IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 30, 2021, accepted September 2, 2021, date of publication September 15, 2021,
date of current version September 27, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3113124

Hunt for Unseen Intrusion: Multi-Head Self-Attention Neural Detector

SEONGYUN SEO[1], SUNGMIN HAN[1], JANGHYEON PARK[1], SHINWOO SHIM[2], HAN-EUL RYU[2], BYOUNGMO CHO[2], AND SANGKYUN LEE[1]

[1]School of Cybersecurity, Korea University, Seoul 02841, Republic of Korea
[2]Cyber Warfare Research and Development Laboratory, LIG Nex1, Seongnam-si 13488, Republic of Korea

Corresponding author: Sangkyun Lee (sangkyun@korea.ac.kr)

This work was supported by LIG Nex1.

ABSTRACT A network intrusion detection (NID) system plays a critical role in cybersecurity. However, the existing machine learning-based NID research has a vital issue that their experimental settings do not reflect real-world situations where unknown attacks are constantly emerging. In particular, their train and test sets are from a single data set, which inevitably overestimates the detection power since all test attack types are known in training, and test cases will have similar characteristics to the training data. This paper introduces a new strategy to constitute test data with updated traffic with attack types not included in training data. In the proposed setting, the prediction accuracy of the existing detectors is dropped by about 20% compared to what has been reported. Also, in-depth analysis of detection performance by attack types has revealed that the existing models have strength at certain attack types but struggle to detect the other attack types such as DoS, DDoS, web attack, and port scan. To overcome the issues, we propose a new neural detector, called MHSA, based on a multi-head self-attention mechanism whose architecture suits better to capture scattered pieces of evidence in network traffic. Our model improved the overall detection performance by 29% in false positive rate at the true positive rate of 0.9 and by 9% in AUC over the current state-of-the-art models, successfully detecting the attacks that are not well captured before. Furthermore, we show that our proposed MHSA model even outperforms the best ensemble detector constructed by joining the state-of-the-art classifiers.

INDEX TERMS Deep neural network, intrusion detection, multi-head attention, realistic prediction performance evaluation, self-attention.

I. INTRODUCTION

With the massive increase in connectivity and the expansion of the attack surface, the risk and severity of network security threats have grown dramatically. The network intrusion detection (NID) system is a popular measure for defending against network security threats by monitoring network traffic and detecting malicious activities. Recently, machine learning and deep learning-based methods have been proven effective in creating NID systems with high accuracy and availability. A variety of studies have been conducted on designing effective machine learning and deep learning approaches for NID systems in detecting malicious network traffic [1]–[21], [41]–[48], [55], [58].

Despite many existing studies on applying machine learning and deep learning to NID systems, there are only a few considerations in designing realistic experiment settings to properly evaluate the detection performance of NID models. First, test data consists of the same attack types as in training data in the previous studies. As a result, the trained models are tested only against attack types known in training time, inevitably limiting accurate evaluation of prediction performance in the real world where unseen kinds of attacks may exist. In real-life NID systems, precise detection of emerging or zero-day attacks is a critical capability. Second, attack patterns tend to change over time in the actual situation, and therefore it will be better to consider separate data sets for training and testing where more recent attacks reside in the test set. In fact, this reflects a common strategy for data collection in intrusion detection studies – for example, IDS-2017 (also known as CIC-IDS-2017) [30] contains up-to-date attacks that may not be included in the IDS-2012 (a.k.a. ISCX-IDS-2012) [29] data set. For a more realistic performance evaluation, we suggest a new strategy to use

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

TABLE 1. Prediction accuracy of the state-of-the-art malicious traffic detectors evaluated in different experimental settings. In their original experiments, training and test data are from a single data set. A more realistic evaluation where the test set contains newer and different attacks shows much lower accuracy values than what has been reported in the original literature.

