electronics

Article

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

Hyeonmin Kim [1] and Young Yoon [2,*]

[1] Hana Securities, Seoul 07321, Republic of Korea; rlagusals1232@naver.com
[2] Department of Computer Science, Hongik University, Seoul 04066, Republic of Korea
* Correspondence: young.yoon@hongik.ac.kr

Abstract: Using traditional methods based on detection rules written by human security experts presents significant challenges for the accurate detection of network threats, which are becoming increasingly sophisticated. In order to deal with the limitations of 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 that contain binary and 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 these features, we combine a text 1D CNN and a multi-head attention network in a novel fashion. We validated the effectiveness of our classification technique on 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 performance compared to previous studies, achieving highly accurate classification with an F1-score of 0.998. Our model can also preprocess and classify 150,000 network threats per minute, helping security agents in the field maximize their time and analyze more complex attack patterns.

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

check for updates

Citation: Kim, H.; Yoon, Y. An Ensemble of Text Convolutional Neural Networks and Multi-Head Attention Layers for Classifying Threats in Network Packets. *Electronics* **2023**, *12*, 4253. https://doi.org/10.3390/electronics12204253

Academic Editor: Cheng-Chi Lee

Received: 9 September 2023
Revised: 28 September 2023
Accepted: 12 October 2023
Published: 14 October 2023

1. Introduction

The attack surface is increasing as mobile devices proliferate and interact with various IoT and cloud platforms [1–3]. The increased attack surface attracts diverse and sophisticated cyber threats that are becoming more challenging to detect. For example, in July 2021, hundreds of sites in the United States were affected by ransomware attacks due to the 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 prior incidents where IP cameras were hacked, multiple households suffered the leakage of a large amount of private information. In December 2021, there were infamous cases of malware propagation exploiting the Apache Log4j vulnerability [6]. We demonstrated that once a connected vehicle is infiltrated, erroneous control messages can be easily fed into the CAN (Controller Area Network) bus [7]. Tian et al. demonstrated that the deep learning model can be maliciously altered to disturb various autonomous functions of connected unmanned aerial vehicles [8]. These incidents show that cyber threats continue to occur in various ways as the ubiquity of online services, applications, and mobile devices grows.

Methods such as secure coding, vulnerability diagnosis, intrusion detection systems (IDS), and firewalls have been applied to prevent cyber attacks. IDS, in particular, employ techniques for analyzing the contents of network flows or packets to detect and notify malicious intent from network traffic [9,10]. IDS is commonly equipped with detection rules written in widely adopted Snort format to define known threats [11]. However, as cyber attacks evolve, attackers have devised various evasion methods [12]. 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 for new attack types. Human security experts require 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, this 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, including a payload and a header specifying the 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 have applied natural language processing (NLP) techniques based on deep learning to better understand the textual data in payloads [13,14]. 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 of structures, unlike ordinary natural language texts. Therefore, we cannot blindly use pre-trained language models such as BERT [15] and Reoberta [16] 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 *Conv. 1D 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 are tokenized into n-grams, and the word embeddings are used to learn the contextual information. Then, the 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 the attention weights of each payload token to learn the structural features leading to threats. We also trained the multi-kernel 1D CNN to discover the association between the 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 into a softmax function that selects the likely type of threat.

We evaluated our novel method on the CICIDS2017 open dataset [17] and real-world network datasets collected by the Education Cyber Security Center (ECSC) in South Korea. Since January 2022, the ECSC has been using our technique to detect and classify approximately 1.2 million network threats daily.

The contributions of this study can be summarized as follows:

1. We propose a composite neural network structure with a novel attention-based ensemble layer.
2. This study boasts a highly accurate attack classification capability with a micro F1-score of 0.998 based on the CICIDS2017 dataset.

3. We significantly reduce false positives, which avoids diverting the attention of security analysts from true threats.
4. We combine Text-CNN techniques with a customized Transformer architecture to deal with payloads containing varying lengths of content.
5. In addition to the simulated public dataset, we tested the effectiveness of our solution on a large-scale intrusion system currently operating in Korea to monitor threats against educational institutions in South Korea. This study also discusses the practical issues observed in the AI-based IDS in production.
6. Besides attack classification, the attention-based mechanism enables the identification of critical parts within the payload.
7. This study provides a comprehensive performance analysis of different payload preprocessing methods.
8. The CMAE model can classify 150,000 threats within a minute. With the CMAE model, the Korean Education Cyber Security Center (ECSC) screens 1.28 million daily threats occurring across educational institutions nationwide in real time. Since no threat has gone unnoticed, security has been greatly enhanced.

The rest of this paper is structured as follows. Section 2 discusses the related works; Section 3 describes the mechanism of our approach; Section 4 presents the evaluation results; Section 5 discusses the limitations of CMAE-based threat monitoring and directions for future works; and finally, we present our conclusions in Section 6.

2. Related Works

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

2.1. Rule-Based Approaches

The most widely adopted practice is composing threat detection rules according to Snort [18]. Many previous studies have developed rules for detecting network threats. ATLANTIC [19] utilized flow-based information and detected the threat of malicious traffic by calculating the entropy deviation of traffic-flow data. In [20], entropy and the characteristics of multiple traffic were also utilized based on network-flow information. Ref. [21] developed a flow-based system that detects abnormal activity by applying the chi-squared technique to IP flow characteristics. Some studies have examined packet payloads. In PAYL [22], 256 features are extracted for each 1-byte payload. The extracted 1-byte payload calculates the frequency distribution within the payload flowing through each host and port and calculates the mean and the standard deviation. Then, it calculates similarity using the Mahalanobis distance and detects an attack if it exceeds a specific threshold value. In [23], abnormal network flow showing the Web of Things application logic violation was detected using the RETE-based rule engine. In [24], the features of a segmented payload were represented as two-gram tokens. In [25], 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-service (DoS) attacks.

These rule-based methods have much room for improvement in terms of the accurate detection of network threats. The process of extracting features is manual and excessively complex. The criteria for abnormal incidents are based on thresholds set laboriously through 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 machine [26], support vector data description in concert with DBSCAN [27], naive Bayes [28,29], and random forest [30,31], have been utilized.

Recently, deep learning techniques have seen significant advancements in various fields, as evidenced by numerous studies [32–34]. For network threat detection, in [35], anomalous traffic was detected 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 [36,37]. Refs. [38,39] utilized DNNs (deep neural networks) to detect anomalies in IoT devices and network traffic data. Ref. [40] used a CNN and an LSTM to learn the threat patterns inside payload data. Various deep learning techniques have been trained on the CICIDS2017 dataset [41] generated during simulated network attacks. Yu et al. used a hierarchical packet byte-based CNN to classify attacks in a PCAP file [42]. In [43], RTIDS was used with a Transformer network to model the threats captured in the CICIDS datasets.

