that the appropriate relationship between batch size and learning rate suitable for the selected optimizer, RAdam, could be an essential factor. An approach [59], which suggested a positive correlation between batch size and running rate while comparing ADAM and SGD optimizers, became the basis for this consideration. We have come to the following important conclusions from their research. It can be a crucial point in our experiment that the designated initial learning rate and batch size may significantly influence the learning process. Accordingly, it can be essential to find a suitable learning rate parameter for the batch size of the learning process in the first stage. This value designation can rapidly lead to the optimal minima by smoothly setting the learning direction at the beginning of the learning process. Therefore, the batching in XENTC is between 1 and 4, a smaller value than the general one, and we regarded it as appropriate that the initial value of the learning rate should be set as a smaller value than the default one(1E-03).

Based on the above two considerable factors, we have concentrated on the designation of the learning rate parameter of the selected optimizer, RAdam. So, we have set the initial value of the learning rate to a small value, 1E-05. This initial learning rate value is adapted automatically and effectively by the optimizer RAdam for the learning process. By setting a proper initial value of the learning rate in RAdam [57], which uses the corrected variance of the learning rate according to a specific batching, we aimed to uncover how the learning progress proceeds in terms of immediacy and stability. Relevant details are presented in the following Section IV, 'Experiment, Results and Analysis.'

And we did not adjust the batch size separately. As aforementioned, the reason is that sufficient and similar learning advantages are reflected in the CL (Comparative Learning) method through the specific batching skill. A batch size is only a collection of sequentially injected data to be used for model training. But the batching skill in CL is to control the total amount of data injected into a model at once for concurrent comparison. Because the maximum input is restricted to '4' at once in our model, we could progress the experiment within the extent. So, semantically, the batch size can be seen as set to 1 from a general point of view.

IV. EXPERIMENT, RESULT, AND ANALYSIS

A. RESOURCES AND METRICS

XENTC was implemented using Python 3.7.13 and PyTorch 1.12.1 based on CUDA 11.6. All these experiments are conducted by a server with a CPU of 12 Core AMD Ryzen 9 5900X @ 3.70 GHz and a GPU of NVIDIA GeForce RTX 3080 Ti (12GB Memory).

$$Accuracy = \frac{TP + TN}{(TP + FN + FP + TN)} \tag{7}$$

$$Recall = \frac{TP}{(TP + FN)} \tag{8}$$

$$Precision = \frac{TP}{(TP + FP)} \tag{9}$$

$$F1score = \frac{(2 \times Recall \times Precision)}{(Recall + Precision)} \tag{10}$$

In Equation (7) $\sim$ (10), 'T' means TRUE, 'F' means FALSE, 'P' means POSITIVE, and 'N' means NEGATIVE. In other words, 'TP' and 'TN' are the predicted parts that match the actual values, and 'FP' and 'FN' mean the predicted parts that differ from the actual values.

The accuracy is the percentage of the total that the model classifies correctly. The recall is the percentage of positive values the model classifies as positive. The precision is the proportion of positive values among those classified as positive by the model. The F1 Score is the harmonic average of precision and recall.

And the following Confusion Matrix in Figure 9 is a table for comparing predicted values and actual values to measure prediction performance through training, where 'True label' means actual values and 'Predicted label' means predicted values.

B. PERFORMANCE OF OUR MODEL

Using XENTC for each traffic and application data type, classification Performance values, which consist of accuracy, precision, recall, and F1 score are presented in Figure 9. From our classification results, email, AIM chat, and ICQ applications have appeared to be the most difficult to classify. For reference, similar results were presented in other previous related studies using the same ISCX2016 dataset as ours [32], [54]. AIM Chat and ICQ are used for online chat, and these applications include voice calls, video calls, and chat. These three characteristic forms in AIM Chat and ICQ applications can be clustered with Email in the network classification domain. Because these confusions are intensified by Email, which is added its own characteristic of Email, Email is not easy to distinguish from AIM Chat and ICQ applications and reveals a lower score in the classification result.