Model	Setup	Train data	Test data	Accuracy
E-GRU	[39]	IDS-2012	IDS-2012	99.99%
	Realistic	IDS-2012	IDS-2017	78.26%
HA	[17]	IDS-2012	IDS-2012	99.95%
	[17]	IDS-2017	IDS-2017	99.45%
	Realistic	IDS-2012	IDS-2017	77.69%
CNN-LSTM	[20]	IDS-2017	IDS-2017	99.91%
	Realistic	IDS-2012	IDS-2017	82.68%

an old data set (IDS-2012) for training and a newer data set (IDS-2017) for testing, where the test set includes unseen attack types (for example, botnet and web attacks) and more recent attacks. Our strategy will allow us a more realistic evaluation of NID models for detecting future attacks.

The need for a better evaluation strategy can be seen more concretely in Table 1. The table shows the prediction accuracy values of the deep learning-based state-of-the-art malicious traffic detectors, encoded gated recurrent unit (E-GRU) [39], hierarchical attention (HA) [17], and convolutional neural network long short-term memory (CNN-LSTM) [20], reported in their original literature, where the training and the test data are from a single data set (either IDS-2012 or IDS-2017). When we try a more realistic setup where we train with IDS-2012 and test with IDS-2017, we can see that prediction accuracy values have dropped significantly, indicating that the original test scores may overestimate the detection performance on future attacks.

Our new analysis also hints that the existing methods may have certain weaknesses in detecting different types of attacks. In Table 2, we present the detection rates of the detection models for each attack type. The numbers indicate that the state-of-the-art models have competent performance on specific attack types but can be inadequate to identify the other attacks such as denial of service (DoS), distributed denial of service (DDoS), web attack, and port scan. Some of these attacks are comprised of seemingly legitimate packets, making them hard to detect. For example, in a Slowloris attack [49], the attacker sends a partial HTTP header to a victim. The action causes the victim to keep the connection open, as the victim does not start processing the HTTP request until it receives the complete header. Then, the attacker sends the next portion of the header periodically before the time out is reached so that the victim cannot release the resources allocated to it. In this way, the attacker maintains many connections simultaneously and uses up the system resources of the victim. That is, the attacker occupies all available connection sockets, and all legitimate connection attempts are blocked. However, it is challenging to detect these types of attacks, as the evidence of malicious behaviors exists in various forms.

For improving the detection performance on the attack types mentioned above, which are stealthy by design, it is necessary to analyze the behaviors of packets from a variety of perspectives. As shown in Table 2, the recurrent neural network models are not enough to capture multiple aspects of representations from packets. We propose to use the multi-head self-attention (MHSA) mechanism [22] to get packet information from multiple perspectives. That is, the mechanism allows our model to use information selectively from multiple representation subspaces at different positions simultaneously. This will allow our model to achieve higher detection performance than the existing models on DoS, DDoS, port scan, and web attacks, as we discuss later in the experiments.

Table 2 also shows an important fact that each model has its own strong points and weaknesses in detecting certain types of attacks. For example, the Bi-LSTM model is outstanding in identifying the DoS, infiltration, and botnet attacks, whereas the E-GRU model outperforms all other models on detecting the DDoS attacks. To combine the strengths of the models, a typical approach is to construct an ensemble model based on the state-of-the-art detectors or the attack-type best models. We will show later that our proposed MHSA detector even outperforms the best ensemble model with a significant margin.

Our contributions in this work can be summarized as follows:

- We introduce a new evaluation strategy using different data sets in training and testing, where the test set can have updated patterns and new types of attacks. This strategy better reflects real intrusion detection scenarios where the development depends on archived data and attackers keep trying to change traffic patterns and invent new attacks to avoid detection.
- We propose the MHSA neural network as a better alternative to the state-of-the-art models, especially for detecting DoS, DDoS, port scan, and web attacks that are not well captured by the existing methods.
- Our proposed MHSA model, a single detector, defeats the best ensemble model based on the state-of-the-art detectors where each base model has its own strength in identifying different attack types.