Class imbalance is a significant issue, as threat data are typically much less abundant compared to normal data. This issue makes supervised learning methods susceptible to overfitting and bias problems. Many unsupervised learning-based studies have been conducted to address this issue. Noteworthy unsupervised learning approaches have employed autoencoder methods [44,45].

2.3. The Employment of NLP Techniques

Research that employs deep learning from an NLP perspective using payload data is also relevant to our approach. In [46], 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 the packet data version, flags, protocols, and source and destination IPs and ports was combined to form a single sentence. Given the input data in sentences, word embeddings were used to extract the meaning of the packet. An LSTM model was used to learn temporal information between fields in the packet header. As a result, promising performance was achieved on the ISCX-IDS-2012, USTC-TFC-2016, Mirai-RGU, and Mirai-CCU datasets.

In TR-IDS [47], word embeddings and Text-CNN were applied using payload data, and then classification was performed using random forests, along with the extracted statistical features of the flow data. On the ISCX2012 dataset, TR-IDS classified infiltrations, brute-force attacks against SSH, DDoS attacks, and HTTP-DoS attacks, with accuracy as high as 99.13%.

In [48], a block sequence structure was created using packet payload data, and a detection model was designed based on LSTM and CNN models and a multi-head self-attention mechanism, achieving accuracy between 97% and 99% in the detection of 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 a 1D CNN and multi-head attention layers to the payload data. We use separate multi-kernel 1D CNN models for the payload and header data. Instead of employing a single model, we *combine* 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 in previous works. Our ensemble model is now an integral part of the ECSC, which is in charge of monitoring 1.2 million threat cases daily from educational institutions in South Korea. Our model classifies threats with an F1-score of 0.998. Additionally, threats identified with over 99% confidence are automatically registered as attack incidents so that the burden of security agents 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 of network threat detection methods.

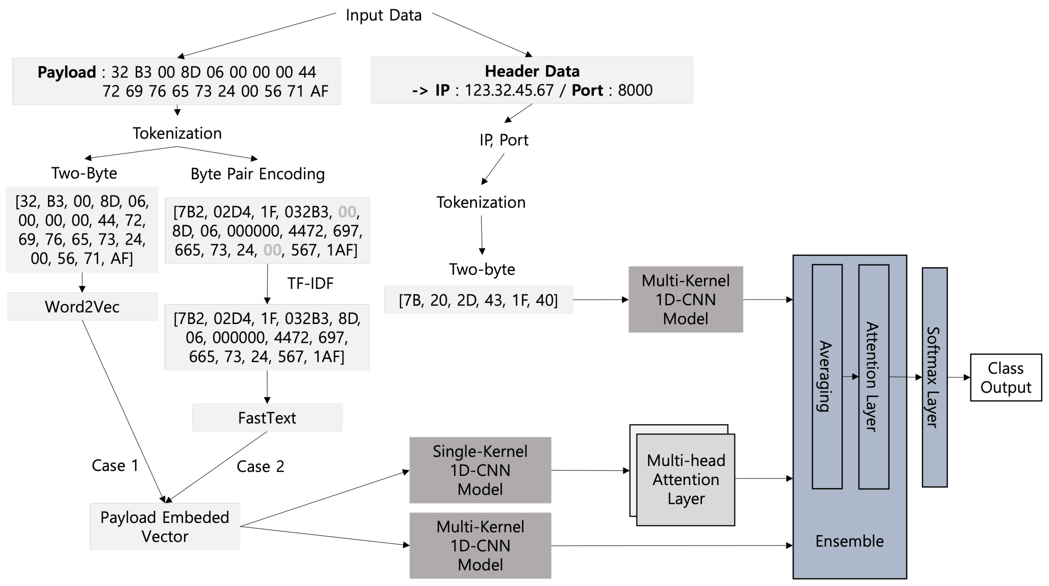
Research Works	**Data**		**Preprocessing**			**Word Embedding**		**Modeling Method**				**Real Data Tested**
	Payload	**Header**	**n-Gram**	**BPE**	**Tf-Idf**	**Word2Vec**	**FastText**	**Text 1D CNN**	**Sequential Models**	**Attention**	**Ensemble**	
[48]	O		Block Sequence Construction			Block Embedding			O	O		
[49]	O			O			O		O	O		
[46]		O				Linguistic Embedding			O			
[47]	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 describe our methodology.

3.1. An Overview of Conv. 1D Multi-Head Attention Ensemble

The modeling structure of our proposed Conv. 1D Multi-Head Attention Ensemble (CMAE) model is shown in Figure 1. We tokenize the payload and header data and apply embedding techniques to generate the input data. The CMAE model receives a total of three inputs for training. The 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 being combined.



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
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]
TF-IDF
[7B2, 02D4, 1F, 032B3, 8D, 06, 000000, 4472, 697, 665, 73, 24, 567, 1AF]
Word2Vec
FastText
Case 1
Case 2
Payload Embeded Vector
IP, Port
Tokenization
Two-byte
[7B, 20, 2D, 43, 1F, 40]
Multi-Kernel 1D-CNN Model
Single-Kernel 1D-CNN Model
Multi-head Attention Layer
Multi-Kernel 1D-CNN Model
Averaging
Attention Layer
Ensemble
Softmax Layer
Class Output

Figure 1. The overall framework for CMAE-based modeling of network threats

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

The output latent vectors from 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.

An explanation of the individual components of the learning structure is provided in the following subsections.

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. In 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 [50].

This study used the following tokenization methods for payload and header data. First, we tokenized the payload data by dividing it into two-byte units and representing it in hexadecimal format ('00', 'ec') [51].

A payload can be represented by 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 [52] 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 [53]. The OOV problem can be solved by using subword segmentation methods [54,55]. For example, the word *birthplace* can be divided into smaller subwords, *birth* and *place*. For subword segmentation, we used byte-pair encoding (BPE) [56] to tokenize the payload data into variable-length n-bytes. We used Sentencepiece, an open-source package that implements the BPE algorithm [57]. However, since the payload data contain 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 the 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 [58]. *Term Frequency (TF)* is a value that shows how often a particular word appears within a document. However, if the same word is frequently used in multiple documents, it may be a special character or word that commonly appears in sentences, such as commas or articles. In such cases, these characters or words may make it difficult to distinguish 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 becomes less relevant, its TF-IDF score approaches zero.

We removed the tokens with the lowest TF-IDF scores from the payload after the BPE-based tokenization was conducted. We found that removing the 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 3000 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 similarities between words and understand contextual features [59,60].

Methods used to represent words as vectors can be divided into sparse representation [61] and distributed representation [62]. 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 the distributed representation method, which 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 between different types of threats in payloads. This problem was later solved using convolution and max-pooling.