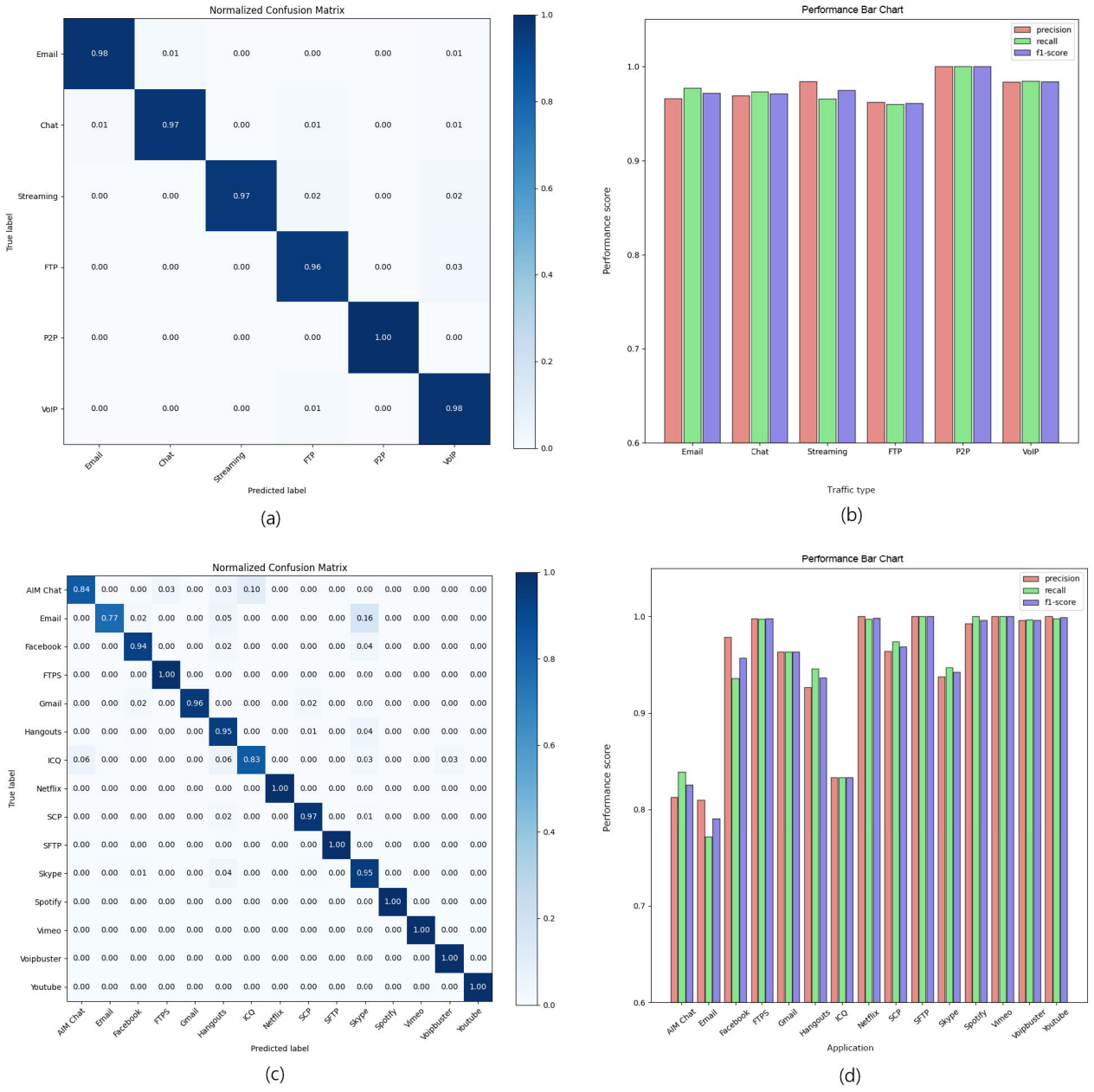

FIGURE 9. Performance values for each class were obtained by applying the test data to the model. (a) & (b) Confusion matrix, precision, recall, and F1 score for 6 classes according to the traffic type. (c) & (d) Confusion matrix, precision, recall, and F1 score for 15 classes according to application.

Figure 10 (a) and (b) are curves illustrating the change process of accuracy and F1 score, respectively. At the end of each training epoch using train data, those values were obtained using test data. The stabilization of the model in this learning process can be observed. Although it was not depicted in Figure 10, the accuracy and F1 score of the model before learning was 6% before the training phase. But, after one epoch, it rapidly improved to 79% accuracy.

While performing the former 10 epochs of the model, it was observed that the performance was improved to 95% accuracy. During the latter 10 epochs after the former 10 epochs, it was observed that the model was stabilized and reached one of the best performances. When the HW mentioned above is used, the time required to infer a single packet of 128 bytes in size is 0.0093 seconds/packet.

In the case of 6 classes, the model was trained using each first packet of 22,254 flows comprising 80% of 27,814 flows. And the training time for 20 epochs was 100 minutes for the model to have a 98.10% F1 score. Considering both model's separate pre-training and the tokenizer's learning using network traffic data was unnecessary, it can be

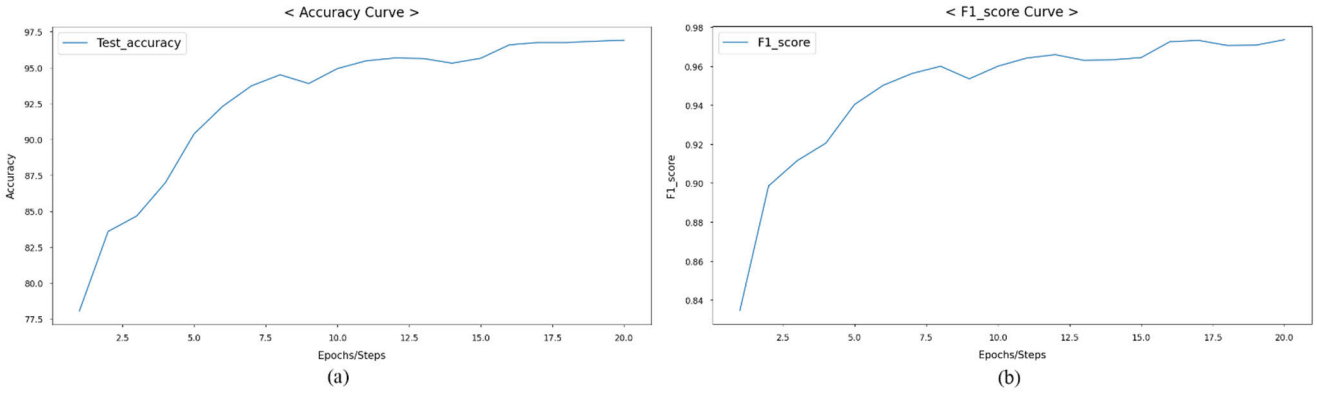

FIGURE 10. (a) Accuracy Curve of the model trained for 20 epochs. (b) F1 score Curve of the model trained for 20 epochs.

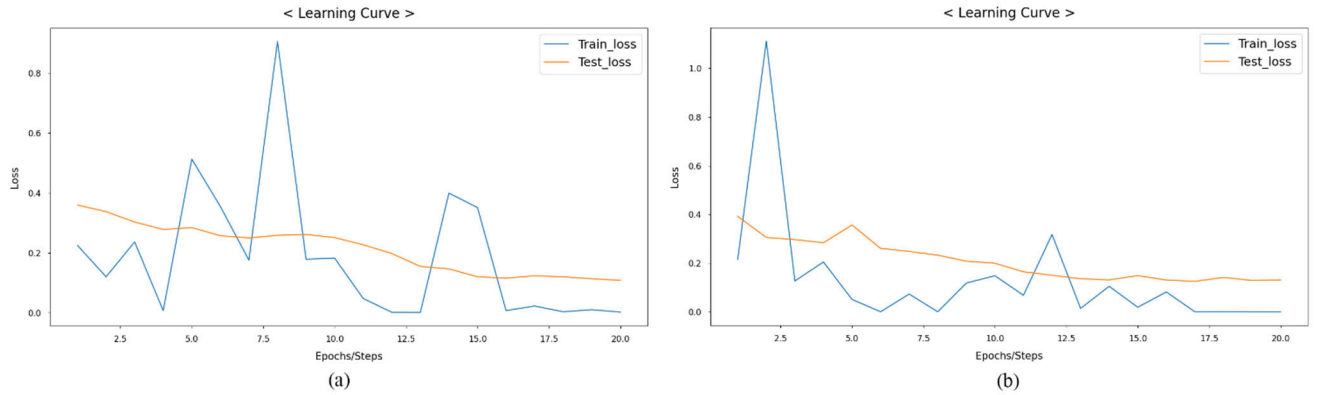

FIGURE 11. (a) Learning Curve of the model, which is not applied with RCW. (b) Learning Curve of the model, which is applied with RCW.

