Article

The Ensemble of Text Convolutional Neural Networks and Multi-Head Attention Layers for Classifying Threats in Network Packets

Hyeon Min Kim and Young Yoon *

Posted Date: 12 September 2023

doi: 10.20944/preprints202309.0686.v1

Keywords: Network Threat Classification; Multi-Head Attention; Ensemble Machine Learning; Packet Payload Processing

Article

The Ensemble of Text Convolutional Neural Networks and Multi-Head Attention Layers for Classifying Threats in Network Packets

Hyeon Min Kim [1,†] and Young Yoon [2,3,*,‡]

1 Hana Financial Investment ; hyeonminkim@hanafn.com
2 Neouly Incororated; young.yoon@hongik.ac.kr
3 Department of Computer Engineering, Hongik University; young.yoon@hongik.ac.kr
* Correspondence: young.yoon@hongik.ac.kr
† 82, Uisadang-daero, Yeongdeungpo-gu, Seoul, Republic of Korea.
‡ 94, Wausan-ro, Mapo-gu, Seoul, Republic of Korea.

Abstract: Using traditional methods based on detection rules written by human security experts, there is a significant challenge in accurately detecting many network threats that are increasingly becoming sophisticated. In order to deal with the limitation of the traditional methods, network threat detection techniques utilizing artificial intelligence technologies such as machine learning are being extensively studied. Research has also been conducted on analyzing various string patterns in network packet payloads through natural language processing techniques to detect attack intent. However, due to the nature of packet payloads containing binary data as well as text data, a new approach is needed that goes beyond typical natural language processing techniques. In this paper, we study a token extraction method optimized for payloads using n-gram and byte pair encoding techniques. Furthermore, we generate embedding vectors that can understand the context of the packet payload using algorithms such as Word2Vec and FastText. We also compute the embedding of various header data associated with packets, such as IP addresses and ports. Given the combination of these features, we ensemble a text 1D-CNN and a multi-head attention network in a novel fashion. We validated the effectiveness of our classification technique with the CICIDS2017 open dataset and over half a million data collected by The Education Cyber Security Center (ECSC) currently operating in South Korea. The proposed model showed remarkable progress compared to previous studies, demonstrating a highly accurate classification performance with an F1-Score of 0.998. Our model can also preprocess and classify 150,000 network threats per minute to help security agents in the field divert their time to analyzing more complex attack patterns.

Keywords: network threat classification; multi-head attention; ensemble machine learning; packet payload processing

1. Introduction

Attack surface is increasing as the mobile devices proliferate and they interact with various IoT and Cloud platforms [1–3]. The increased attack surface attracts diverse and sophisticated cyber threats that are getting more challenging to detect. For example, in July 2021, hundreds of sites in the United States were affected by ransomware attacks due to the sneaky exploitation of Kaseya's IT management solution[4]. In November of the same year, connected wall pads working as smart home hubs in a Korean apartment complex were hacked for the first time [5]. Similar to the prior incidents of IP cameras being hacked, multiple households suffered leakage of a large amount of private information. In December 2021, there were infamous cases of malware propagation exploiting the Apache Log4j vulnerability [6]. These incidents show that cyber threats continue to occur in various ways as the ubiquity of online services and applications grows.

Methods such as secure coding, vulnerability diagnosis, intrusion detection systems (IDS), and firewalls have been applied to prevent cyber attacks. IDS, in particular, employs techniques for analyzing the contents of network flows or packets to detect and notify malicious intent from network traffic [7,8]. IDS is commonly equipped with detection rules written in widely-adopted Snort format to define known threats [9]. However, as cyber-attacks evolve, attackers have devised various evasion methods [10]. Rule-based approaches are effective against known threats. However, detecting new types of unknown attacks is difficult. To solve this problem, detection rules must be constantly updated upon the appearance of new attack types. Human security experts must take a significant amount of time to analyze network data to understand the mechanism of reported threats. Also, the quality of detection rules heavily relies on the level of knowledge and experience of the security experts who author those rules. Hence, such a time-consuming, labor-intensive, and manual approach to devising detection rules is ineffective, especially in a situation with vast threat data pouring in.

Many studies have been conducted on detecting attacks by identifying abnormal characteristics from formatted data blocks called packets transmitted through a network. A packet typically contains various information, such as a payload as well as a header specifying IPs and ports of sources and destinations. Recently, many studies have used deep learning techniques to model packets for attack detection. In particular, the payload of a packet may contain scripts, queries, or natural language texts for attacks. Recent studies applied natural language processing (NLP) techniques based on deep learning to better understand the textual data in payloads [11,12]. NLP techniques have been applied in various fields, such as content summarization, translation, and text classification of spam mail and news articles. Generative AI services for Q&A, such as ChatGPT, have recently emerged. However, payload data deals with information such as scripts composed in structures different from ordinary natural language texts. Therefore, we cannot blindly use pre-trained language models such as BERT [13] and Reoberta [14] to understand the intent in a given payload.

We have developed new natural language processing techniques applicable specifically to packet payloads and header data to address the problems mentioned above. We propose a *Conv1D-Multi-Head Attention-Ensemble* (CMAE) model that can accurately and instantly detect network threats. We trained the CMAE model with input features extracted from payloads and header data in various ways. Specifically, the payload data is tokenized into n-grams, and the word embeddings are used to learn the context information. Then, 1D-CNN (Convolutional Neural Network) model yields and summarizes features. The intermediate output of the 1D-CNN model is injected into a *multi-head attention layer* that computes attention weights of each payload token to learn the structural features leading to threats. We also trained multi-kernel 1D-CNN to discover the association between payloads, header data and types of threats. The models trained with payload and header data are combined with an ensemble method using an averaging layer and a self-attention layer. The final feature vector is entered to a softmax function that picks the likely type of threats.

We evaluated our novel method with CICIDS2017 open dataset [15] and real-world network datasets collected by the Education Cyber Security Center (ECSC) in South Korea. Since January 2022, ECSC has been using our technique to detect and classify approximately 1.2 million network threats daily.

The major contributions of this study can be summarized as follows:

First, we propose a composite method of extracting useful features from payloads and header data to improve the performance of threat detection and classification. Specifically, tokenization, unnecessary token removal, word embedding, integer encoding, and padding were carefully devised and tested.

Secondly, we crafted separate deep learning models suitable for payload and header data learning. We then devised a unique ensemble layer that averages the output vector of the different models, followed by an attention layer. Our ensemble mechanism took advantage of the separate models and achieved highly accurate threat classification results. The experimental results obtained using the CICIDS2017 dataset showed that the CMAE model achieved an f1-score of 0.99 in classifying threat types. We have observed the same level of accuracy with the ECSC dataset, which shows that our approach is uniquely proven highly efficient in a real-world security monitoring system. In contrast to the previous studies, we achieved the high accuracy without sampling and balancing the proportion of the training set.

The rest of the paper is structured as follows: Section 2 presents related works; Section 3 explains the mechanism of our approach in detail; Section 4 provides evaluation results; Section 5 discusses limitations of the CMAE-based threat monitoring and directions for future works; and finally we conclude in Section 6.

2. Related work

In this section, we put our work in the context of previous studies using detection rules, machine learning and NLP techniques.

2.1. The Rule-Based

The most widely adopted practice is composing threat detection rules according to Snort [16]. Many previous studies have come up with rules for detecting network threats. ATLANTIC [17] utilized flow-based information and detected the threat of malicious traffic by calculating the entropy deviation of traffic flow data. In [18], entropy and the characteristics of multiple traffic were also utilized based on network flow information. [19] is a flow-based system that detects abnormal activity by applying the chi-squared technique to IP flow characteristics. Some studies examined packet payloads. In PAYL [20], 256 features were extracted for each 1-byte of payload. The extracted 1-byte payload calculates the frequency distribution within the payload flowing through each host and port and calculates the mean and standard deviation. Then, it calculates similarity using the Mahalanobis distance and detects an attack if it exceeds a specific threshold value. In [21], abnormal network flow showing the Web of Things application logic violation was detected using the RETE-based rule engine. [22] represented features of a payload segmented in two-gram tokens. In [23], association rule mining was conducted on network flows and their features using the FP-Growth algorithm to discover abnormal activities suspected to be port scans, brute-force attacks, and denial of services (DoS).