3.4.1. Using Word2Vec

Word2Vec is an algorithm that uses distributed representation to obtain the embedding vectors, which has shown outstanding performance in classifying words and documents [63,64]. There are two primary Word2Vec methods: CBOW (Continuous Bag of Words) and skip-gram [65]. CBOW predicts the middle word from the surrounding words.

In the CBOW method, 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 further 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.

Contrary to the CBOW method, the skip-gram method predicts the surrounding words from the center word. This paper employed the skip-gram method, which is generally considered better than the CBOW method [65]. We only applied the Word2Vec algorithm to payloads that had been tokenized into two-byte units. We first randomly initialized a vector for each two-byte word in our vocabulary. Then, we went through each position t and defined the center word at that position as c. To identify the context words, we set a window of size m so that the skip-gram method examined the tokens in position $t - m$ to $t + m$. The skip-gram method tried to maximize the log-likelihood J of the context words given the center word at t in a token sequence θ according to Equation (1).

$$J(\theta) = -\frac{1}{T}\sum_{i=1}^{T}\sum_{-m\leq j\leq m} logP(w_{t+j}|w_t;\theta) \tag{1}$$

The skip-gram method uses deep learning to maximize the log-likelihood using a neural network. We chose log-likelihood because it is easy to compute derivatives during training.

Instead of raw payload bytes, we took the Word2Vec vector of a payload token to learn the context and discern attack intentions more accurately.

3.4.2. Using FastText

The Word2Vec algorithm described earlier has several known drawbacks [66]. First, it does not accurately reflect the morphological characteristics of words. 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 format. Therefore, Word2Vec can be applied without significant problems with sparse words or OOV terms. If the tokens are divided further using the BPE algorithm, the number of tokens increases, allowing for more detailed context information to be extracted. However, the OOV problem may arise. We addressed this issue using a FastText [67] 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 addressed.

FastText extracted the subwords according to the following steps:

1. Initialize the vocabulary with all two-byte tokens.
2. Calculate the frequency of each byte.
3. Repeat the following steps until the desired vocabulary size is reached:

   (a) Find the most frequent pair of consecutive bytes.
   (b) Merge the pair to create a new subword.
   (c) Update the frequency counts of all the bytes that contain the merged pair.
   (d) Add the new subword unit to the vocabulary.
4. Represent the vocabulary of the subword units.

3.5. Text 1D CNN Model

The embeddings of the payloads pass through convolutional neural networks (CNNs) for another round of feature extraction. A CNN is a representative deep learning model for image classification [68]. A CNN consists of convolutional layers and pooling layers. In the convolutional layers, feature extraction of the image is performed by moving an nxm-sized matrix called a *kernel* over the image to extract a feature map from the 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. One-dimensional CNNs can be used for one-dimensional data such as time series and texts. To extract the features of the payload, we applied a 1D CNN model [69].

An example of using a single-kernel 1D CNN is shown in Figure 2. The single-kernel 1D CNN partially tokenizes a payload into a 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'. As the kernel moves down the matrix, it continues to perform convolutional operations until it reaches the bottom, ultimately generating a vector with eight elements. An optimal kernel size can be determined experimentally through a hyperparameter tuning process.

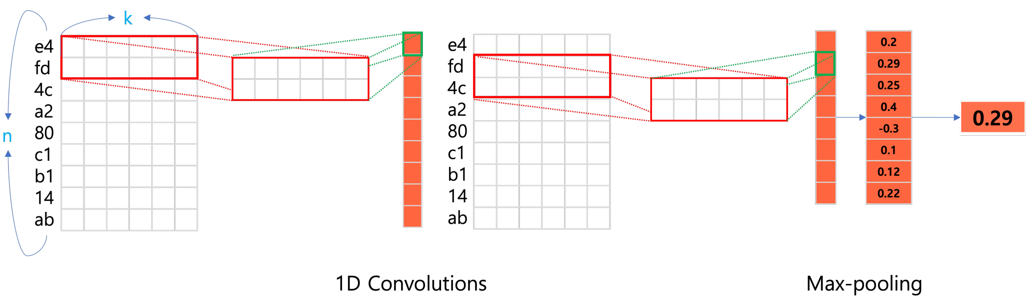


Figure 2. Processing a payload through the 1D CNN model.

Once the convolution is completed, various pooling methods can be applied. The most commonly used is *max-pooling*. Figure 2 shows the max-pooling process, which 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 embeddings pass through two stages of convolution and pooling operations. After the second max-pooling, positional encoding functions are applied to each element according to Equation (2), where the scale value n is empirically set to 10,000 [70].

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

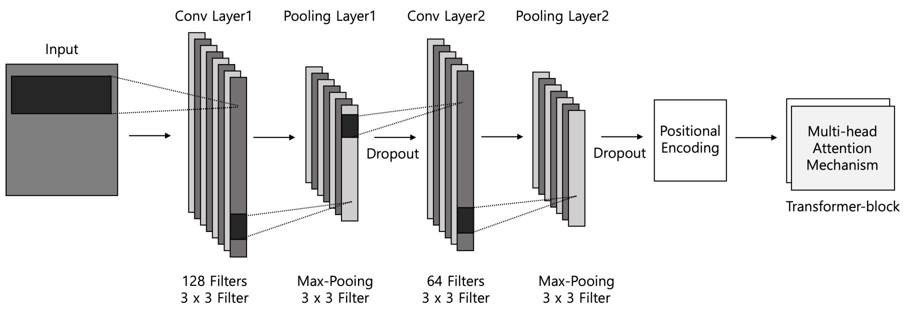


Figure 3. One-dimensional CNN connected to a Transformer block with multi-head attention layers.

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 [70]. Alternatively, we used a 1D CNN model with variable-size kernels, as shown in Figure 4, which uses a filter with a size of 128 and kernels with sizes of 3, 4, and 5. A global max-pooling operation is applied to each intermediate result filtered with variable kernel sizes. The global max-pooling results are concatenated and passed to the dense layer before they reach the ensemble layer, which averages the feature vectors returned by the other models. The multi-kernel structure captures the association among the various payload parts at different scales. This multi-perspective feature analysis through the various kernels is comparable to the mechanism implemented with the multi-head attention layers. The multi-kernel 1D CNN is worthy of being combined with other models.