TABLE 2. Comparison of performance values depending on whether or not RCW is applied to the model's loss function.

Class Imbalance Problem	Performance Values			
	Accuracy	Precision	Recall	F1-score
Class unweighted	0.9754	0.9799	0.9744	0.9771
Class weighted	**0.9772**	**0.9830**	**0.9790**	**0.9810**

seen as a short training time to obtain the remarkable performance.

C. CLASS IMBALANCE PROBLEM

When multiclass data has an imbalanced class distribution, an indicator for comparing performances can be the F1 score value because the accuracy value of the model cannot be the proper value reflecting such a situation. The traffic type data of 6 classes were used for the experiment of the relative class weight. In Table 2, two models learned for 20 epochs are distinguished by whether the RCW has been applied. When comparing the F1 score, the model applied with the RCW is observed to be 0.39% better. Furthermore, Figures 10 (a) and (b) reveal the Accuracy and F1 score curves of the model that the relative class weight was applied. The learning has proceeded rapidly and stably, even with a few epochs. Figure 11 (b), which is the learning curve of the model applied with RCW, is more stable than the one of a nonapplied model in Figure 11 (a).

D. COMPARISON WITH OTHER MODEL

Table 3 compares recent approaches that have achieved State of the Art (SOTA) using the ISCXVPN2016 dataset for the past few years with our XENTC. The ISCXVPN2016 dataset was recently reconciled and consisted of 15 applications. So, when comparing ours and other models, it should be considered that such a point is an unavoidable result for comparison. And XENTC was set to utilize CL to obtain one of the best performances. Reference [12] achieved a 1.57% better accuracy than ours for 6 classes, but the result for 12 classes was 12.1% lower. So, it can be limited to use generally for diverse network environments.

Similarly, [54] achieved a 0.66% better F1 score than ours for 17 classes. But the distribution with other performance values is also significantly dispersed. For the classification of a single packet, the BERT base model [18] had a 2.38% higher F1 score than our XENTC. The former used 12 Transformer blocks and 512 bytes of the packet for the test. The latter used 1 Transformer block of the student model and 63 bytes of a packet for the test. Furthermore, the test time of

TABLE 3. A performance comparison of XENTC with other sota models.

Model Comparison	Unit	Input Data		Data Sets (# of Classes)	Performance Values		Test Time (s) (per packet)
		Bytes of Packet	# of Packets		Accuracy	F1-score	
1D-CNN [12]	Session	784	N	ISCX (12 classes)	0.8660	-	0.0024
	Flow	784	N		0.8650	-	-
	Session	784	N	ISCX (6 classes)	0.9830	-	-
	Flow	784	N		0.9860	-	-
1D-CNN [54]	Packet	1500	1	ISCX (17 classes)	0.9758	0.9765	0.0029
				ISCX (12 classes)	0.9329	0.9321	-
Attention + LSTM [32]	Session	1500	10	ISCX (12 classes)	0.9360	-	0.0307
				ISCX (6 classes)	0.9700	-	-
Self-attention [25]	Packet	50	1	ISCX (6 classes)	0.9433	0.9448	0.0011
BERT [18]	Session	512	5	ISCX (17 classes)	0.8519	0.7306	-
	Packet	512	1		0.9962	0.9937	0.1557
	Session	512	5	ISCX (12 classes)	0.9729	0.9733	-
	Packet	512	1		0.9890	0.9890	-
XENTC (Ours)	Session	**63**	**4**	ISCX (15 classes)	**0.9637**	**0.9463**	**0.0151**
	Packet	**63**	**1**		**0.9651**	**0.9699**	**0.0093**
	Session	**63**	**4**	ISCX (6 classes)	**0.9703**	**0.9706**	**0.0151**
	Packet	**63**	**1**		**0.9772**	**0.9810**	**0.0093**

ET-BERT was 0.1557 seconds/packet. It is over 16 times longer than the 9.3 ms of our XENTC. Because each model's performance and test time was different, it was difficult to conclude fragmentarily that our model is more efficient than other work. Consequently, we evaluated that XENTC with the 96.99 and 98.10% F1 score for two types of data, respectively, can be an acceptable and feasible alternative.

E. EFFECT OF COMPARATIVE LEARNING

Table 4 compares both Single Learning (SL) and Comparative Learning (CL) performance values. Compared to the CL, SL uses only a single attribute packet inputted into the model. And then, each CL means two or four multiple attribute packets of a sequential structure are inputted into the model. Those were trained for 10 epochs. As presented in Table 4, the performance values of the CL (Batch size = 2 or 4) applied model are superior to the SL applied one.

Afterward, we proceeded with 10 more additional training epochs and compared each learning method using the F1 score. Table 5 shows that SL achieved a 97.28% F1 score, and the CL (Batch size = 4) achieved a 98.10% F1 score. So, the identical interpretation of the former result, which proceeded only 10 training epochs, can be possible.

Furthermore, another advantage of the CL was revealed. In Table 4 and Table 5, the more multiple attribute packets are put into the model, the less time one training epoch consumes. But the maximum input is restricted to '4' at once.

TABLE 4. A comparing the performance values of models, which proceeded training with progress for 10 epochs according to whether CL was applied or not and the degree of CL.

Learning Method	Performance Values (for 10 epochs)				Training Time (per 1 epoch)
	Accuracy	Precision	Recall	F1-score	
Single Learning (Batch size = 1)	0.9001	0.9398	0.9522	0.9444	13m 55s
Comparative Learning (Batch size = 2)	0.9302	0.9547	0.9485	0.9515	7m 48s
Comparative Learning (Batch size = 4)	**0.9372**	**0.9561**	**0.9573**	**0.9562**	**4m 50s**

TABLE 5. A comparing the performance values of models, which proceeded training with progress for 20 epochs according to whether CL was applied or not and the degree of CL.

Learning Method	Performance Values (for 20 epochs)				Training Time (per 1 epoch)
	Accuracy	Precision	Recall	F1-score	
Single Learning (Batch size = 1)	0.9658	0.9691	0.9763	0.9728	13m 55s
Comparative Learning (Batch size = 2)	0.9678	0.9785	0.9664	0.9723	7m 48s
Comparative Learning (Batch size = 4)	**0.9772**	**0.9830**	**0.9790**	**0.9810**	**4m 50s**

F. MUTUAL ATTENTION RELATIONSHIP

Compared to XENTC, the limitations of [27], which used the 1D-CNN as a model baseline and suggested interpretability using the LIME tool, are as follows. The 1D-CNN converts designated adjacent features to an abbreviated form in the middle process through the convolution mechanism. It could be hard to bring a level of understanding that can be reused by human intervention.