These rule-based methods have much room to be improved in terms of the accurate detection of network threats. The process of extracting features is excessively manual and complex. The criteria for abnormal incidents were based on thresholds set laboriously by trial and error. The effectiveness of these methods depends on the knowledge of security experts and has a large variability. These methods inevitably force security experts to revise the rules whenever new attack patterns emerge.

2.2. The Usage of Machine Learning

Machine learning techniques that can automatically learn data characteristics are rapidly evolving in various ways to overcome the limitations of traditional methods. For network threat detection research, machine learning techniques such as support vector machines [24], support vector data description in concert with DBSCAN [25], Naive-Bayesian [26,27], and random forest [28,29] have been utilized.

Recently, deep learning techniques have significantly advanced in various fields, as evidenced by numerous studies [30–32]. For network threat detection studies, [33] detected anomalous traffic using Spatio-Temporal information modeling based on the C-LSTM method. Other studies have employed CNNs (Convolutional Neural Networks) to learn attacks from traffic data converted into images [34,35]. [36,37] utilized DNNs (Deep Neural Networks) to detect anomalies in IoT devices and network traffic data. [38] used CNN and LSTM to learn the threat patterns inside payload data. Various deep learning techniques have been trained with CICIDS2017 dataset [39] generated during simulated network attacks. In particular, RTIDS [40] used the Transformer network to model the threats captured in CICIDS datasets.

Class imbalance is a significant issue as threat data is typically much less abundant than normal data. Such an issue makes supervised learning methods susceptible to overfitting and bias problems.

Many unsupervised learning-based studies have been conducted to address such an issue. Noteworthy unsupervised learning approaches employed autoencoder methods [41,42].

2.3. The Employment of NLP Techniques

Research that employs deep learning from the NLP perspective using payload data is also relevant to our approach. In [43], word embeddings and LSTM models were applied using packet-based data. The field information of packet headers was merged into one sentence for learning. Specifically, information on packet data version, flags, protocols, and source and destination IPs and ports were combined to form a single sentence. Given the input data in sentences, word embeddings were used to extract the meaning of the packet. The LSTM model was used to learn temporal information between fields in the packet header. As a result, a promising performance was shown with the ISCX-IDS-2012, USTC-TFC-2016, Mirai-RGU, and Mirai-CCU datasets.

In TR-IDS [44], word embeddings and Text-CNN were applied using payload data, and then classification was done using random forests along with the extracted statistical features of the flow data. Given the ISCX2012 dataset, TR-IDS classified infiltration, bruteforce attack against SSH, DDoS, and HTTP-DoS with accuracy as high as 99.13%.

In [45], a block sequence structure was created using packet payload data, and a detection model was designed based on LSTM and CNN models and multi-head self attention mechanism. [45] achieved from 97% to 99% accuracy in detecting web attacks.

The above studies show that high accuracy can be achieved when applying deep learning models from an NLP perspective using payload data. We propose a different strategy for finding the optimal features in payload data by applying techniques such as n-gram tokenization, removing unnecessary tokens based on TF-IDF, and word embedding. We apply 1D-CNN and multi-head attention layers to the payload data. We used separate multi-kernel 1D-CNN models for payload and header data. Instead of pinning it down to a single model, we *ensembled* the different models whose output vectors are merged into a latent attention vector for classifying network threats. This comprehensive deep learning structure was designed to deal with complex real-world threat patterns not seen in the typical simulated threat datasets used by the previous works. Our ensemble model is now an integral part of the ECSC in charge of monitoring 1.2 million threat cases daily from educational institutions in South Korea. Our model classifies threat with an F1-score of 0.998. Additionally, threats identified with over 99% confidence are automatically registered as attack incidents so that security agents' burden can be significantly relieved. Therefore, the practical value of our solution has been proven in the field.

A comparison of our approach with various existing methodologies is summarized in Table 1.

Table 1. Comparison between NLP-based Network Threat Detection Methods

Research Works	Data		Pre-Processing			Word Embedding		Modeling Method				Real Data Tested
	Payload	Header	n-gram	BPE	Tf-Idf	Word2Vec	FastText	Text 1D-CNN	Sequential Models	Attention	Ensemble	
[45]	O		Block Sequence Construction			Block Embedding			O	O		
[46]	O			O			O		O	O		
[43]		O				Linguistic Embedding			O			
[44]	O	O	O			O		O				
Our Method	O	O	O	O	O	O	O	O	O	O	O	O

3. Methodology

In this section, we explain our methodology in detail.

3.1. An Overview of Conv1D Multi-Head Attention Ensemble

We follow the modeling structure, Conv1D Multi-Head Attention Ensemble (CMAE), as shown in Figure 1. We tokenize the payload and header data and apply embedding techniques to generate input data. The CMAE model receives a total of three inputs for training. An encoded payload data can be processed through a chain of single-kernel 1D-CNN, max-pooling, and multi-head attention layers. The same payload embedding can be fed into a multi-kernel 1D-CNN model and bypass the multi-attention layers before ensembling.

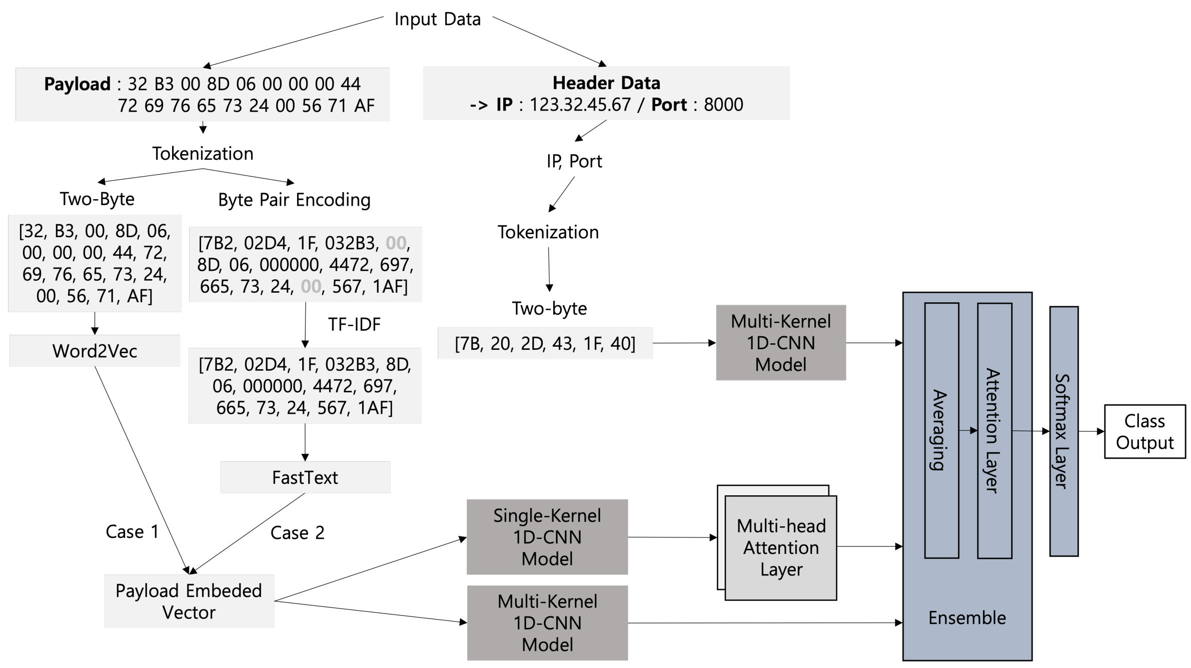
Input Data
Payload : 32 B3 00 8D 06 00 00 00 44 72 69 76 65 73 24 00 56 71 AF
Header Data -> IP : 123.32.45.67 / Port : 8000
Tokenization
IP, Port
Two-Byte
Byte Pair Encoding
[32, B3, 00, 8D, 06, 00, 00, 00, 44, 72, 69, 76, 65, 73, 24, 00, 56, 71, AF]
[7B2, 02D4, 1F, 032B3, 00, 8D, 06, 000000, 4472, 697, 665, 73, 24, 00, 567, 1AF]
Tokenization
Two-byte
TF-IDF
Word2Vec
[7B, 20, 2D, 43, 1F, 40]
Multi-Kernel 1D-CNN Model
[7B2, 02D4, 1F, 032B3, 8D, 06, 000000, 4472, 697, 665, 73, 24, 567, 1AF]
FastText
Case 1
Case 2
Payload Embeded Vector
Single-Kernel 1D-CNN Model
Multi-head Attention Layer
Multi-Kernel 1D-CNN Model
Averaging
Attention Layer
Softmax Layer
Ensemble
Class Output

Figure 1. The Overall Framework for CMAE-based Modeling of Network Threats

The other input data uses IP-port pairs that are encoded in two-byte tokens fed into a separate multi-kernel 1D-CNN model.