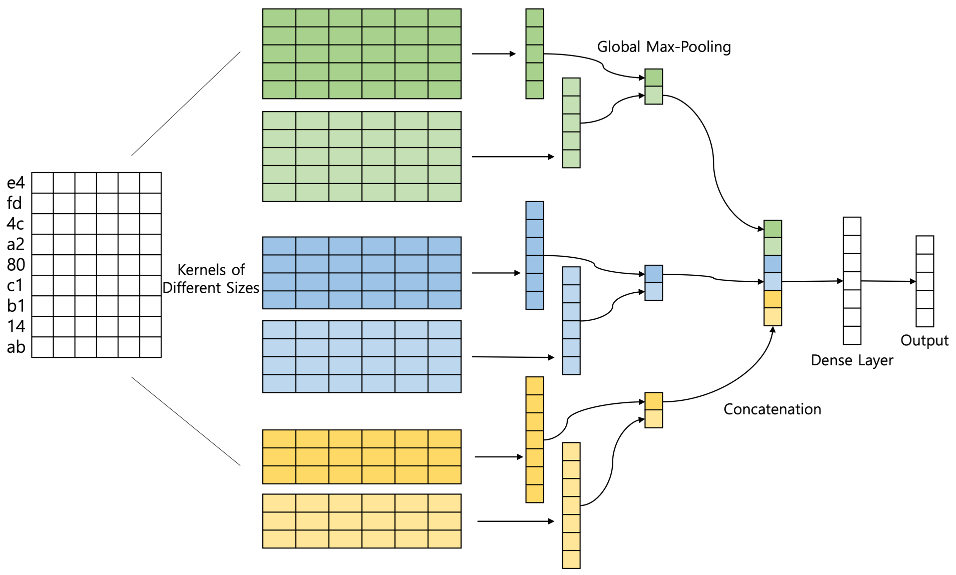


Figure 4. Multi-kernel 1D CNN model architecture.

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 also divided into two-byte tokens.

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 in both single-kernel and multi-kernel CNNs. Not all payloads are 1500 bytes in length. As mentioned earlier, the default values need to be padded for payloads shorter than the default length so that the void can be filled. 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 computational overhead of the subsequent layers, such as the multi-head attention layers and dense layers, is significantly reduced. This performance acceleration is needed to ensure the prompt detection of a threat before it causes substantial damage.

3.6. Multi-Head Attention

Another critical component of our model 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 the 1D CNN, Q, K, and V vectors are created. Each of these vectors undergoes a linear transformation. Given the transformed vectors, we apply the scaled dot-product attention function, as defined in Equation (3) [70]. 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.

$$Attention = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{3}$$

$$head_i = Attention(QW_i^Q KW_i QW_i^Q) \tag{4}$$

$$MultiHeadAttention = Concat(head_1, \ldots, head_h)W^O \tag{5}$$

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 training.

3.7. The Ensemble Layer

Finally, the information learned through the feature extraction method and multiple models is passed to the ensemble block, which consists of an averaging layer [71] and an attention layer [72]. The objective of this unique introduction of the attention layer into the ensemble block is to extract the semantic features of the aggregate embeddings from different models.

The attention weight values are passed to the softmax function [73]. 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 by the ECSC, managed by the Ministry of Education in South Korea. We aimed 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 was generated through simulated attacks in a manner similar to actual attack incidents [17]. The CICIDS2017 dataset includes various attack packets, such as DoS attacks, port-scan attacks, distributed denial-of-service (DDoS) attacks, 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 generated over five days between 3 and 7 July 2017. The class distribution in the CICIDS2017 we used is shown in Table 2. The CICID2017 dataset originally contained 2,830,743 attack instances. However, we used a refined version of the CICIDS2017 dataset, which excludes packets with unintentional missing values [74]. The refined CICIDS2017 dataset contains 2,230,983 attack instances.

Table 2. The class distribution of the CICIDS2017 dataset.

Class	The Number of Cases
Benign attacks	528,265
DoS attacks	250,720
DDoS attacks	124,111
Port-scan attacks	107,778
Attacks against FTP & SSH	13,231
Bot attacks	1905
Web hacking	1648

We used a real dataset from the ECSC to validate the practicality and generality of our model. The ECSC in South Korea monitors massive network traffic flowing in and out of all educational institutions and general hospitals operated by universities. Educational institutions in South Korea are vulnerable to cyber attacks due to the relatively large number of publicly accessible resources.

Figure 5 illustrates an overview of the ECSC. TMS (Threat Management System) appliances are deployed in every educational institution. The ECSC distributes Snort rules to the TMS appliances and updates them periodically. TMS appliances are the first line of defense and filter out cyber threats based on the Snort rules. TMS appliances capture 1500 bytes of payload the moment a threat is detected. These payloads are transferred to the ECSC and stored in ElasticSearch [75]. The CMAE, which is located at the core of the ECSC, retrieves six months' worth of threat incidents every week to undertake the modeling procedure. The fully-trained CMAE model classifies the threats from the TMS appliances in real time. The classification results of the CMAE are automatically reported and escalated for an immediate response if the softmax confidence score is above the configurable threshold, currently set to 99%.

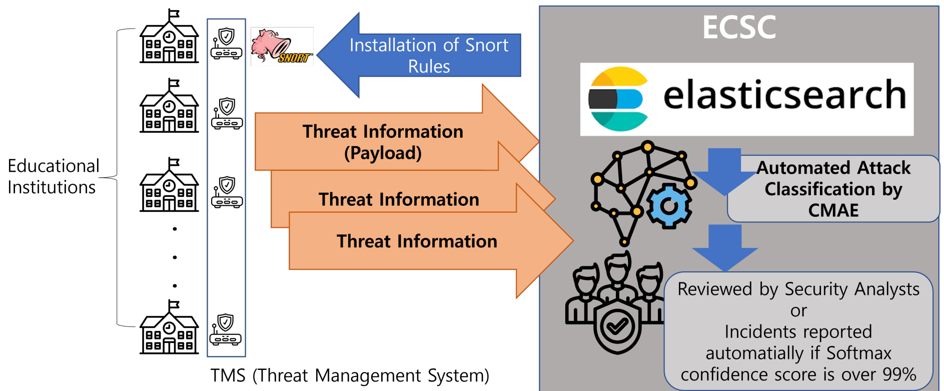

Educational Institutions
Installation of Snort Rules
Threat Information (Payload)
Threat Information
Threat Information
TMS (Threat Management System)
ECSC
elasticsearch
Automated Attack Classification by CMAE
Reviewed by Security Analysts or Incidents reported automatially if Softmax confidence score is over 99%

Figure 5. The overview of the ECSC currently running in South Korea.

On average, the ECSC sees 1.28 million threat events each day. Many of these events have gone unnoticed due to the limited number of security agents at the ECSC. Approximately 60,000 attacks are confirmed annually. Labeling millions of threat instances was infeasible. Instead of complete enumeration, the ECSC filtered out the threats posing the highest risks according to NIST SP 800-30 [76]. Duplicate threats were also removed. This refinement process narrowed down the original dataset to one using the class distribution shown in Table 3. The dataset was constructed between September 2021 and February 2022. We cannot publish the ECSC dataset for security reasons, as it contains payloads with sensitive private information. However, we can show that the ECSC categorizes attacks into one of six types: intrusion attempts, malware infections, web hacking, transit exploitations, DoS attacks, and hacking emails. Packets that are falsely identified as a threat by the Snort rules are labeled as benign packets.