Whereas the byte-level feature can be directly linked to the field unit constituting the packet, respectively, in our XENTC. It makes it possible to be interpreted intuitively by humans. Our XENTC uses the NLP mechanism as a baseline and does not perform tokenizer learning for the above interpretation. If the learning process of the tokenizer included in the PBSM module has been performed, the unique meaning may even be lost according to the mechanism [60] of merging pairs to increase the likelihood of the corpus.

In the Self-attention mechanism, the attention values of each feature represent only the relationship of how much attention is paid to other packet fields. Still, Mutual Attention Relationship (MAR) has a different meaning distinct from the attention given by the original self-attention mechanism. MAR grafts the transformative application of the self-attention mechanism to the characteristics of maintaining the packet structure, giving interpretability to the classification result since we can find factors that acted as important in the overall perspective using the TOP-5 ranking in MAR.

While maintaining the uniqueness of the byte unit field in terms of the packet structure, the process of analyzing the internal structure is as follows. It is related to how TCNN uses 128 bytes of input data for ENTC. Firstly, to minutely investigate the attention correlation of 63 bytes,

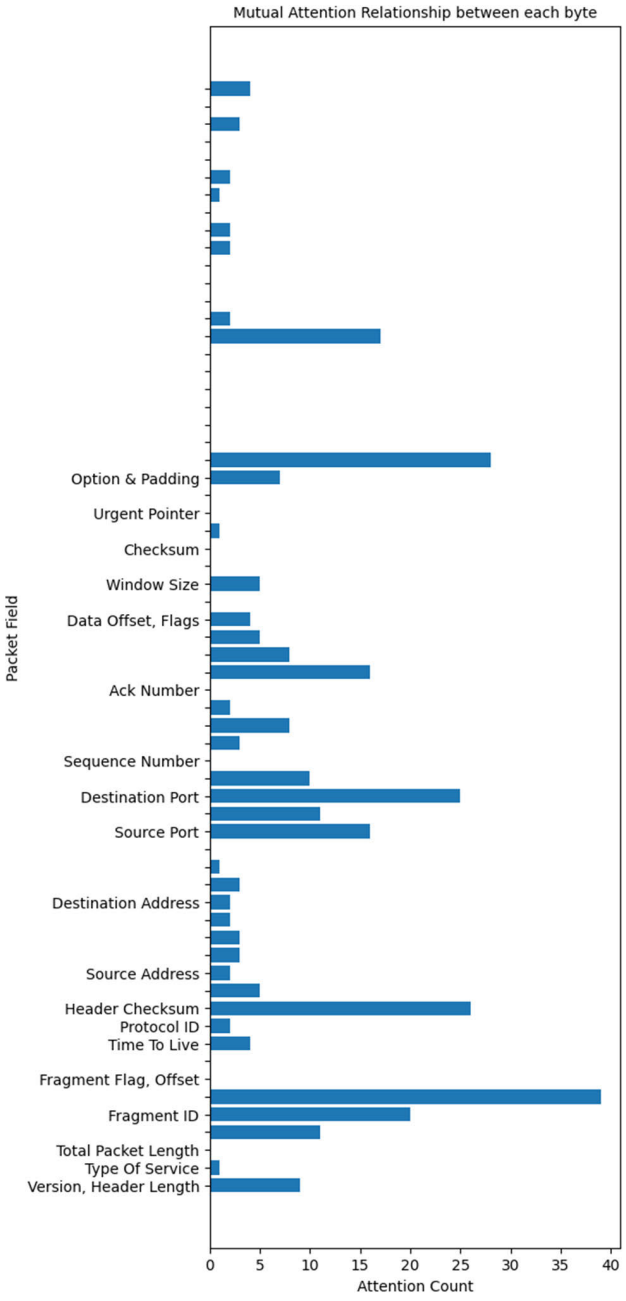

FIGURE 12. Bar chart visualizing important inference factors from the perspective of the packet structure. 'Attention Count' reflects the total of TOP-5 among the number of each byte's mutual attention.

which is the length corresponding to both packet header and partial payload, the attention values of both three special tokens and 62 blank characters should be removed. Secondly, with the remaining 63 bytes, TOP-5 among the attention values, which received attention from both their byte and other bytes for classification, are arranged in descending order. Thirdly, we calculate the total by accumulating these TOP-5 number of times, which is not weighted by ranking. Finally, each obtained total of the byte is illustrated in Figure 12. Each total presents the number implying the element of the MAR corresponding to each byte of the packet structure.

It can be explained clearly using the following example for the meaning of the blanks in Figure 12. The 'destination port' comprises 2 bytes in the packet structure. So, the following blank of the 'destination port' is the last byte of the 'destination port'. As aforementioned, the 'Attention Count' presents the total number of attentions reflecting the TOP-5 among the number of mutual attentions of each packet field. According to the correspondence with the packet structure, Figure 12 can be directly utilized as an explanatory measure. Therefore, we investigated each feature's importance based on it. Among the inputted features to TCNN, we have found particular bytes that play an important role for ENTC regarding the packet structure.

First of all, The IP address, which is padded with a value of 0 through preprocessing, can be received attention slightly. Because, commonly, the BERT attention mechanism gives attention to its own byte and adjacent bytes. So, we analyzed ed that it would be appropriate not to give a significant meaning from the above point of view. Likewise, a low 'Attention Count' of a particular byte can be interpreted as meaningless.

Through mutual attention relationship, the important features of a packet have been revealed in the order of Fragment ID, Header Checksum, Source Port and Destination Port, and Ack Number. Furthermore, the specific byte of the partial payload was observed to have relatively high relationship characteristics. However, the order of the important features is subtly varied whenever the model conducts training. That is because the model's parameters can be changed while training. In addition, the below aspect should be noticed. While the ISCXVPN2016 dataset was collected, the collection-conducted site on the entire network might be unequally distributed. Therefore, if the datasets gathered from various sites are used, the order of the important features used for the classification task can also be changed.

G. VISUALIZATION OF CLASSIFICATION RESULTS

In [61], The high-dimensional data extracted from the model's last layer can be presented in low-dimensional space using the t-distributed Stochastic Neighbor Embedding (t-SNE). The t-SNE is more appropriate for reducing the high-dimensionality of data than the classical linear method, PCA. Using the proposed t-SNE, the classification result of 784-dimensional data belonging to the high dimension has been visualized in 2-dimension to support XAI [62].