The rest of this paper is organized as follows. Section II introduces the existing machine learning and deep learning models for NID systems. Section III provides the details of our proposed MHSA model and attention mechanisms, which play a crucial role in our architecture. Also, we describe how to construct ensemble models. In Section IV, we evaluate the prediction performance of our model in a realistic setting where unseen or updated attacks may exist, comparing it to the state-of-the-art detectors. Section V concludes the paper.

II. RELATED WORK

The network intrusion detection (NID) system, a key component of cybersecurity, protects critical assets by identifying anomalous behaviors in the network traffic. Depending on the

TABLE 2. The detection rates of state-of-the-art models to each attack type. All models are trained on the IDS-2012 data set and tested on the IDS-2017 data set. The boldface numbers indicate the best detection rate in each attack type.

	Benign	Brute Force	DoS	Heartbleed	Infiltration	Web Attack	DDoS	Port Scan	BotNet
Bi-LSTM	67.40%	48.73%	**58.30%**	90.91%	**100.00%**	5.30%	36.42%	72.76%	**82.66%**
E-GRU	81.94%	38.28%	19.24%	63.64%	45.71%	51.01%	**94.41%**	89.55%	26.71%
HA	85.00%	**83.92%**	9.17%	**100.00%**	71.43%	**78.32%**	2.28%	**98.03%**	20.39%
CNN-LSTM	**90.39%**	30.32%	15.67%	0.00%	2.86%	13.10%	16.26%	95.27%	12.06%
Total no. of instances	2,103,377	8,155	176,765	11	35	2,076	85,217	158,740	1,932

detection method, the NID systems can be largely categorized into signature-based and behavior-based NID systems [33]. The signature-based NID system [34] aims to detect malicious network activities by examining specific patterns in network traffic. Despite their popularity, this method tends to be insufficient for detecting unknown attacks for which patterns have not been analyzed. The behavior-based NID system, also known as an anomaly-based NID system [35], analyzes the network traffic and identifies malicious activities from potential attacks. Unlike the signature-based NID system, the behavior-based NID system performs better at identifying unseen attacks. With the growing complexity and variety of software, new vulnerabilities are emerging continuously. Moreover, easy access to advanced technology such as artificial intelligence is contributing to the rapid evolution of network security threats. In this situation, the principal focus of the research and development in the field of NID has become the behavior-based NID system [35], where the machine learning-based [1]–[8], [41]–[45] and deep learning-based approaches [9]–[21], [46]–[48] have been popular.

Machine learning algorithms such as support vector machine (SVM) [36], random forest [37] and k-nearest neighbor algorithm (k-NN) [38] have been actively adopted for NID systems in classifying network attacks. Ikram and Cherukuri [1] employed SVM with principal component analysis (PCA) for detecting malicious network traffic. The automatic parameter selection technique is used to optimize the SVM parameters, such as the kernel parameters and the soft margin parameter in their model. In [2], Vijayananda *et al.* proposed a NID system in wireless mesh networks. The proposed system consists of multiple SVM classifiers with a genetic algorithm-based feature selection method. The individual SVM classifiers are assigned to detection of each type of attack, trained with informative features of each attack type selected by the proposed feature selection technique. In [6], Ingre *et al.* proposed an NID model using decision trees. The proposed model uses the correlation-based feature selection method to reduce the number of features. Li *et al.* [8] presented a multi-layers anomaly detection model which extracts features from different network layers. For reducing the redundancy and noise caused by information from multiple network layers, they proposed an algorithm with the combination of PCA and random forest. Additionally, k-NN [5] and clustering methods [4] have been used to detect network attacks.