Table 3. The class distribution of the ECSC dataset.

Class	The Number of Cases
Benign attacks	428,120
Intrusion attempts	68,616
Malware infections	5541
Hacking emails	397
Transit exploitations	191
Web hacking	135
DDoS attacks	7

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

4.2. Implementation and Parameter Configuration

We implemented our model using Tensorflow (https://www.tensorflow.org/) and Keras 2.6. Our model was tested on an 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. The NVIDIA DGX-1 ran on Ubuntu 18.04.5 LTS. We executed our model in a container with Docker v19.03.14.

Table 4 lists the main parameters used for the experiment. Note that these parameters were empirically set.

Table 4. Parameters for the CMAE.

Parameter	Setting
n-gram	two-byte
BPE	vocab_size 32,000, 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
Dropout	0.2
Loss function	binary_crossentropy and categorical_crossentropy
Activation function	GELU and softmax
Optimizer	AdaBeliefOptimizer (learning_rate = 5×10^{-4}, epsilon = 1×10^{-16}, weight_decay = 1×10^{-4})
Scheduler	ReduceLROnPlateau (factor = 0.3, patience = 2, min_lr = 1×10^{-5})
Epochs	50
Batch size	32
Early Stopping	5

We used two-byte tokens for the n-gram tokenization. For BPE, the GE refers to the number of words required for training to distinguish the token units. In this paper, we set 32,000 as the vocabulary size. The payload length was set to the maximum found 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 were smaller than 1500 bytes. Thus, these packets may require padding of the default values.

The embedding dimension size was set to 64 for both Word2Vec and FastText. The smallest dimension size without compromising accuracy was 64. With the compact embedding setting, we could reduce the 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 the skip-gram method over the CBOW method for the embedding. Twenty tokens with the least relevancy were removed from payloads according to the TD-IDF score.

For the 1D CNN model, the filter sizes were set to 128 and 64 and the kernel size was set to 3. For the multi-kernel 1D CNN, the filter size was set to 128 and the kernel sizes were set to 3, 4, and 5. The parameters for the 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. The 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 multiclass classification as the loss function. The Gaussian error linear unit (GELU) function [77] was used as the activation function, and softmax was applied to the last classification layer. The AdaBeliefOptimizer [78] was used as the optimization function, and the ReduceLROnPlateau scheduler from Keras was applied to improve performance. We trained over 50 epochs with a batch size of 32. The early stoppage was set to 5.

4.3. Evaluation Metric

In this study, performance was evaluated using the F1-score. The precision is the ratio of true positives to the total number of positive predictions (Equation (6)). The recall is the ratio of true positives to the total number of actual positives (Equation (7)). The F1-score is the harmonic mean of the precision and recall (Equation (8)). We computed class-wise F1-scores.

We used the micro average, as defined in Equation (9), to assess the overall classification performance. The micro average accounts for the imbalance in data between classification classes. Therefore, we chose the micro average over the macro average, which takes the simple average of the F1-scores of all the classification classes:

- TP (true positives): The number of test results that indicate that a threat is classified into the right type.
- TN (true negatives): The number of test results that correctly indicate that a threat is not classified into the wrong type.
- FP (false positives): The number of test results that wrongly indicate that a threat is correctly classified.
- FN (false negatives): The number of test results that wrongly indicate that a threat is not classified as an incorrect type.

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

$$Recall = \frac{TP}{TP + FN} \tag{7}$$

$$F1 - Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{8}$$

$$
\begin{aligned}
Micro\_precision &= \frac{TP_1 + \ldots + TP_N}{TP_1 + FP_1 + \ldots TP_n + FP_n} \\
Micro\_Recall &= \frac{TP_1 + \ldots + TP_N}{TP_1 + FN_1 + \ldots TP_n + FN_n} \\
Micro\ Average &= \\
2 * \frac{Micro\_Precision * Micro\_Recall}{Micro\_Precision + Micro\_Recall} & \\
n\text{: class label} &
\end{aligned} \tag{9}
$$

4.4. Performance Assessment

We compared the performance of various previous models and our new model on the CICIDS 2017 dataset. The performance of the models is summarized in Table 5. The CMAE outperformed previous state-of-the-art algorithms [41–43]. Both RTIDS and the CMAE performed better than approaches not employing a Transformer architecture. The attention mechanism we used was more effective than an approach based mainly on Text-CNN [42] in terms of capturing the structure of malicious packets. However, we took advantage of the pooling mechanism to summarize the packet payload before running it through the attention layer. The combination of Text-CNN, an attention layer, and an ensemble layer helped achieve a 0.8% higher F1-score compared to RTIDS [43].

Table 6 shows the true negatives and true positives for each attack type in the CICIDS2017 dataset. The true negatives indicate the number of benign cases that were correctly classified as non-attacks. Prior to the application of the CMAE, these true negatives were misclassified as attacks according to the imperfect Snort rules used by the ECSC. Therefore, the CMAE can help reduce false positives that divert the attention of security agents away from genuine threats. Table 6 shows that the CMAE accurately identified all attack types, with an F1-score above 0.99.

Table 5. Performance of the classification algorithms using payload data.

Algorithm	Precision	Recall	F1-Score
PayloadEmbeddings [41]	0.953	0.953	0.953
PBCNN [42]	0.982	0.983	0.983
RTIDS [43]	0.983	0.987	0.990
CMAE Binary Classification	0.999	0.999	0.999
CMAE Multiclass Classification	0.998	0.998	0.998

Table 6. Micro performance results of the CMAE using CICIDS2017 dataset.

Type of Attack	True Negatives	True Positives	F1-Score
Port-scan attacks	379,549	107,733	0.9924
DDoS attacks	379,549	124,111	0.9984
Bot attacks	379,549	1905	0.9997
Web hacking	166,715	1658	0.9980
DoS attacks	386,906	250,720	0.9992
Attacks against FTP-SSH	392,699	13,233	0.9987

However, some large malware attached to emails or distributed DoS (DDoS) attacks were misclassified due to lack of context. Large malware can be more effectively distinguished with careful analysis of the file format, such as PE (Portable Executable) for files on Windows or APK for files on Android [79]. Large Transformer models [80] require identifying malware larger than 1500 bytes. However, we may have to trade speed for accuracy with the larger Transformer models.