Likewise, we utilize the t-SNE for XAI. We used 13,456 pieces of test data for 15 classes of the application classification task. Figure 13 (a) reveals the t-SNE result for the above test data, which is reduced to 2-dimensional data from the 784-dimensional data using the model's parameters before training. Figure 13 (b) shows the visualization result of the t-SNE obtained by putting the same test data into the model trained for 20 epochs.

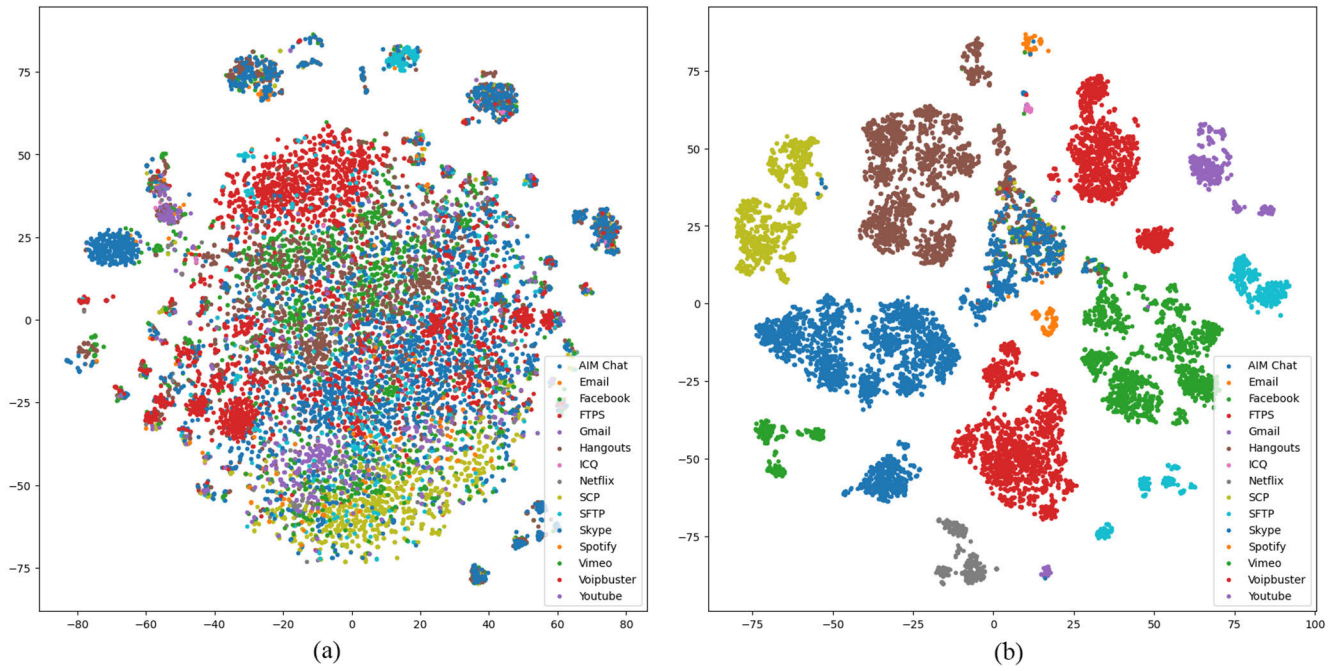

FIGURE 13. (a) A two-dimensional classification visualization of the model before training using t-SNE (b) A two-dimensional classification visualization of the model after training using t-SNE.

In Figure 9 (c) and Figure 9 (d), which were previously presented, the three applications (AIM chat, Email, ICQ) had relatively lower performance values than others. Therefore, even Figure 13 (b) illustrates a non-distinguished form that cannot be placed apart from other applications. However, the difference due to a high drastic dimension reduction must be considered.

V. DISCUSSION

A. RELATIONSHIP BETWEEN THE NUMBER OF FEATURES AND THE MODEL'S PERFORMANCE

Table 6 presents the highest performance values within 10 training epochs with traffic data of 6 classes. It was trained according to 6 types of scenarios for each hyperparameter. As presented in Figure 10 of Section IV, it can be considered that most of the performance values reached a certain amount of upper limitation in the learning curve even within 10 training epochs.

So, comparable experiments of the model were performed for 10 training epochs using a combination of 3 types of partial input packets and two types of weight decay as the experimental setting. In Table 6, looking into scenarios 1, 3, and 5 selectively, the performance values were compared while progressively increasing the number of bytes per packet by 63, 126, and 189. It is observed that the accuracy, precision, recall, and F1 score have not continuously increased even when the number of variables used to fit the model was increased.

There was a reverse instance where the above case could not be worked in the CNN and Resnet [63]. The model's learning effect was improved as the number of variables used to proceed with the model's learning increased in their research. But, through our experiment, we observed that additional variables do not necessarily increase the performance of the DL model.

Usually, Variables that increase the model's performance through training can be defined as Response Variables. And variables that can cause learning confusion to be defined as Noise Variables. However, considering the two cases above means that the effect of specific variables can differ according to the model or circumstances. Therefore, the particular variables do not always work as Noise Variables.

In the case of confining both interpretation and application extent related to the experiment result of XENTC to a subset of the real possibility, the following review can be reasonable

TABLE 6. 6 scenarios using a combination of 3 types of partial packets and 2 types of weight decay. Result of performance values and the required time for both training and inference latency according to 6 scenarios.

Scenario #	Hyper Parameters		Performance Values				Training Time (per 1 epoch)	Inference Latency
	Weight Decay	Bytes of Packet (Max Length)	Accuracy	Precision	Recall	F1-score		
1	0	63 (128)	0.9372	0.9561	0.9573	0.9562	4m 50s	13ms
2	0.01	63 (128)	0.8224	0.8756	0.9155	0.8861	4m 37s	-
3	0	126 (256)	0.9333	0.9502	0.9502	0.9504	7m 21s	20ms
4	0.01	126 (256)	0.8089	0.8607	0.9045	0.8617	7m 06s	-
5	0	189 (384)	0.9160	0.9355	0.9438	0.9393	11m 01s	30ms
6	0.01	189 (384)	0.8152	0.8575	0.8630	0.8539	10m 52s	-

TABLE 7. A comparison of each model's performance according to scaling level of dimension normalization.