The output latent vectors by these different models are consolidated in an averaged vector in the ensemble block. This vector then enters a single-head self-attention layer to obtain the association between the collective feature elements. The vector of attention values is passed to the softmax function to generate the final class output.

A detailed explanation of the individual components of the learning structure is provided in each subsection below.

3.2. *Input Data Tokenization*

To model packet payload and header data using a neural network, it is necessary to convert text into numeric vectors. As the first step, we need to tokenize a given text. A token refers to a grammatical unit of language that can no longer be divided. A text can be divided into either sentence tokens or word tokens [47].

This study used the following tokenization methods for payload and header data. First, we tokenized payload data by dividing it into two-byte units and representing it in hexadecimal format ('00', 'ec') [48].

A payload can be represented with 256 tokens when tokenizing in two-byte tokens. Although the preprocessing speed is fast when tokenizing in two-byte tokens, the subtle context in a text may not be fully captured due to the limited number of tokens.

User-defined contents with many out-of-vocabulary (OOV) words make payload tokenizing challenging. Existing libraries such as NLTK [49] have learned contextual information for commonly used human spoken words. When these pre-trained libraries without customization are used to encode the payload, the correctness of payload tokenization and proper learning of contextual meanings cannot be guaranteed [50]. The OOV problem can be solved with subword segmentation methods [51,52]. For example, the word, *birthplace*, can be divided into smaller subwords, *birth* and *place*. For subword segmentation, we used byte pair encoding (BPE) [53] to tokenize the payload data into variable-length n-bytes. We used Sentencepiece, an open-source package that implements BPE algorithm[54]. However, since the payload data contains many user-defined tokens, pre-training was conducted from scratch using Sentencepiece. Using the newly trained model with Sentencepiece, unconventional variable-length n-byte tokens such as 'cf', '059', and '80100366' were observed.

The strengths and weaknesses of the tokenization methods discussed here were identified after performance evaluation.

3.3. Applying TF-IDF

TF-IDF is a statistical measure that indicates a word's importance in a particular document when there are multiple documents [55]. *Term Frequency (TF)* is a value that shows how often a particular word appears within a document. However, suppose the same word is frequently used in multiple documents. In that case, it may be a special character or word that commonly appears in sentences, such as commas and articles. Such characters or words may not help distinguish the difference between documents. *Inverse Document Frequency (IDF)* is the value obtained by taking the reciprocal of the document frequency. The lower the IDF value, the more frequently the word appears in multiple documents. TF-IDF is the product of TF and IDF, and as a word gets less relevant, its TF-IDF score approaches zero.

We removed tokens with the lowest TF-IDF score from the payload after the BPE-based tokenization was conducted. We found that removing irrelevant tokens enhanced the accuracy of threat modeling.

3.4. Token Embedding

In NLP, vectorization refers to the process of quantifying the features of each word. This process is called *word embedding*. For example, if a sentence consists of 3,000 words and each word is represented as a 50-dimensional vector through word embedding, a 3000*50 feature matrix is created. This feature matrix makes it possible to identify the similarity between words and understand contextual features [56,57].

The method of representing words as vectors can be divided into sparse representation [58] and distributed representation [59]. Sparse representation is achieved through one-hot encoding, where only the value at the predetermined index is set to one, and all other values are set to zero. The disadvantage of this representation method is that it cannot detect the similarity between each word.

The solution to the sparsity problem is distributed representation that vectorizes the semantics of words into a multi-dimensional space. The resulting vector is called an *embedding*. Distributed representation assumes that words that appear in similar contexts have similar meanings. This representation is created by training on a dataset of several words and distributing the word's meaning across multiple dimensions.

We applied two embedding algorithms, *Word2Vec* and *FastText*, to obtain the embedding of every token in a payload.

Note that an extra *padding* of default tokens is necessary because the length of each payload can vary. With the padding, multiple payloads are set to the same size. However, the padding can cause a side effect of having too many common default tokens appearing across payloads. The common default tokens can make it difficult to distinguish different types of threats in payloads. This problem is later solved with convolution and max-pooling.

3.4.1. Using Word2Vec

Word2Vec is an algorithm that uses the distributed representation to obtain embedding vectors, which has shown outstanding performance [60]. Word2Vec has two primary methods, CBOW (Continuous Bag of Words) and Skip-Gram [61]. CBOW predicts the middle word from the surrounding words.

In CBOW, the word to be predicted is called the center word, and the words used for prediction are called the context words. Word2Vec learns by sliding through the words, looking at the surrounding words of the center word, and updating the vector values of each word. During training, the vector corresponding to a word that does not appear within the window is updated to become farther away from the center word vector. On the other hand the vectors corresponding to the surrounding words that appear within the window are updated to become closer to the center word vector.

In contrast to CBOW, Skip-Gram predicts the surrounding words from the center word. This paper employed Skip-Gram, which is generally known to be better than CBOW [61]. Additionally, we applied the Word2Vec algorithm only to payloads that have been tokenized into two-byte units.

3.4.2. Using FastText

The Word2Vec algorithm described earlier has several known drawbacks [62]. First, it does not reflect the morphological characteristics of words well. For example, the words *teach*, *teacher*, and *teachers* are relevant to each other. However, in Word2Vec, these words are individually embedded. Thus, the semantic relationship between the embeddings of these words is not adequately captured. Additionally, embedding sparse words is difficult, and the algorithm cannot handle OOV words. Since Word2Vec constructs a vocabulary on a word-by-word basis, if a new OOV term is introduced, the entire dataset must be retrained. The OOV problem can persist as new words continue to emerge.

When the payload is tokenized in two-byte units, the tokens are limited to 256 words represented in hexadecimal. Therefore, Word2Vec can be applied without significant problems with sparse words or OOV terms. If the tokens are divided further through the BPE algorithm, the number of tokens increases, allowing for more detailed context information to be extracted. However, the OOV problem may arise. We handled this issue with a FastText [63] technique. FastText is a word embedding method developed by Facebook that extends the Word2Vec model. It embeds words that are combinations of n-grams of characters. For example, when n=3, the word *apple* is vectorized into the six tokens <ap, app, ppl, ple, le, apple>. Therefore, if the dataset is sufficient, all subwords of every word can be generated, and the OOV problem can be solved.

3.5. Text 1D-CNN Model

The embedding of the payloads goes through convolutional neural networks (CNN) for another round of further feature extraction. CNN is a representative deep learning model for image classification [64]. CNN consists of convolutional layers and pooling layers. In the convolutional layer, feature extraction of the image is performed through the following operation. Moving an nxm-sized matrix called a *kernel* over the image extracts a feature map from complex pixel information. Then, the feature map is downsampled through the pooling operation to reduce its size. Traditional CNNs have been used for two-dimensional image data. 1D-CNN can be used for one-dimensional data such as time series and texts. To extract the features of the payload, we applied the 1D-CNN model [65].