The rate of occurrence of DDoS attacks in the real world, as shown in Table 3, is relatively low. Due to the small number of samples, learning to identify DDoS events is challenging. Since DDoS events take place over multiple packets over time, we can identify them using the values of the attributes such as Flow Bytes/s, Flow Packets/s, PSH Flags, Fwd URG Flags, Bwd URG Flags, CWE Flag Count, Fwd Avg Bytes/Bulk, Fwd Avg

Packets/Bulk, Fwd Avg Bulk Rate, Bwd Avg, Bytes/Bulk, Bwd Avg Packets/Bulk, and Bwd Avg Bulk Rate. However, we did not use these values due to several missing values in the CICIDS2017 dataset. We can re-address the DDoS issues with more complete network flow information in the future.

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

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

According to Table 7, approaches using the BPE and two-byte tokenization methods achieved similarly high performance. Two-byte tokenization incurred lower computational overhead. Therefore, two-byte tokenization is more suitable for mission-critical classification needs. Two-byte tokenization performed better with a maximum payload length compared to a length of 1500 bytes. We can attribute this outcome to the CMAE's ability to capture the essential characteristics of payloads, even with a smaller vocabulary size.

The classification performance using the ECSC dataset is shown in Table 8. We used two-byte tokenization and Word2Vec embedding, with the payload length fixed at 1500 bytes. The CMAE achieved an F1-score of 0.9966 for the binary classification of payload-only data. The performance of multiclass classification on the payload-only data mirrored that of binary classification. When the encoding of the headers (IPs and port) was included in the input data, the F1-score of multiclass classification improved to 0.9980. These results were consistent with those on the CICIDS2017 dataset, where the CMAE outperformed the state-of-the-art algorithms. Notably, the BPE-based tokenization, TF-IDF-based token removal, and FastText embedding did not perform well.

Table 7. Performance of the CMAE on the CICIDS2017 dataset 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

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

5. Discussion

The CMAE is an integral part of the 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 the ECSC.

According to the ECSC, simple intrusion attempts comprise the majority of 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 within a minute. Therefore, 1.28 million daily threats can be screened within eight minutes. Prior to the CMAE, a large portion of daily threats went unnoticed due to a shortage of staff. The ECSC provides enhanced security with the total screening of all threat events. The labor costs of manual monitoring can be significantly reduced, which helps the ECSC deal with the volume of threat data every day.

However, there is much room for improvement in our model. First, regarding bit rates, the CMAE processes threat data at approximately 30 Mbps when the payload size is set to 1500 bytes. IDS devices deployed in educational institutions in South Korea typically screen 1 Gbps network traffic. To keep up with the line speed and replace the imperfect rule-based classification engines in the IDS devices, the CMAE needs to use dozens more GPUs, which is not a cost-effective solution. In reality, the ECSC uses IDS devices as the first response to threats, and the CMAE works as a secondary classification engine. This security monitoring chain still helps reduce massive threat data that pass unnoticed due to limited human resources. However, because it is infeasible for the 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 using 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 the training quality of the CMAE using 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, even though the CMAE is trained using 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 training data are needed for the pre-trained CMAE to work out of the box. However, collecting real-world data is extremely difficult due to confidentiality and privacy concerns.

The attention mechanism of the CMAE can help security agents identify the parts of the network data that pose 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. The kernel structure and striding mechanism need to be revised to tackle this particular problem with the 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 to maintaining high classification accuracy.

6. Conclusions

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 using 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 the n-grams, word embedding models, and various deep learning networks to obtain the optimal classification system. As a result, even though only payload data were utilized in the CICIDS2017 dataset, it outperformed the state-of-the-art models with an F1-score of up to 0.9997. We also applied our model to a large-scale security surveillance center (the 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 proved to be 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 the various models. Our ensemble model could classify 150,000 threats per minute on an Nvidia V100-32GB GPU machine, thereby significantly relieving the monitoring workload of security agents. Despite these benefits, more research needs to be conducted on improving the generality of the model so that it can work on different premises out of the 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, Y.Y.; Methodology, H.K. and Y.Y.; Software, H.K.; Validation, H.K. and Y.Y.; Formal analysis, H.K. and Y.Y.; Investigation, H.K. and Y.Y.; Resources, Y.Y.; Data curation, H.K. and Y.Y.; Writing—original draft, H.K. and Y.Y.; Writing—review & editing, H.K. and Y.Y.; Visualization, H.K. and Y.Y.; 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).

Data Availability Statement: The CICIDS2017 data we used in this paper is available in https://www.unb.ca/cic/datasets/ids-2017.html. We refined the CICIDS2017 data according to the method explained by [74]. The data collected by KERIS cannot be shared due to the national security policy in South Korea.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Yoon, Y.; Ban, D.; Han, S.W.; Woo, H.U.; Heo, E.; Shin, S.H.; Lee, J.; An, D. Terminal, Cloud Apparatus, Driving Method of Terminal, Method for Processing Cooperative Data, Computer Readable Recording Medium. U.S. Patent 11,228,653, 18 January 2022.
2. Yoon, Y.; Ban, D.; Han, S.; An, D.; Heo, E. Device/cloud collaboration framework for intelligence applications. In *Internet of Things*; Elsevier: Amsterdam, The Netherlands, 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. *Int. J. Inf. Secur.* **2023**, *22*, 833–867. [CrossRef]
4. DailySECU. Kaseya VSA Ransomware Attack. Available online: https://dailysecu.com/news/articleView.html?idxno=133066 (accessed on 11 October 2023).
5. News Directory 3. Apartment Wall Pad Hacking. Available online: https://www.newsdirectory3.com/apartment-wall-pad-hacking-private-life-video-leaked-police-investigation-ministry-of-science-and-technology-must-cover-camera-lens/ (accessed on 11 October 2023).
6. Security Magazine. Log4j Vulnerability Continues to Threaten Enterprise Security. Available online: https://www.securitymagazine.com/articles/97162-log4j-vulnerability-continues-to-threaten-enterprise-security (accessed on 11 October 2023).
7. Kang, D.M.; Yoon, S.H.; Shin, D.K.; Yoon, Y.; Kim, H.M.; Jang, S.H. A Study on Attack Pattern Generation and Hybrid MR-IDS for In-Vehicle Network. In Proceedings of the 2021 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), Jeju Island, Republic of Korea, 13–16 April 2021; pp. 291–294. [CrossRef]
8. Tian, J.; Wang, B.; Guo, R.; Wang, Z.; Cao, K.; Wang, X. Adversarial Attacks and Defenses for Deep-Learning-Based Unmanned Aerial Vehicles. *IEEE Internet Things J.* **2022**, *9*, 22399–22409. [CrossRef]
9. Lee, W.; Stolfo, S.J.; Mok, K.W. Mining in a data-flow environment: Experience in network intrusion detection. In Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, 15–18 August 1999; pp. 114–124.
10. Alaidaros, H.; Mahmuddin, M.; Al Mazari, A. An overview of flow-based and packet-based intrusion detection performance in high speed networks. In Proceedings of the International Arab Conference on Information Technology, Riyadh, Saudi Arabia, 11–14 December 2011; pp. 1–9.
11. Patel, S.K.; Sonker, A. Rule-based network intrusion detection system for port scanning with efficient port scan detection rules using snort. *Int. J. Future Gener. Commun. Netw.* **2016**, *9*, 339–350. [CrossRef]
12. Nasi, E. Bypass Antivirus Dynamic Analysis. Limitations of the AV Model and How to Exploit Them. 2014. Available online: https://wikileaks.org/ciav7p1/cms/files/BypassAVDynamics.pdf (accessed on 11 October 2023).