Dimension Normalization	Performance Values			
	Accuracy	Precision	Recall	F1-score
None	0.7751	0.7128	0.8006	0.7303
64	0.8231	0.8704	0.9008	0.8788
128	0.8797	0.9235	0.9435	0.9293
256	**0.9372**	**0.9561**	**0.9573**	**0.9562**

in determining the applicability of our experiment result. As presented in Table 6, we regarded the payload in the network traffic as a kind of Noise Variable. We made it increase in the order of 23 (scenario 1), 86 (scenario 3), and 149 (scenario 5). It was observed that the model's performance values did not improve even though the Noise Variables increased. Instead, it was revealed that adding the Noise Variable can deteriorate the model's performance. Due to a load of training time, our six scenarios were based on the model trained for 10 epochs. Because of the slight quantitative difference in performance values in scenario 3, we proceeded with additional training for up to 20 epochs to eliminate curiosity about whether training is sufficient. As a result, the accuracy of scenario 3, 96.91%, has been even lower than the accuracy of scenario 1, 97.72%. This can even be interpreted as an example of the curse of dimensionality [64].

In terms of services that provide traffic classification based on limited computing resources, a model that provides a high classification performance within a limited time can be effective on-site. As shown in Table 6, as the number of bytes of the input packet increases, the training time and test time also increase proportionally. Accordingly, Our XENTC can be compared with another approach [18] that uses 512 bytes of a single packet, including the whole payload, as input data. In conclusion, if a difference in the model's performance is not remarkable, it should not be overlooked that the Noise Variable may be a factor to be considered in the data planning stage according to the model baseline.

B. INTERVENTION AS TO FEATURE SCALE

When RAdam is used for optimization, the model can use regularization according to the weight decay value deciding the level of L2 regularization of the loss function [65]. Therefore, how the weight decay value is set as 0 or another value correlates with a suppression effect of overfitting.

$$C = C_0 + \frac{\lambda}{2n}\sum_{\omega}^{n}\omega^2 \tag{11}$$

$$\omega \to \omega - \frac{\partial C_0}{\partial \omega} - \frac{\eta\lambda}{n}\omega = \left(1 - \frac{\eta\lambda}{n}\right)\omega - \eta\frac{\partial C_0}{\partial \omega} \tag{12}$$

It can be figured out by looking through the L2 regularization method. A general formula of the L2 regularization is given in equation (11). C is the regularized loss function in that the regularized term is added to the original loss. C_0 is the original loss function. n is the number of training data, and ω is weight. λ is a constant as a regularization variable. We can acquire equation (12) through a partial derivative for ω on equation (11). At this time, it is progressed to lessen the value, and we treat it as weight decay. Therefore, in the L2 regularization, we can ensure that the weight decay plays a role in preventing the specific weights from growing abnormally and impacting the model's learning remarkably.

In Table 6, we tested additional scenarios 2, 4, and 6 by setting the weight decay to 0.01. Similar to scenarios 1, 3, and 5, it was observed that performance values did not increase even when the number of variables used in the model increased in scenarios 2, 4, and 6. Even though the weight decay value was not 0 but 0.01, the model's performance values did not improve. These experiments need to be considered for affiliation with the following experiment.

Table 7 is the highest value among performance values within 10 training epochs, the same as the previous experiment. To compare with the L2 regularization effect, the weight decay was fixed to 0 while the experiment was performed. The level of dimension normalization can be seen as the same as setting the recognizable number of words in the NLP model. The level dimension normalization, a hyperparameter of the model, is designated to 256 because each packet byte was converted to integers between 0 and 255. This was used as a standard of performance comparison with each model, which fixed the hyperparameter in order of 128, 64, and None, respectively. The result of the experiment is presented in Table 6. The lower the level of dimension normalization was, the lower the model's performance was.

As previously stated, Table 6, related to L2 regularization, can be interpreted in connection with Table 7. Table 7 shows that the normalization of 256 dimensions can be understood to prevent the model from underfitting. Because it means adjusting the feature scale to match the byte level of the packet. Regarding the same perspective, a weight decay value to a specific value, not 0, also brought the same effect presented in Table 6. Therefore, we can derive the following semantics from Table 6 and 7. Assigning a biased numeric to a weight decay value can cause the same underfitting effect as adjusting the feature scale, which is controlled by the level of dimension normalization, to a value out of the deviant level. It means that the above two hyperparameters can bring a similar mainspring even to the effect of the model's training [27].

In conclusion, it should be noted in the experiment that the above semantic correlation needs to be considered while adjusting the hyperparameters of the NLP model for adaptation to the ENTC task.

VI. CONCLUSION

A. SUMMARY

In this paper, by applying a packet header and a partial payload to a lightweight model in the ENTC task, we showed an approach that derives remarkable results in processing

time. A tailored model of BERT, a large-scale model, does not have much meaning in the language domain. However, in the ENTC area, where network resources are limited, the lightweight model showing remarkable performance could have considerable meaning. Also, to reflect the natural environment between experiments, we used a public dataset with class imbalance characteristics as it is. But we applied RCW to deal with the imbalance and observed the advantages on performances. A model applied RCW revealed better learning progress and performance values than not.

And we designed to compare and classify four multi-attribute sequences at once by extending and applying the learning of differences between different sentences among the tasks of the BERT model. A model in that CL was applied revealed better performance than not.

Our XENTC is based on BERT, which aims for NLP tasks and uses a sequential property of the input features. Our mutual attention relationship between each byte is established on this common characteristic. From the design point of view of the XENTC model, we treat each byte unit of the packet as the same as an individual word in NLP. Therefore, the mutual attention relationship of the data input to the model was counted by a byte unit. So, through a mechanism that can check the ranking, an attempt was made to expand into the XAI area that humans can intuitively understand through direct linkage with the packet structure. By extension, it contains the possibility of additional research for the evolutive preprocess using our XAI mechanism. In addition, the clustering of data representing the classification results of the model before training and the model after training was visualized using t-SNE.

The explanation of the inference process of the model was secured while maintaining a remarkable and acceptable accuracy to the ENTC problems. Accordingly, when comparing the performance of XENTC with recent SOTA methods, we analyzed that our model would be a reasonable alternative.

B. FUTURE WORK

In the future, we plan to develop the ENTC model from the following four perspectives by extending the model in this paper.