The example of using a single-kernel 1D-CNN is shown in Figure 2. The single-kernel 1D-CNN carries out the following process. Suppose a payload is partially tokenized into the two-byte format as 'e4, fd, 4c, a2, 80, c1, b1, 14, ab.' The embedding of this payload is shown as a matrix in Figure 2, where n is the sentence length, and k is the dimension of the embedding vector. A kernel with a size of 2 initially performs convolutional operations on 'e4, fd' and subsequently on 'fd, 4c'. The kernel repeats this operation as it strides down until the bottom of the matrix and obtains a vector with eight elements. An optimal kernel size can be determined experimentally through a hyperparameter tuning process.

Once the convolution completes, various pooling methods can be applied. The most commonly used is *max-pooling*. Figure 2 shows the max-pooling process that selects the maximum value from the vector returned by the convolution layer whose kernel size was set to 2.

The 1D-CNN structure we used is shown in Figure 3. The input embedding goes through two stages of convolution and pooling operations. After the second max-pooling, positional encoding functions are applied to each element according to Equation (1), where the scale value n is set to 10,000, empirically [66].

$$
\begin{aligned}
PE(x, 2i) &= \sin\left(\frac{x}{n^{2i/d}}\right) \\
PE(x, 2i+1) &= \cos\left(\frac{x}{n^{2i/d}}\right) \\
x&: \text{location of a token} \\
n&: \text{user-defined scalar} \\
d&: \text{dimension of the token embedding} \\
i&: \text{column index, } 0 \le i < d/2
\end{aligned}
\quad (1)
$$

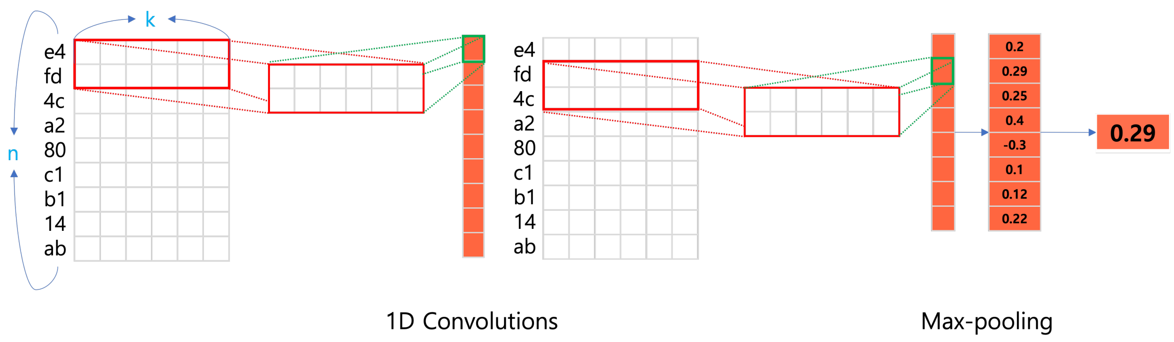

Figure 2. Processing a Payload through the 1D-CNN Model

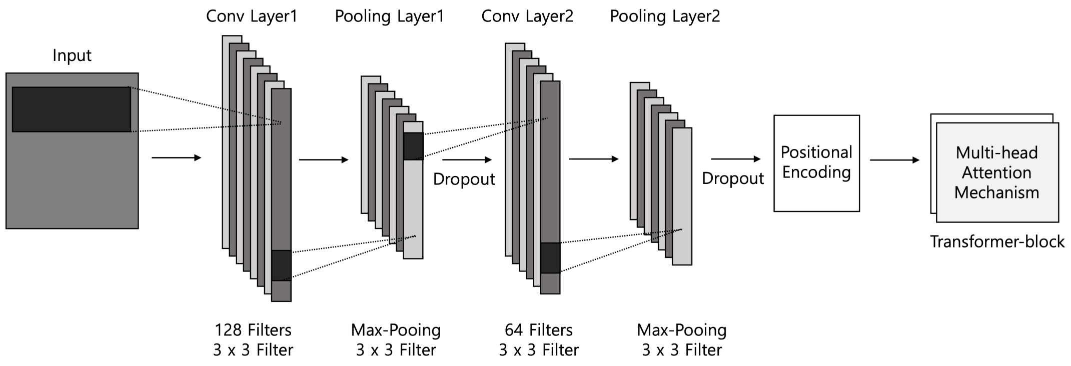

Figure 3. 1D-CNN Connected to a Transformer-block with Multi-Head Attention Layers

Each element is assigned a unique vector of positions through the positional encoding process. The embedding with the positional encoding values is passed to a *Transformer* block that implements a multi-head attention mechanism [66]. Alternatively, we used a 1D-CNN model with variable-size kernels, as shown in Figure 4. It uses a filter with a size of 128 and kernels with sizes 3, 4, and 5. Global max-pooling operation is applied to each intermediate result filtered with variable kernel sizes. Global max-pooling results are concatenated and are passed to the dense layer before it reaches the ensemble layer that averages the feature vectors returned by other models. The multi-kernel structure captures the association among various payload parts at different scales. Such multi-perspective feature analysis through various kernels is comparable to the mechanism implemented with multi-head attention layers. The multi-kernel 1D-CNN is worthy of consideration for being ensembled with other models.

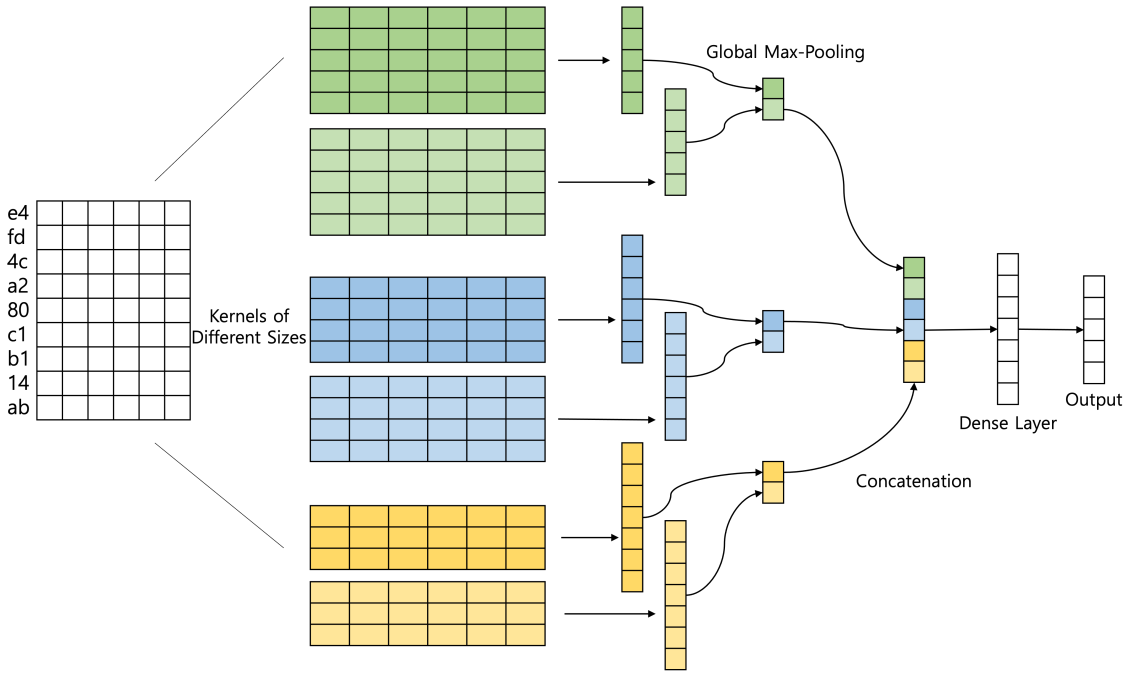

Figure 4. Muti-Kernel 1D-CNN model Architecture

IP and port information in the packet header is also fed into a separate multi-kernel 1D-CNN model. An IPv4 address was converted to an IPv6 address, then converted to hexadecimal format and broken into two-byte tokens. Ports were divided into two-byte tokens as well.

Sometimes, IP and port information alone can be the most salient threat indicator, regardless of the payload content. Therefore, we assigned a separate inference network for the IP and port information.