13. Ito, M.; Iyatomi, H. Web application firewall using character-level convolutional neural network. In Proceedings of the 2018 IEEE 14th International Colloquium on Signal Processing & Its Applications (CSPA), Penang, Malaysia, 9–10 March 2018; pp. 103–106.
14. Hao, S.; Long, J.; Yang, Y. Bl-ids: Detecting web attacks using bi-lstm model based on deep learning. In Proceedings of the International Conference on Security and Privacy in New Computing Environments, Tianjin, China, 13–14 April 2019; pp. 551–563.
15. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv* **2018**, arXiv:1810.04805.
16. 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* **2019**, arXiv:1907.11692.
17. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSP* **2018**, *1*, 108–116.
18. Roesch, M. Snort: Lightweight intrusion detection for networks. In Proceedings of the 13th USENIX Conference on System Administration, Seattle WA, USA, 7–12 November 1999; Volume 99, pp. 229–238.
19. 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. In Proceedings of the NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium, Istanbul, Turkey, 25–29 April 2016; pp. 27–35.
20. Chang, S.; Qiu, X.; Gao, Z.; Qi, F.; Liu, K. A flow-based anomaly detection method using entropy and multiple traffic features. In Proceedings of the 2010 3rd IEEE International Conference on Broadband Network and Multimedia Technology (IC-BNMT), Beijing, China, 26–28 October 2010; pp. 223–227.
21. Muraleedharan, N.; Parmar, A.; Kumar, M. A flow based anomaly detection system using chi-square technique. In Proceedings of the 2010 IEEE 2nd International Advance Computing Conference (IACC), Patiala, India, 19–20 February 2010; pp. 285–289.
22. Wang, K.; Stolfo, S.J. Anomalous payload-based network intrusion detection. In Proceedings of the International Workshop on Recent Advances in Intrusion Detection, Sophia Antipolis, France, 15–17 September 2004; pp. 203–222.
23. 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. [CrossRef]
24. Perdisci, R.; Ariu, D.; Fogla, P.; Giacinto, G.; Lee, W. McPAD: A multiple classifier system for accurate payload-based anomaly detection. *Comput. Netw.* **2009**, *53*, 864–881. [CrossRef]
25. Yoon, Y.; Choi, Y. On Multilateral Security Monitoring and Analysis With an Abstract Tomogram of Network Flows. *IEEE Access* **2018**, *6*, 24118–24127. [CrossRef]
26. Reddy, R.R.; Ramadevi, Y.; Sunitha, K.N. Effective discriminant function for intrusion detection using SVM. In Proceedings of the 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Jaipur, India, 21–24 September 2016; pp. 1148–1153.
27. Zolotukhin, M.; Hämäläinen, T.; Kokkonen, T.; Siltanen, J. Analysis of HTTP requests for anomaly detection of web attacks. In Proceedings of the 2014 IEEE 12th International Conference on Dependable, Autonomic and Secure Computing, Dalian, China, 24–27 August 2014; pp. 406–411.
28. Koc, L.; Mazzuchi, T.A.; Sarkani, S. A network intrusion detection system based on a Hidden Naïve Bayes multiclass classifier. *Expert Syst. Appl.* **2012**, *39*, 13492–13500. [CrossRef]
29. Zhang, Z.; George, R.; Shujaee, K. Efficient detection of anomolous HTTP payloads in networks. In Proceedings of the SoutheastCon 2016, Norfolk, VA, USA, 30 March–3 April 2016; pp. 1–3.
30. Farnaaz, N.; Jabbar, M. Random forest modeling for network intrusion detection system. *Procedia Comput. Sci.* **2016**, *89*, 213–217. [CrossRef]
31. Primartha, R.; Tama, B.A. Anomaly detection using random forest: A performance revisited. In Proceedings of the 2017 International Conference on Data and Software Engineering (ICoDSE), Palembang, Indonesia, 1–2 November 2017; pp. 1–6.
32. 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. [CrossRef]
33. 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. *Mater. Charact.* **2021**, *172*, 110906. [CrossRef]
34. 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. [CrossRef]
35. Kim, T.Y.; Cho, S.B. Web traffic anomaly detection using C-LSTM neural networks. *Expert Syst. Appl.* **2018**, *106*, 66–76. [CrossRef]
36. Wang, W.; Zhu, M.; Zeng, X.; Ye, X.; Sheng, Y. Malware traffic classification using convolutional neural network for representation learning. In Proceedings of the 2017 International Conference on Information Networking (ICOIN), Da Nang, Vietnam, 11–13 January 2017; pp. 712–717.
37. Wang, Y.; An, J.; Huang, W. Using CNN-based representation learning method for malicious traffic identification. In Proceedings of the 2018 IEEE/ACIS 17th International Conference on Computer and Information Science (ICIS), Singapore, 6–8 June 2018; pp. 400–404.
38. 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. *Appl. Sci.* **2021**, *11*, 7050. [CrossRef]
39. 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. [CrossRef]