Recently, as deep learning has shown good performance in a variety of fields such as image recognition [25] and machine translation [22], there have also been various studies for applying deep learning in the NID task. Among them, convolutional neural networks (CNNs) and recurrent neural networks (RNNs) have achieved remarkable results. CNN models [11]–[13], [46]–[48] detect network attacks by capturing spatial patterns from network traffic data converted to grayscale images. On the other hand, RNN models [14]–[20] classify network attacks by extracting temporal patterns in network traffic data. In [48], Wang *et al.* used the two-dimensional CNN architecture to detect malicious network traffic. Also, Xiao *et al.* [11] proposed an end-to-end encrypted traffic classification method with the one-dimensional CNN. It was shown that the proposed method achieved performance improvement compared to the two-dimensional CNN model and the machine learning-based model. Zhou *et al.* [12] presented the NID model based on the LeNet-5 architecture. Their model achieved over 99% accuracy on the Moore data set [54]. In addition, there have been numerous studies that treat network traffic data as time-series data and use RNNs to identify malicious network traffic. Hwang *et al.* [26] proposed a packet-based NID model using a novel word embedding mechanism and long short-term memory (LSTM). The word embedding mechanism extracts the semantic meaning of a packet, and LSTM networks capture sequential information in a packet for detecting attacks. In [17], Han *et al.* applied the hierarchical attention network (HAN) [27] to the NID system, where HAN is composed of bidirectional gated recurrent units (GRUs) and attention layers. Using attention mechanisms, the model can focus on the crucial parts of network traffic, which improves the detection performance to over 99% on the IDS-2012 data set and IDS-2017 data set. Furthermore, the models combining CNNs with RNNs have been proposed to improve the detection performance. In [19], Wang *et al.* presented a novel intrusion detection system called hierarchical spatial-temporal features-based intrusion detection system (HAST-IDS), which is composed of CNNs for learning low-level spatial features of network traffic and LSTM networks for learning high-level temporal features. In [21], Su *et al.* designed a traffic anomaly detection model which consists of multiple convolutional layers, a bidirectional LSTM layer, and an attention mechanism. The convolutional layers capture local features of network traffic, whose outputs are fed into the bidirectional LSTM layer to learn time-series features in

packets and build packet embedding vectors. Then, the attention layer analyzes the important degree of packet embedding vectors to obtain fine-grained features which are more salient for detecting malicious network traffic.

Also, there are several studies employing ensemble methods to improve the robustness and detection performance of NID systems by combining multiple detectors. For example, Moustafa *et al.* [55] proposed the AdaBoost [59] ensemble classifier composed of decision trees, naive Bayes models, and artificial neural networks. The Adaboost ensemble model achieved over 95% detection rate on UNSW-NB15 [56] and NIMS botnet [57] data sets with simulated internet of things sensor data, showing better predictive power than the individual classifiers included in the ensemble. Furthermore, in [58], Sharma *et al.* proposed another ensemble-based NID system using extreme learning machines (ELMs) [60], where a dedicated ELM is trained to handle a specific type of attack using a different subset of features chosen per attack type.

Although the state-of-the-art models have reported outstanding detection performance, our experiment tells that they may not be satisfactory for detecting all types of attacks, as shown in Table 2. This observation has motivated us to propose a neural detector based on multi-head self-attention, capturing the scattered evidence in multiple forms from network packets to improve the detection performance on the attack types that the existing models cannot identify. Also, current literature has evaluated detection performance using training and test sets from a single data set, which we believe is not enough to reflect the real-world environment where unknown types of attacks may exist.

In addition, we construct ensemble models to enhance the overall detection performance by strengthening the weak points of single models by combining selected single models advantageous for different attack types. We show that, however, our suggested single neural detector MHSA even outperforms the best ensemble model, in both attack-type detection rate and overall detection performance.

III. METHODOLOGY

In this section, we describe our proposed detection model using multi-head self-attention and attention mechanisms (we denote our model as MHSA). In addition, we explain the ensemble method that combines the strength of each model in detecting different types of attacks.