Max-pooling plays a vital role for both single-kernel and multi-kernel CNNs. Not all payloads are 1500 bytes in length. As mentioned earlier, default values have to be padded to the payload shorter than the default length, so that the void can be filled in. Having too many common default values across the payloads can make it difficult for our model to distinguish between different threat patterns present in the payloads. Ruling out meaningless values through the max-pooling operation can help CNNs extract only the essential features.

Max-pooling also speeds up the classification process as the feature set is compressed to a concise yet semantically distinct vector. The computation overhead at succeeding layers, such as the multi-head attention layers and dense layers, is significantly reduced. Such performance acceleration is needed to ensure prompt detection of threat before it causes substantial damage.

3.6. Multi-Head Attention

$$Attention = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2}$$

$$head_i = Attention(QW_i^Q KW_i QW_i^Q) \tag{3}$$

$$MultiHeadAttention = Concat(head_1, ..., head_h)W^O \tag{4}$$

Another critical component of our is the multi-head attention layers to which the single-kernel 1D-CNN is connected, as shown in Figure 3. From an embedding vector of a payload refined by 1D-CNN, Q, K, and V vectors are created. Each of these vectors goes through a linear transformation. Given the transformed vectors, we applied the scaled dot product attention function, as defined in Equation (2) [66]. The outcome of this function shows the attention scores of each part of the input embedding. The parts with high attention scores are the critical distinguishing points for the given embedding.

Each of the multiple heads uses a different linear transformation. Thus, each head produces attention scores in a different representation space. The output of each attention head is combined and multiplied by a weight matrix W^O. Multiple attention heads are computed to unravel as many meaningful relationships between tokens as possible from different perspectives.

The attention weights in QK^T and W^O are learned during the training.

3.7. The Ensemble Layer

Finally, the information learned through the feature extraction method and multiple models is passed to the ensemble block that consists of an averaging layer [67] and an attention layer [68]. This unique introduction of the attention layer to the ensemble block is to extract the semantic features from the aggregate embeddings from different models.

The attention weight values are passed to the softmax function [69]. The merged information is classified through a fully connected layer.

4. Evaluation

In this section, we evaluate the performance of our model.

4.1. Datasets

We evaluated the performance of our model using the CICIDS2017 public dataset and the threat dataset gathered at the ECSC managed by the Ministry of Education in South Korea. We aim to find an optimal classification model through various experiments using these datasets.

The CICIDS2017 dataset maintained by the Canadian Institute of CyberSecurity for research purposes consists of benign and malicious network packets. This dataset is generated by simulated attacks in a manner similar to actual attack incidents [15]. The CICIDS2017 dataset includes various attack packets such as DoS, port scan, distributed denial of service (DDoS), brute-force attacks against FTP and SSH services, bot attacks, and Web hacking, in addition to normal packets. The CICIDS2017 dataset is provided in a PCAP file format and contains network flow information such as IP addresses and ports of sources and destinations. The dataset was collected over the five days between July 3rd and July 7th, 2017. We used a refined version of the CICIDS2017 dataset that excludes incomplete packets with unintentional missing values [70]. The class distribution in the CICIDS2017 we used is shown in Table 2.

Table 2. The Class Distribution of CICIDS2017 Dataset

Class	The number of Cases
Benign	528,265
DoS	250,720
DDoS	124,111
Port scan	107,778
Attack against FTP & SSH	13,231
Bot Attack	1,905
Web Hacking	1,648

We used the real dataset from ECSC to validate the practicality and generality of our model. ECSC in South Korea monitors massive network traffic flows of educational institutions nationwide. Education institutions in South Korea have been vulnerable to cyber attacks due to the relatively large number of publicly accessible resources. Over a million threats are reported daily from approximately a thousand intrusion detection systems (IDS) running Snort rules.

Due to the limited number of security agents at ECSC, labeling millions of threat instances was infeasible. Instead of complete enumeration, ECSC filtered out the threats posing the highest risks according to NIST SP 800-30 [71]. Duplicate threats were removed as well. This refinement process narrowed down the original dataset to one with the class distribution shown in Table 3. The dataset was constructed from September 2021 to February 2022. For security reasons, we are not allowed to disclose the ECSC dataset. However, we can show that show that ECSC categorizes attacks into six types: intrusion attempts, malware infections, Web hacking, transit exploitation, DoS, and hacking emails. Packets that are falsely identified as a threat by the Snort rules are all labeled as benign packets.

For each dataset, we had 30% for a test set and 70% for a train set. Note that the class distributions are skewed in both datasets. Unlike previous studies, we trained our model without sampling or balancing the distribution.

Table 3. The Class Distribution of ECSC Dataset

Class	The Number of Cases
Benign	428,120
Intrusion Attempts	68,616
Malware	5,541
Hacking Email	397
Exploitation of Transit	191
Web Hacking	135
DDoS	7

4.2. Implementation and Parameter Configuration

We implemented our model with Tensorflow and Keras 2.6. Our model was tested on NVIDIA DGX-1 with a 2.2 GHz 20-core Intel Xeon E5-2698 v4 CPU, sixteen 2133 MHz 32GB DDR4 LRDIMM, and eight NVIDIA Tesla V100-32GB GPUs. NVIDIA DGX-1 is operated with Ubuntu 18.04.5 LTS. We executed our model in a container with Docker v19.03.14.

Table 4 lists the main parameters necessary for the experiment. Note that these parameters are empirically set.

Table 4. Parameters for CMAE

Parameter	Setting
n-gram	two-byte
BPE	vocab_size 32000, max_sentence_length MAX
Word2Vec	embedding_size 64, window_size 5, min_count 5, Skip_gram
FastText	embedding_size 64, window_size 5, min_count 5, Skip_gram
TF-IDF	Twenty least relevant tokens removed
Word embedding size	64
Payload length	MAX or 1500 bytes
1D-CNN	fillter_size (128, 64), kernel_size 3
Multi-kernel 1D-CNN	fillter_size 128, kernel_size (3,4,5)
Multi-head attention	d_model 128, num_heads 2
Drop out	0.2
Loss function	binary_crossentropy and categorical_crossentropy
Activation function	GELU and softmax
Optimizer	AdaBeliefOptimizer (learning_rate=5e-4, epsilon=1e-16, weight_decay=1e-4)
Scheduler	ReduceLROnPlateau (factor=0.3, patience=2, min_lr=1e-5)
Epochs	50
Batch size	32
Early Stopping	5

We used two-byte tokens for n-gram tokenization. For BPE, the GE refers to the number of words required for training to distinguish token units. In this paper, we set 32,000 for the vocabuluary size. The payload length was set to the maximum as seen in the dataset or the default 1500 bytes. The entire content was embedded when the payload length was set to the max. However, many packets are smaller than 1500 bytes. Thus, these packets may require padding of default values.

The embedding dimension size was set to 64 for both Word2Vec and FastText. 64 was the smallest dimension size without compromising accuracy. With the compact embedding setting, we could reduce training time. The window size, which determines how many surrounding words are used to understand the meaning, was set to 5. The minimum count was set to exclude tokens that appeared fewer than five times in the entire word frequency. Additionally, we chose skip-gram over CBOW for the embedding method. Twenty tokens with the least relevancy are removed from payloads according to the TD-IDF score.

For the 1D-CNN model, the filter size was 128 and 64, while the kernel size was set to 3. For the Multi-kernel 1D-CNN, the filter size was 128, and the kernel size was set to 3, 4, and 5. The parameters for multi-head attention were set to 128 for the entire dimension. The number of attention heads was set to 2. Thus, each 128-dimensional payload token vector was divided by 2 to obtain 64-dimensional Q, K, and V vectors. Dropout was set to 0.2 to prevent overfitting. Binary cross-entropy was used as the loss function for binary classification between normal and malicious data. Categorical cross-entropy was used for multi-class classification as the loss function. The Gaussian error linear unit (GELU) function [72] was used as the activation function, and softmax was applied to the last classification layer. AdaBeliefOptimizer [73] was used as the optimization function, and the ReduceLROnPlateau scheduler from Keras was applied to improve performance. We trained over 50 epochs. We had a batch size of 32. The early stoppage was set to 5.