40. Liu, H.; Lang, B.; Liu, M.; Yan, H. CNN and RNN based payload classification methods for attack detection. *Knowl.-Based Syst.* **2019**, *163*, 332–341. [CrossRef]
41. Hassan, M.; Haque, M.E.; Tozal, M.E.; Raghavan, V.; Agrawal, R. Intrusion detection using payload embeddings. *IEEE Access* **2021**, *10*, 4015–4030. [CrossRef]
42. Yu, L.; Dong, J.; Chen, L.; Li, M.; Xu, B.; Li, Z.; Qiao, L.; Liu, L.; Zhao, B.; Zhang, C. PBCNN: Packet bytes-based convolutional neural network for network intrusion detection. *Comput. Netw.* **2021**, *194*, 108117. [CrossRef]
43. Wu, Z.; Zhang, H.; Wang, P.; Sun, Z. RTIDS: A robust transformer-based approach for intrusion detection system. *IEEE Access* **2022**, *10*, 64375–64387. [CrossRef]
44. Xu, H.; Chen, W.; Zhao, N.; Li, Z.; Bu, J.; Li, Z.; Liu, Y.; Zhao, Y.; Pei, D.; Feng, Y.; et al. Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications. In Proceedings of the 2018 World Wide Web Conference, Lyon, France, 23–27 April 2018; pp. 187–196.
45. Yousefi-Azar, M.; Varadharajan, V.; Hamey, L.; Tupakula, U. Autoencoder-based feature learning for cyber security applications. In Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, USA, 14–19 May 2017; pp. 3854–3861.
46. 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. *Appl. Sci.* **2019**, *9*, 3414. [CrossRef]
47. Min, E.; Long, J.; Liu, Q.; Cui, J.; Chen, W. TR-IDS: Anomaly-based intrusion detection through text-convolutional neural network and random forest. *Secur. Commun. Netw.* **2018**, *2018*, 4943509. [CrossRef]
48. Liu, J.; Song, X.; Zhou, Y.; Peng, X.; Zhang, Y.; Liu, P.; Wu, D.; Zhu, C. Deep anomaly detection in packet payload. *Neurocomputing* **2021**, *485*, 205–218. [CrossRef]
49. Kim, Y.; Ko, Y.; Euom, I.; Kim, K. Web Attack Classification Model Based on Payload Embedding Pre-Training. *J. Korea Inst. Inf. Secur. Cryptol.* **2020**, *30*, 669–677.
50. Grefenstette, G.; Tapanainen, P. What Is a Word, What Is a Sentence? Problems of Tokenisation. 1994. Available online: https://www.dfki.de/~neumann/qa-course/grefenstette94what.pdf (accessed on 11 October 2023).
51. Wang, Y.; Xiang, Y.; Zhou, W.; Yu, S. Generating regular expression signatures for network traffic classification in trusted network management. *J. Netw. Comput. Appl.* **2012**, *35*, 992–1000. [CrossRef]
52. Loper, E.; Bird, S. Nltk: The natural language toolkit. *arXiv* **2002**, arXiv:cs/0205028.
53. Lochter, J.V.; Silva, R.M.; Almeida, T.A. Deep learning models for representing out-of-vocabulary words. In Proceedings of the Brazilian Conference on Intelligent Systems, Rio Grande, Brazil, 20–23 October 2020; pp. 418–434.
54. Zhang, Z.; Zhao, H.; Ling, K.; Li, J.; Li, Z.; He, S.; Fu, G. Effective subword segmentation for text comprehension. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2019**, *27*, 1664–1674. [CrossRef]
55. 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: Berlin/Heidelberg, Germany, 2018; pp. 53–64.
56. Sennrich, R.; Haddow, B.; Birch, A. Neural machine translation of rare words with subword units. *arXiv* **2015**, arXiv:1508.07909.
57. Kudo, T.; Richardson, J. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv* **2018**, arXiv:1808.06226.
58. Aizawa, A. An information-theoretic perspective of tf–idf measures. *Inf. Process. Manag.* **2003**, *39*, 45–65. [CrossRef]
59. Kenter, T.; De Rijke, M. Short text similarity with word embeddings. In Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, Melbourne, Australia, 18–23 October 2015; pp. 1411–1420.
60. Liu, Y.; Liu, Z.; Chua, T.S.; Sun, M. Topical word embeddings. In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, Austin, TX, USA, 25–30 January 2015.
61. Rubinstein, R.; Bruckstein, A.M.; Elad, M. Dictionaries for sparse representation modeling. *Proc. IEEE* **2010**, *98*, 1045–1057. [CrossRef]
62. Howard, M.W.; Kahana, M.J. A distributed representation of temporal context. *J. Math. Psychol.* **2002**, *46*, 269–299. [CrossRef]
63. Goldberg, Y.; Levy, O. word2vec Explained: Deriving Mikolov et al.'s negative-sampling word-embedding method. *arXiv* **2014**, arXiv:1402.3722.
64. Lee, H.; Yoon, Y. Engineering doc2vec for automatic classification of product descriptions on O2O applications. *Electron. Commer. Res.* **2018**, *18*, 433–456. [CrossRef]
65. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient estimation of word representations in vector space. *arXiv* **2013**, arXiv:1301.3781.
66. Horn, F. Context encoders as a simple but powerful extension of word2vec. *arXiv* **2017**, arXiv:1706.02496.
67. Bojanowski, P.; Grave, E.; Joulin, A.; Mikolov, T. Enriching word vectors with subword information. *Trans. Assoc. Comput. Linguist.* **2017**, *5*, 135–146. [CrossRef]
68. Zeiler, M.D.; Fergus, R. Visualizing and understanding convolutional networks. In Proceedings of the European Conference on Computer Vision, Zurich, Switzerland, 6–12 September 2014; pp. 818–833.
69. Azizjon, M.; Jumabek, A.; Kim, W. 1D CNN based network intrusion detection with normalization on imbalanced data. In Proceedings of the 2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), Fukuoka, Japan, 19–21 February 2020; pp. 218–224.

70. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 6000–6010.
71. Mehrkanoon, S. Deep neural-kernel blocks. *Neural Netw.* **2019**, *116*, 46–55. [CrossRef]
72. Bahdanau, D.; Cho, K.; Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv* **2014**, arXiv:1409.0473.
73. Luong, M.T.; Pham, H.; Manning, C.D. Effective approaches to attention-based neural machine translation. *arXiv* **2015**, arXiv:1508.04025.
74. Alrowaily, M.; Alenezi, F.; Lu, Z. Effectiveness of machine learning based intrusion detection systems. In Proceedings of the International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage, Atlanta, GA, USA, 14–17 July 2019; pp. 277–288.
75. Elasticsearch, B. Elasticsearch. Software, Version 6. 2018. Available online: https://www.elastic.co/kr/downloads/past-releases/elasticsearch-6-0-0 (accessed on 11 October 2023).
76. Stoneburner, G.; Goguen, A.; Feringa, A. *Risk Management Guide for Information Technology Systems*; NIST Special Publication 800-30; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2002.
77. Hendrycks, D.; Gimpel, K. Gaussian error linear units (gelus). *arXiv* **2016**, arXiv:1606.08415.
78. 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. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 18795–18806.
79. Lee, K.W.; Oh, S.T.; Yoon, Y. Modeling and Selecting Optimal Features for Machine Learning Based Detections of Android Malwares. *KIPS Trans. Softw. Data Eng.* **2019**, *8*, 427–432.
80. Dong, Z.; Tang, T.; Li, L.; Zhao, W.X. A survey on long text modeling with transformers. *arXiv* **2023**, arXiv:2302.14502.