Firstly, it is necessary to study redesign from the packet structure point of view. Because a tokenizer based on not byte unit but each packet field can be developed. Secondly, the NLP method [13], [25] and the CV method [18], [63] are possible to process for packets and flows in ENTC problems. Therefore, it is necessary to conduct research that compares the semantics of the two approaches in various ways according to the input unit in more detail. Thirdly, unsupervised learning through stream mining can be a better candidate for the training method. So, we would like to compare it with supervised learning. Fourthly, it is necessary to compare the preprocessing mechanisms with each other. For performance improvement of the model semantically by applying the relatable XAI viewpoint. In other words, it means finding the optimal preprocessing method by circling three sequential flows in order of different preprocessing methods, performances, and inter-byte relationship comparison.

REFERENCES

[1] 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, 4th Quart., 2008, doi: 10.1109/SURV.2008.080406.

[2] P. Velan, M. Cermák, P. Celeda, and M. Drasar, "A survey of methods for encrypted traffic classification and analysis," *Int. J. Netw. Manag.*, vol. 25, no. 5, pp. 355–374, Sep. 2015, doi: 10.1002/nem.1901.

[3] F. Gringoli, L. Salgarelli, M. Dusi, N. Cascarano, F. Risso, and K. C. Claffy, "GT: Picking up the truth from the ground for internet traffic," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 5, pp. 12–18, Oct. 2009, doi: 10.1145/1629607.1629610.

[4] J. Ma, K. Levchenko, C. Kreibich, S. Savage, and G. M. Voelker, "Unexpected means of protocol inference," in *Proc. 6th ACM SIGCOMM Conf. Internet Meas.*, Oct. 2006, pp. 313–326, doi: 10.1145/1177080.1177123.

[5] A. Dainotti, A. Pescape, and K. C. Claffy, "Issues and future directions in traffic classification," *IEEE Netw.*, vol. 26, no. 1, pp. 35–40, Jan. 2012, doi: 10.1109/MNET.2012.6135854.

[6] 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.

[7] A. Vladutu, D. Comaneci, and C. Dobre, "Internet traffic classification based on flows' statistical properties with machine learning," *Int. J. Netw. Manag.*, vol. 27, no. 3, p. e1929, May 2017, doi: 10.1002/nem.1929.

[8] Y.-N. Dong, J.-J. Zhao, and J. Jin, "Novel feature selection and classification of internet video traffic based on a hierarchical scheme," *Comput. Netw.*, vol. 119, pp. 102–111, Jun. 2017, doi: 10.1016/j.comnet.2017.03.019.

[9] R. B. Yanai, M. Langberg, D. Peleg, and L. Roditty, "Realtime classification for encrypted traffic," in *Experimental Algorithms* (Lecture Notes in Computer Science), vol. 6049, P. Festa, Ed. Berlin, Germany: Springer, 2010, pp. 373–385, doi: 10.1007/978-3-642-13193-6_32.

[10] T. T. T. Nguyen, G. Armitage, P. Branch, and S. Zander, "Timely and continuous machine-learning-based classification for interactive IP traffic," *IEEE/ACM Trans. Netw.*, vol. 20, no. 6, pp. 1880–1894, Dec. 2012, doi: 10.1109/TNET.2012.2187305.

[11] X. Zhu, N. Shu, H. Wang, and T. Wu, "A distributed traffic classification model based on federated learning," in *Proc. 7th Int. Conf. Big Data Comput. Commun. (BigCom)*, Aug. 2021, pp. 75–81, doi: 10.1109/BigCom53800.2021.00022.

[12] 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.

[13] 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.

[14] S. Rezaei and X. Liu, "Multitask learning for network traffic classification," in *Proc. 29th Int. Conf. Comput. Commun. Netw. (ICCCN)*, Aug. 2020, pp. 1–9, doi: 10.1109/ICCCN49398.2020.9209652.

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

[16] C. Zhang, C. An, J. H. Wang, Z. Zhao, T. Yu, and J. Wang, "SAFSN: A self-attention based neural network for encrypted mobile traffic classification," in *Proc. IEEE Int. Conf. Internet Things (iThings), IEEE Green Comput. Commun. (GreenCom), IEEE Cyber, Phys. Social Comput. (CPSCom), IEEE Smart Data (SmartData), IEEE Congr. Cybermatics (Cybermatics)*, Dec. 2021, pp. 330–337, doi: 10.1109/iThings-GreenCom-CPSCom-SmartData-Cybermatics53846.2021.00060.

[17] 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.

[18] 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*.

[19] 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 Manag.*, vol. 18, no. 4, pp. 4225–4246, Dec. 2021, doi: 10.1109/TNSM.2021.3098157.

[20] Z. Meng, M. Wang, J. Bai, M. Xu, H. Mao, and H. Hu, "Interpreting deep learning-based networking systems," in *Proc. Annu. Conf. ACM Special Interest Group Data Commun. Appl., Technol., Architectures, Protocols Comput. Commun.*, Jul. 2020, pp. 154–171, doi: 10.1145/3387514.3405859.

[21] 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.

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

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

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

[25] 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.

[26] M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why should I trust you?' Explaining the predictions of any classifier," 2016, *arXiv:1602.04938*.

[27] Y. Wang, X. Yun, Y. Zhang, C. Zhao, and X. Liu, "A multi-scale feature attention approach to network traffic classification and its model explanation," *IEEE Trans. Netw. Service Manag.*, vol. 19, no. 2, pp. 875–889, Jun. 2022, doi: 10.1109/TNSM.2022.3149933.

[28] D. Upadhyay, J. Manero, M. Zaman, and S. Sampalli, "Gradient boosting feature selection with machine learning classifiers for intrusion detection on power grids," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 1, pp. 1104–1116, Mar. 2021, doi: 10.1109/TNSM.2020.3032618.

[29] C. Hsieh, Y. Chen, W. Beh, and A. A. Wu, "Feature selection framework for XGBoost based on electrodermal activity in stress detection," in *Proc. IEEE Int. Workshop Signal Process. Syst. (SiPS)*, Oct. 2019, pp. 330–335, doi: 10.1109/SiPS47522.2019.9020321.

[30] Y. Hua, "An efficient traffic classification scheme using embedded feature selection and LightGBM," in *Proc. Inf. Commun. Technol. Conf. (ICTC)*, May 2020, pp. 125–130, doi: 10.1109/ICTC49638.2020.9123302.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] C. Liu, L. He, G. Xiong, Z. Cao, and Z. Li, "FS-Net: A flow sequence network for encrypted traffic classification," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, Apr. 2019, pp. 1171–1179, doi: 10.1109/INFOCOM.2019.8737507.