4.3. Evaluation Metric

In this study, performance is evaluated using F1-score. The precision is the ratio of true positives to the total number of positive predictions (Equation (5)). The recall is the ratio of true positives to the total number of actual positives (Equation (6)). F1-score is the harmonic mean of precision and recall (Equation (7)). We computed class-wise F1-scores.

We used micro average as defined in Equation (8) to assess the overall classification performance as well. Micro average accounts for the imbalance in data between classification classes. Therefore, we choose the micro average over the macro average that takes the simple average of F1-scores of all classification classes.

- TP(True Positives): The number of test results which indicate that a threat is classified into a right type
- TN(True Negatives): The number of test results which correctly indicate that a threat is not classified into a wrong type
- FP(False Positives): The number of test results which wrongly indicate that a threat is correctly classified
- FN(False Negatives): The number of test results which wrongly indicate that a threat is not classified to an incorrect type

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

$$Recall = \frac{TP}{TP + FN} \tag{6}$$

$$F1 - Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{7}$$

$$\begin{aligned} Micro\_precision &= \frac{TP_1 + ... + TP_N}{TP_1 + FP_1 + ...TP_n + FP_n} \\ Micro\_Recall &= \frac{TP_1 + ... + TP_N}{TP_1 + FN_1 + ...TP_n + FN_n} \\ Micro\ Average &= \\ 2 * \frac{Micro\_Precision * Micro\_Recall}{Micro\_Precision + Micro\_Recall} \\ n&\text{: class label} \end{aligned} \tag{8}$$

4.4. Performance Assessment

We compared the performance of various previous models and our new model using the CICIDS 2017 dataset. We summarized the performance of the models in Table 5. CMAE outperformed previous state-of-the-art algorithms [39,40]. Through the ensemble layer, we achieved 0.8%p higher f1-score than RTIDS.

Table 5. Performance of the Classification Algorithms Using Payload Data

Algorithm	Precision	Recall	F1-Score	Accuracy
PayloadEmbeddings [39]	0.953	0.953	0.953	0.953
RTIDS [40]	0.983	0.987	0.990	0.985
CMAE Binary Classification	0.999	0.999	0.999	0.999
CMAE Multiclass Classification	0.998	0.998	0.998	0.998

Unlike previous studies, we conducted experiments using all data without sampling or balancing the data between classes. In Table 5, we compared the classification performance for each attack type using CMAE with BPE, TF-IDF and FastText. We confirmed that the F1-score for all attack types was over 0.99. We observed that CMAE was resilient to significant data imbalance between threat types.

We also varied the data preprocessing methods and measured the performance of binary and multiclass classifications.

According to Table 7, approaches with BPE and two-byte tokenization methods showed similarly high performance. Two-byte tokenization incurs less computation overhead. Therefore, the two-byte tokenization can be more suitable for mission-critical classification needs. The two-byte tokenization performed better with a maximum payload length than with a length of 1500 bytes. We attribute this outcome to CMAE's ability to capture essential characteristics from the payloads even with a smaller vocabulary size.

The classification performance using ECSC dataset is shown in Table 8. We used the two-byte tokenization and Word2Vec embedding. The payload length was fixed to 1500 bytes. CMAE yielded an F1-score of 0.9966 for binary classification of payload-only data. Performance of the multiclass classification on the payload-only data was on-par with that of binary classification. When the encoding of the headers (IPs and port) was included in the input data, F1-score of the multiclass classification improved to 0.9980. The result was consistent with the one using CICIDS2017, as CMAE outperformed the state-of-the-art algorithms. We observed that the BPE-based tokenization, TF-IDF-based token removal and FastText embedding did not show favorable performance.

We can infer that the ensemble model is robust enough to model payload with simpler tokenization and embedding.

Table 6. Micro Performance Result of CMAE Using CICIDS2017 dataset

Type of Attack	False Positives	True Positives	F1-Score
Port Scan	379,549	107,733	0.9924
DDoS	379,549	124,111	0.9984
Bot Attack	379,549	1,905	0.9997
Web Hacking	166,715	1,658	0.9980
DoS	386,906	250,720	0.9992
Attack against FTP-SSH	392,699	13,233	0.9987

Table 7. Performance of CMAE using CICIDS2017 without Sampling

Tokenization and Embedding	Class	Payload Length	F1-Score
BPE + TF-IDF + FastText	multiclass	Max	0.9958
BPE + TF-IDF + FastText	binary	Max	0.9992
BPE + TF-IDF + FastText	binary	1500	0.9912
2byte + Word2Vec	multiclass	Max	0.9982
2byte + Word2Vec	binary	Max	0.9997
2byte + Word2Vec	binary	1500	0.9907

Table 8. Classification Performance Using ECSC Data

Input Data	Tokenization & Embedding	Classification	Payload Length	F1-Score
Payload	two-byte + Word2vec	binary	1500	0.9966
Payload	two-byte + Word2vec	multiclass	1500	0.9965
Payload + IP + port	two-byte + Word2vec	multiclass	1500	0.9980

5. Discussion

CMAE is currently an integral part of ECSC, functioning as an automated network threat classification engine. Threats identified with over 99% confidence by the softmax function are automatically registered as a real attack incident and shared among all cyber agents managed by ECSC.

According to ECSC, simple intrusion attempts count the majority of the attacks. Screening these threats with high accuracy has enabled security agents to spend more time devising countermeasures for more complex and sophisticated threats. Approximately 150,000 threat incidents can be classified in a minute. Manual monitoring labor costs can be significantly reduced, which helps ECSC deal with sheer volume of threat data everyday.

However, there is a lot to be improved in our model. First, regarding bit rates, CMAE processes threat data at approximately 30 Mbps, given the payload size set to 1,500 bytes. IDS devices deployed in educational institutions in South Korea typically screen 1Gbps network traffic. To keep up with the line speed and replace the imperfect rule-based classification engines in the IDS devices, CMAE has to use dozens more GPUs, which is not a cost-effective solution. In reality, ECSC uses IDS devices as the first responder to threats, and CMAE works as a secondary classification engine. Such a security monitoring chain still helps reduce massive threat data that pass unnoticed due to the limited human resource. However, because it is infeasible for CMAE to examine the entire packet upfront in line speed, the threat surveillance system can miss many threats previously not defined by Snort rules. Quantizing the neural network is an easy way to speed up the machine learning algorithm. Acceleration with recently emerging neural processing units can also be considered, as they are known to perform an order of magnitude better than GPUs.

We have proven CMAE training quality with synthetic and real-world datasets. However, whether the pre-trained CMAE can perform well on a new premise with different network data patterns is unclear, although CMAE is trained with a broad spectrum of threat data collected from a thousand IDS devices. Each different site has to construct its own CMAE training facility, which is far from being cost-effective. More diverse train data is needed to make the pre-trained CMAE work out-of-box. However, collecting real-world data is extremely difficult due to confidentiality and privacy concerns.

The attention mechanism of CMAE can help security agents identify the parts of the network data causing the threats. However, precisely pinpointing the causes is infeasible due to convolution and max-pooling. As an alternative to the 1D-CNN structure, every token can be learned directly through the self-attention block. However, due to the padding of many meaningless default values, the system solely based on self-attention suffers from performance degradation. Kernel structure and striding mechanism are to be revised to tackle this particular problem with network payload.

Lastly, network threats can exhibit seasonality. Therefore, a trained CMAE instance can become obsolete whenever a drift in the threat pattern occurs. Detecting the drift and proactively re-training the CMAE model is crucial for sustaining high classification accuracy at all times.