A. MODEL ARCHITECTURE

The MHSA model is a deep neural network for network intrusion detection that takes a flow as an input in the form of a $p \times q$ dimensional matrix where p and q are the numbers of packets and bytes, respectively. The MHSA model is essentially a binary classifier and outputs the probability that the given flow will be malicious.

Fig. 1 shows the overall architecture of our proposed MHSA detector model. It has four modules: byte encoder, packet encoder, flow encoder, and network traffic classifier. The byte encoder is intended to capture multiple aspects

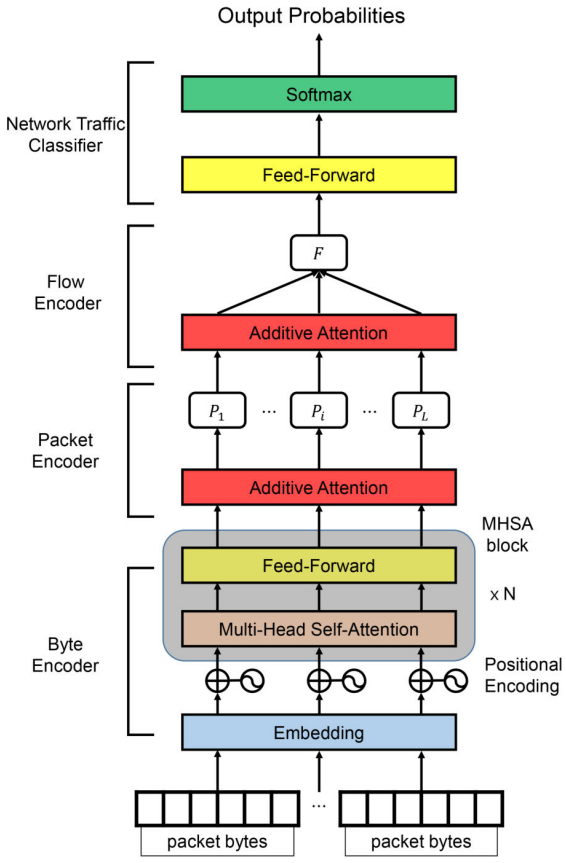


FIGURE 1. The overall architecture of the proposed MHSA (multi-head self-attention) model. The network traffic classifier detects an attack based on a flow-embedding vector constructed using a byte encoder, packet encoder, and flow encoder.

of representations from packets, whose architecture is the same as the encoder part of the Transformer network [22]. The packet encoder builds packet embedding vectors through weighted sums of byte embedding vectors, the outputs of the byte encoder, where the weights correspond to the importance of bytes. In the same way, the flow encoder constructs a flow embedding vector by highlighting the critical packets. Lastly, the network traffic classifier determines whether the network traffic is benign or malicious using the flow embedding vector.

1) BYTE ENCODER

The purpose of the byte encoder is to convert bytes of packets to vector representations. It contains three steps. Firstly, the byte embedding layer transforms a sequence of bytes into a series of embedding vectors. Second, the positional encoding layer injects relative position information into the sequence of byte embedding vectors. In other words, the byte sequence of the ith packet $[b_{i1}, \ldots, b_{iM}]$, where M is the length of a packet, is converted into the sequence of vectors $E_i = [e_{i1}, \ldots, e_{iM}]$, where $e_{ij} \in \mathbb{R}^{d_e}$ is the embedding vector corresponding to the jth byte of the ith packet and d_e is the embedding dimension we obtain after the byte embedding layer and the positional encoding layer.