[36] Y. Chen, T. Zang, Y. Zhang, Y. Zhou, and Y. Wang, "Rethinking encrypted traffic classification: A multi-attribute associated fingerprint approach," in *Proc. IEEE 27th Int. Conf. Netw. Protocols (ICNP)*, Oct. 2019, pp. 1–11, doi: 10.1109/ICNP.2019.8888043.

[37] K.-S. Shim, J.-H. Ham, B. D. Sija, and M.-S. Kim, "Application traffic classification using payload size sequence signature," *Int. J. Netw. Manag.*, vol. 27, no. 5, p. e1981, Sep. 2017, doi: 10.1002/nem.1981.

[38] 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.

[39] N. Williams, S. Zander, and G. Armitage, "A preliminary performance comparison of five machine learning algorithms for practical IP traffic flow classification," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 5, pp. 5–16, Oct. 2006, doi: 10.1145/1163593.1163596.

[40] E. Akleman, "Deep learning," *Computer*, vol. 53, no. 9, p. 17, Sep. 2020, doi: 10.1109/MC.2020.3004171.

[41] G. E. Hinton, S. Osindero, and Y. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006, doi: 10.1162/neco.2006.18.7.1527.

[42] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015, *arXiv:1512.03385*.

[43] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2261–2269, doi: 10.1109/CVPR.2017.243.

[44] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," 2015, *arXiv:1510.00149*.

[45] H. Hagras, "Toward human-understandable, explainable AI," *Computer*, vol. 51, no. 9, pp. 28–36, Sep. 2018, doi: 10.1109/MC.2018.3620965.

[46] F. Xu, H. Uszkoreit, Y. Du, W. Fan, D. Zhao, and J. Zhu, "Explainable AI: A brief survey on history, research areas, approaches and challenges," in *Proc. CCF Int. Conf. Natural Lang. Process. Chin. Comput.*, 2019, pp. 563–574, doi: 10.1007/978-3-030-32236-6_51.

[47] 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.

[48] 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.

[49] 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. Secur. Privacy*, 2016, pp. 407–414, doi: 10.5220/0005740704070414.

[50] R. Longadge and S. Dongre, "Class imbalance problem in data mining review," May 2013, doi: 10.48550/arXiv.1305.1707.

[51] H. Li, "Traffic classification algorithm using CNN and multi-head attention mechanism for representation learning," *J. Phys., Conf.*, vol. 2258, no. 1, Apr. 2022, Art. no. 012001, doi: 10.1088/1742-6596/2258/1/012001.

[52] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Proc. Adv. Neural Inf. Process. Syst.* Cambridge, MA, USA: MIT Press, 2006, pp. 1–8. [Online]. Available: https://proceedings.neurips.cc/paper/2006/hash/5da713a690c067105aeb2fae32403405-Abstract.html

[53] G. Zhong, L.-N. Wang, and J. Dong, "An overview on data representation learning: From traditional feature learning to recent deep learning," 2016, *arXiv:1611.08331*.

[54] M. Lotfollahi, R. S. H. Zade, M. J. Siavoshani, and M. Saberian, "Deep packet: A novel approach for encrypted traffic classification using deep learning," 2017, *arXiv:1709.02656*.

[55] S. Rezaei and X. Liu, "How to achieve high classification accuracy with just a few labels: A semi-supervised approach using sampled packets," 2018, *arXiv:1812.09761*.

[56] M. S. Sharif and M. Moein, "An effective cost-sensitive convolutional neural network for network traffic classification," in *Proc. Int. Conf. Innov. Intell. Informat., Comput., Technol. (ICT)*, Sep. 2021, pp. 40–45, doi: 10.1109/3ICT53449.2021.9581789.

[57] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han, "On the variance of the adaptive learning rate and beyond," 2019, *arXiv:1908.03265*.

[58] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[59] I. Kandel and M. Castelli, "The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset," *ICT Exp.*, vol. 6, no. 4, pp. 312–315, Dec. 2020, doi: 10.1016/j.icte.2020.04.010.

[60] T. Kudo, "Subword regularization: Improving neural network translation models with multiple subword candidates," 2018, *arXiv:1804.10959*.

[61] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.

[62] C. Beliard, A. Finamore, and D. Rossi, "Opening the deep Pandora box: Explainable traffic classification," in *Proc. IEEE INFOCOM Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Jul. 2020, pp. 1292–1293, doi: 10.1109/INFOCOMWKSHPS50562.2020.9162704.
[63] H. Lim, J. Kim, J. Heo, K. Kim, Y. Hong, and Y. Han, "Packet-based network traffic classification using deep learning," in *Proc. Int. Conf. Artif. Intell. Inf. Commun. (ICAIIC)*, Feb. 2019, pp. 46–51, doi: 10.1109/ICAIIC.2019.8669045.
[64] F. Sohil, M. U. Sohali, and J. Shabbir, "An introduction to statistical learning with applications in R," *Stat. Theory Relat. Fields*, vol. 6, no. 1, p. 87, Jan. 2022, doi: 10.1080/24754269.2021.1980261.
[65] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 2017, *arXiv:1711.05101*.

CHANG-YUI SHIN received the B.S. degree in operating analysis from the Korea Military Academy, Seoul, in 2003, the M.S. degree in electronic computer engineering from Korea University, Seoul, in 2007, and the Ph.D. degree from Korea University, Sejong. Since being commissioned as an Army Officer, in 2003, his service department is Signal in Korea Army. At the time, he had researched the field of mobile ad hoc networks. After that, he became interested in feasible research, such as weapon system planning, interoperability, development quality management, and actual operation while working in various organizations. He researches internet traffic classification, network management, and AI during the Ph.D. degree. He works at 'Defense Agency for Technology and Quality' while simultaneously researching in 'Korea University'.

JEE-TAE PARK was born in Busan, South Korea, in 1993. He received the B.S. degree in computer and information science from Korea University, South Korea, in 2017, where he is currently pursuing the Ph.D. degree (integrated program). His research interests include internet traffic classification, network management, and internet security.

UI-JUN BAEK was born in Seoul, South Korea, in 1993. He received the B.S. degree in computer and information science from Korea University, South Korea, in 2018, where he is currently pursuing the Ph.D. degree (integrated program). His research interests include blockchain transaction monitoring, network management, and internet 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 with the Department of Electrical and Computer Engineering, University of Toronto, Canada. He joined Korea University, South Korea, in 2006, where he is currently a Full Professor with the Department of Computer Convergence Software. His research interests include internet traffic monitoring and analysis, service and network management, future internet, and internet security.