6. Conclusion

This paper introduced a model that detects network threats based on learning the patterns in the packet payloads and header data. First, we uniformly divided the payload data into two-byte units and applied the BPE technique to extract an appropriate set of words at various n-gram levels. We then extracted embedding vectors for each word using distributed representation models such as Word2Vec and FastText trained from scratch. We classified threats by a novel ensemble block that aggregates the feature vectors learned by various models, including a single-kernel 1D-CNN model paired with a multi-head attention block, and multi-kernel 1D-CNN models,

To demonstrate the superiority of the proposed ensemble model, we conducted experiments using an open dataset called CICIDS2017. Through this experiment, we adjusted the hyperparameters of n-grams, word embedding models, and various deep learning networks to obtain the optimal classification system. As a result, even though only payload data was utilized in the CICIDS2017 dataset, it outperformed the state-of-the-art models with an F1-score up to 0.9997. We also applied our model to a large-scale security surveillance center (ECSC) currently operated by the Ministry of Education in South Korea. With the real-world ECSC dataset generated by a thousand IDS devices, our model was proven effective, achieving an F1-score of 0.998 in classifying threats. Interestingly, our model was robust even when the packet was tokenized and embedded more simply. The convolution and the max-pooling effectively dealt with the meaningless default values padded to short payloads. Also, our unique ensemble block with an attention layer helped preserve the semantic information from the aggregate feature vectors formed by various models. Our ensemble model can classify 150,000 threats per minute on an Nvidia V100-32GB GPU machine to relieve the monitoring workload for security agents. Despite these benefits, more research has to be done on improving the generality of the model so that it works on different premises out-of-box. We are also seeking model compression and hardware acceleration techniques to screen entire packets for unknown threats at a line speed.

Author Contributions: Conceptualization, K.H.M. and Y.Y; methodology, K.H.M and Y.Y.; software, K.H.M.; validation, K.H.M and Y.Y.; formal analysis, K.H.M and Y.Y.; investigation, K.H.M and Y.Y.; resources, Y.Y.; data curation, K.H.M and Y.Y; writing—original draft preparation, K.H.M and Y.Y.; writing—review and editing, K.H.M and Y.Y.; visualization, K.H.M; supervision, Y.Y.; project administration, Y.Y.; funding acquisition, Y.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Ministry of Trade, Industry & Energy (MOTIE) and the Korea Institute for Advancement of Technology (KIAT), under Grants P0014268 Smart HVAC demonstration support. This research was also supported by the MSIT (Ministry of Sciencd and ICT), Korea under the ITRC (Information Technology Research Center) support program (RS-2023-00259099) supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation, and supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. RS-2023-00240211).

Acknowledgments: We appreciate The Korea Ministry of Education for the opportunity to develop our AI-based intrusion detection system for the Education Cyber Security Center operated by The Korea Education and Research Information Service (KERIS).

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

CMAE	Convolutionam Multi-head Attention Ensemble

References

1. Yoon, Y.; Ban, D.h.; Han, S.W.; Woo, H.U.; Heo, E.h.; Shin, S.H.; Lee, J.k.; An, D.h. Terminal, cloud apparatus, driving method of terminal, method for processing cooperative data, computer readable recording medium, 2022. US Patent 11,228,653.
2. Yoon, Y.; Ban, D.; Han, S.; An, D.; Heo, E. Device/cloud collaboration framework for intelligence applications. In *Internet of Things*; Elsevier, 2016; pp. 49–60.
3. Chimuco, F.T.; Sequeiros, J.B.; Lopes, C.G.; Simões, T.M.; Freire, M.M.; Inácio, P.R. Secure cloud-based mobile apps: attack taxonomy, requirements, mechanisms, tests and automation. *International Journal of Information Security* **2023**, pp. 1–35.
4. dailysecu. Kaseya VSA ransomware attack. https://dailysecu.com/news/articleView.html?idxno=133066, 2021.
5. newsdirectory3. Apartment wall pad hacking. https://www.newsdirectory3.com/apartment-wall-pad-hacking-private-life-video-leaked-police-investigation-ministry-of-science-and-technology-must-cover-camera-lens/, 2021.
6. securitymagazine. Log4j vulnerability continues to threaten enterprise security. https://www.securitymagazine.com/articles/97162-log4j-vulnerability-continues-to-threaten-enterprise-security, 2022.
7. Lee, W.; Stolfo, S.J.; Mok, K.W. Mining in a data-flow environment: Experience in network intrusion detection. Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, 1999, pp. 114–124.
8. Alaidaros, H.; Mahmuddin, M.; Al Mazari, A.; others. An overview of flow-based and packet-based intrusion detection performance in high speed networks. Proceedings of the International Arab Conference on Information Technology, 2011, pp. 1–9.
9. Patel, S.K.; Sonker, A. Rule-based network intrusion detection system for port scanning with efficient port scan detection rules using snort. *International Journal of Future Generation Communication and Networking* **2016**, *9*, 339–350.
10. Nasi, E. Bypass antivirus dynamic analysis. *Limitations of the AV model and how to exploit them* **2014**.
11. Ito, M.; Iyatomi, H. Web application firewall using character-level convolutional neural network. 2018 IEEE 14th International Colloquium on Signal Processing & Its Applications (CSPA). IEEE, 2018, pp. 103–106.
12. Hao, S.; Long, J.; Yang, Y. Bl-ids: Detecting web attacks using bi-lstm model based on deep learning. International Conference on Security and Privacy in New Computing Environments. Springer, 2019, pp. 551–563.
13. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* **2018**.
14. Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* **2019**.
15. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp* **2018**, *1*, 108–116.
16. Roesch, M.; others. Snort: Lightweight intrusion detection for networks. Lisa, 1999, Vol. 99, pp. 229–238.
17. da Silva, A.S.; Wickboldt, J.A.; Granville, L.Z.; Schaeffer-Filho, A. ATLANTIC: A framework for anomaly traffic detection, classification, and mitigation in SDN. NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium. IEEE, 2016, pp. 27–35.
18. Chang, S.; Qiu, X.; Gao, Z.; Qi, F.; Liu, K. A flow-based anomaly detection method using entropy and multiple traffic features. 2010 3rd IEEE International Conference on Broadband Network and Multimedia Technology (IC-BNMT). IEEE, 2010, pp. 223–227.
19. Muraleedharan, N.; Parmar, A.; Kumar, M. A flow based anomaly detection system using chi-square technique. 2010 IEEE 2nd international Advance computing conference (IACC). IEEE, 2010, pp. 285–289.

20. Wang, K.; Stolfo, S.J. Anomalous payload-based network intrusion detection. International workshop on recent advances in intrusion detection. Springer, 2004, pp. 203–222.
21. Yoon, Y.; Jung, H.; Lee, H. Abnormal network flow detection based on application execution patterns from Web of Things (WoT) platforms. *PloS one* **2018**, *13*, e0191083.
22. Perdisci, R.; Ariu, D.; Fogla, P.; Giacinto, G.; Lee, W. McPAD: A multiple classifier system for accurate payload-based anomaly detection. *Computer networks* **2009**, *53*, 864–881.
23. Yoon, Y.; Choi, Y. On Multilateral Security Monitoring and Analysis With an Abstract Tomogram of Network Flows. *IEEE Access* **2018**, *6*, 24118–24127.
24. Reddy, R.R.; Ramadevi, Y.; Sunitha, K.N. Effective discriminant function for intrusion detection using SVM. 2016 International conference on advances in computing, communications and informatics (ICACCI). IEEE, 2016, pp. 1148–1153.
25. Zolotukhin, M.; Hämäläinen, T.; Kokkonen, T.; Siltanen, J. Analysis of HTTP requests for anomaly detection of web attacks. 2014 IEEE 12th International Conference on Dependable, Autonomic and Secure Computing. IEEE, 2014, pp. 406–411.
26. Koc, L.; Mazzuchi, T.A.; Sarkani, S. A network intrusion detection system based on a Hidden Naïve Bayes multiclass classifier. *Expert Systems with Applications* **2012**, *39*, 13492–13500.
27. Zhang, Z.; George, R.; Shujaee, K. Efficient detection of anomolous HTTP payloads in networks. SoutheastCon 2016. IEEE, 2016, pp. 1–3.
28. Farnaaz, N.; Jabbar, M. Random forest modeling for network intrusion detection system. *Procedia Computer Science* **2016**, *89*, 213–217.
29. Primartha, R.; Tama, B.A. Anomaly detection using random forest: A performance revisited. 2017 International conference on data and software engineering (ICoDSE). IEEE, 2017, pp. 1–6.
30. Yoon, Y.; Hwang, H.; Choi, Y.; Joo, M.; Oh, H.; Park, I.; Lee, K.H.; Hwang, J.H. Analyzing basketball movements and pass relationships using realtime object tracking techniques based on deep learning. *IEEE Access* **2019**, *7*, 56564–56576.
31. Hwang, H.; Ahn, J.; Lee, H.; Oh, J.; Kim, J.; Ahn, J.P.; Kim, H.K.; Lee, J.H.; Yoon, Y.; Hwang, J.H. Deep learning-assisted microstructural analysis of Ni/YSZ anode composites for solid oxide fuel cells. *Materials Characterization* **2021**, *172*, 110906.
32. Kang, M.; Lee, W.; Hwang, K.; Yoon, Y. Vision transformer for detecting critical situations and extracting functional scenario for automated vehicle safety assessment. *Sustainability* **2022**, *14*, 9680.
33. Kim, T.Y.; Cho, S.B. Web traffic anomaly detection using C-LSTM neural networks. *Expert Systems with Applications* **2018**, *106*, 66–76.
34. Wang, W.; Zhu, M.; Zeng, X.; Ye, X.; Sheng, Y. Malware traffic classification using convolutional neural network for representation learning. 2017 International conference on information networking (ICOIN). IEEE, 2017, pp. 712–717.
35. Wang, Y.; An, J.; Huang, W. Using CNN-based representation learning method for malicious traffic identification. 2018 IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS). IEEE, 2018, pp. 400–404.
36. Ahmad, Z.; Shahid Khan, A.; Nisar, K.; Haider, I.; Hassan, R.; Haque, M.R.; Tarmizi, S.; Rodrigues, J.J. Anomaly detection using deep neural network for IoT architecture. *Applied Sciences* **2021**, *11*, 7050.
37. Naseer, S.; Saleem, Y.; Khalid, S.; Bashir, M.K.; Han, J.; Iqbal, M.M.; Han, K. Enhanced network anomaly detection based on deep neural networks. *IEEE access* **2018**, *6*, 48231–48246.
38. Liu, H.; Lang, B.; Liu, M.; Yan, H. CNN and RNN based payload classification methods for attack detection. *Knowledge-Based Systems* **2019**, *163*, 332–341.
39. Hassan, M.; Haque, M.E.; Tozal, M.E.; Raghavan, V.; Agrawal, R. Intrusion detection using payload embeddings. *IEEE Access* **2021**, *10*, 4015–4030.
40. Wu, Z.; Zhang, H.; Wang, P.; Sun, Z. RTIDS: A robust transformer-based approach for intrusion detection system. *IEEE Access* **2022**, *10*, 64375–64387.
41. Xu, H.; Chen, W.; Zhao, N.; Li, Z.; Bu, J.; Li, Z.; Liu, Y.; Zhao, Y.; Pei, D.; Feng, Y.; others. Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications. Proceedings of the 2018 world wide web conference, 2018, pp. 187–196.

42. Yousefi-Azar, M.; Varadharajan, V.; Hamey, L.; Tupakula, U. Autoencoder-based feature learning for cyber security applications. 2017 International joint conference on neural networks (IJCNN). IEEE, 2017, pp. 3854–3861.
43. Hwang, R.H.; Peng, M.C.; Nguyen, V.L.; Chang, Y.L. An LSTM-based deep learning approach for classifying malicious traffic at the packet level. *Applied Sciences* **2019**, *9*, 3414.
44. Min, E.; Long, J.; Liu, Q.; Cui, J.; Chen, W. TR-IDS: Anomaly-based intrusion detection through text-convolutional neural network and random forest. *Security and Communication Networks* **2018**, *2018*.
45. Liu, J.; Song, X.; Zhou, Y.; Peng, X.; Zhang, Y.; Liu, P.; Wu, D.; Zhu, C. Deep anomaly detection in packet payload. *Neurocomputing* **2021**.
46. Kim, Y.; Ko, Y.; Euom, I.; Kim, K. Web Attack Classification Model Based on Payload Embedding Pre-Training. *Journal of the Korea Institute of Information Security & Cryptology* **2020**, *30*, 669–677.
47. Grefenstette, G.; Tapanainen, P. What is a word, what is a sentence?: problems of Tokenisation **1994**.
48. Wang, Y.; Xiang, Y.; Zhou, W.; Yu, S. Generating regular expression signatures for network traffic classification in trusted network management. *Journal of Network and Computer Applications* **2012**, *35*, 992–1000.
49. Loper, E.; Bird, S. Nltk: The natural language toolkit. *arXiv preprint cs/0205028* **2002**.
50. Lochter, J.V.; Silva, R.M.; Almeida, T.A. Deep learning models for representing out-of-vocabulary words. Brazilian Conference on Intelligent Systems. Springer, 2020, pp. 418–434.
51. Zhang, Z.; Zhao, H.; Ling, K.; Li, J.; Li, Z.; He, S.; Fu, G. Effective subword segmentation for text comprehension. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* **2019**, *27*, 1664–1674.
52. Wu, Y.; Zhao, H. Finding better subword segmentation for neural machine translation. In *Chinese computational linguistics and natural language processing based on naturally annotated big data*; Springer, 2018; pp. 53–64.
53. Sennrich, R.; Haddow, B.; Birch, A. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909* **2015**.
54. Kudo, T.; Richardson, J. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226* **2018**.
55. Aizawa, A. An information-theoretic perspective of tf–idf measures. *Information Processing & Management* **2003**, *39*, 45–65.
56. Kenter, T.; De Rijke, M. Short text similarity with word embeddings. Proceedings of the 24th ACM international on conference on information and knowledge management, 2015, pp. 1411–1420.
57. Liu, Y.; Liu, Z.; Chua, T.S.; Sun, M. Topical word embeddings. Twenty-ninth AAAI conference on artificial intelligence, 2015.
58. Rubinstein, R.; Bruckstein, A.M.; Elad, M. Dictionaries for sparse representation modeling. *Proceedings of the IEEE* **2010**, *98*, 1045–1057.
59. Howard, M.W.; Kahana, M.J. A distributed representation of temporal context. *Journal of mathematical psychology* **2002**, *46*, 269–299.
60. Goldberg, Y.; Levy, O. word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722* **2014**.
61. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* **2013**.
62. Horn, F. Context encoders as a simple but powerful extension of word2vec. *arXiv preprint arXiv:1706.02496* **2017**.
63. Bojanowski, P.; Grave, E.; Joulin, A.; Mikolov, T. Enriching word vectors with subword information. *Transactions of the association for computational linguistics* **2017**, *5*, 135–146.
64. Zeiler, M.D.; Fergus, R. Visualizing and understanding convolutional networks. European conference on computer vision. Springer, 2014, pp. 818–833.
65. Azizjon, M.; Jumabek, A.; Kim, W. 1D CNN based network intrusion detection with normalization on imbalanced data. 2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC). IEEE, 2020, pp. 218–224.
66. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *Advances in neural information processing systems* **2017**, *30*.
67. Mehrkanoon, S. Deep neural-kernel blocks. *Neural Networks* **2019**, *116*, 46–55.

68. Bahdanau, D.; Cho, K.; Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* **2014**.
69. Luong, M.T.; Pham, H.; Manning, C.D. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025* **2015**.
70. Alrowaily, M.; Alenezi, F.; Lu, Z. Effectiveness of machine learning based intrusion detection systems. International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage. Springer, 2019, pp. 277–288.
71. Stoneburner, G.; Goguen, A.; Feringa, A.; others. Risk management guide for information technology systems. *Nist special publication* **2002**, *800*, 800–30.
72. Hendrycks, D.; Gimpel, K. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415* **2016**.
73. Zhuang, J.; Tang, T.; Ding, Y.; Tatikonda, S.C.; Dvornek, N.; Papademetris, X.; Duncan, J. Adabelief optimizer: Adapting stepsizes by the belief in observed gradients. *Advances in neural information processing systems* **2020**, *33*, 18795–18806.