Lastly, the MHSA blocks are applied repeated N times to the sequence of vectors E_i. The block is composed of a multi-head self-attention layer and a feed-forward layer, depicted as a gray box in Fig. 1. Additionally, residual connection and layer normalization are applied to the outputs of each layer in the MHSA blocks for regularization. To be more specific, an MHSA block can be formalized as follows:

$$X = \text{MultiHead}(Q, K, V),$$
$$\text{FFN}(X) = \max(0, XW_1 + b_1)W_2 + b_2,$$

where $Q \in \mathbb{R}^{M\times d_e}$, $K \in \mathbb{R}^{M\times d_e}$, and $V \in \mathbb{R}^{M\times d_e}$ are the sequence of byte vectors, $W_1 \in \mathbb{R}^{d_e\times d_h}$, $b_1 \in \mathbb{R}^{d_h}$, $W_2 \in \mathbb{R}^{d_h\times d_e}$, and $b_2 \in \mathbb{R}^{d_e}$ are projection parameters, and d_h is the dimension of the hidden layer. For simplicity, the residual connection and layer normalization are omitted in Fig 1. We denote the output of the MHSA blocks on the ith packet as $[v_{i1}, \ldots, v_{iM}]$, where $v_{ij} \in \mathbb{R}^{d_e}$ is the vector representation corresponding to the jth byte of the ith packet. Using the MHSA blocks, the byte encoder captures the multiple scattered evidence of attacks from packets.

2) PACKET ENCODER

The packet encoder builds the packet embedding vectors by highlighting the informative bytes and fading out the meaningless bytes. The packet embedding vector is computed by a weighted sum of byte vectors, where the weights, representing the significance of bytes, are assigned by the additive attention mechanism. The ith packet P_i is described as follows:

$$P_i = \sum_{j=1}^{M} \alpha_{ij} v_{ij},$$

where P_i is the d_e-dimensional vector, α_{ij} is the attention weight assigned to jth byte of the ith packet.

3) FLOW ENCODER

Not all packets of the network flow are equally critical to detect the attack. Therefore, the flow encoder constructs the flow embedding vector by reflecting the importance of packets. The flow embedding vector is computed by a weighted sum of packet embedding vectors, where the weights are generated in the same way as the packet encoder. The flow embedding vector F is computed as follows:

$$F = \sum_{i=1}^{L} \alpha_i P_i,$$

where the dimension of F is d_e and L is the number of packets that compose the network flow. F is the flow embedding vector that contains all information of the entire network traffic.

4) NETWORK TRAFFIC CLASSIFIER

The network traffic classifier identifies malicious network traffic with the flow embedding vector F. The classifier is composed of a feed-forward layer with the softmax function:

$$Y = \text{softmax}(FW_f + b_f),$$

where $W_f \in \mathbb{R}^{d_e\times d_c}$ and $b_f \in \mathbb{R}^{d_c}$ are projection parameters and d_c is the number of classes. Y represents the probabilities for each class.

B. ATTENTION MECHANISM

Our proposed model makes use of the attention mechanism, which gives the ability to the classifier to use input information selectively by discovering and focusing on important parts of the inputs. The mechanism has provided improved accuracy in the fields of computer vision [53] and natural language processing [22], [52], thereby getting more interest from researchers.

1) ADDITIVE ATTENTION

Additive attention highlights the significant elements of an input sequence, where the importance of elements is determined by attention weights. For a given sequence of vectors $\mathrm{x} = (x_1, \ldots, x_n)$, where n is the length of a vector sequence, attention weight α_i is computed as follows:

$$u_i = \tanh(W^T x_i + b),$$
$$\alpha_i = \frac{\exp(u_i^T c)}{\sum_j \exp(u_j^T c)}, \quad (1)$$

where $x_i \in \mathbb{R}^d$ is a vector representation of the ith element, and $W \in \mathbb{R}^{d\times w}$, $b \in \mathbb{R}^w$, and $c \in \mathbb{R}^w$ are parameters in an additive attention network. As shown in (1), the hidden representation u_i is built by applying linear transformation to a vector x_i. Then, the attention weight is assigned to a vector based on the similarity of u_i with a context vector c. The context vector c can be considered as a fixed query asking for which vectors are significant in the input sequence. The final representation of the input sequence is formulated as follows:

$$X = \sum_{i=1}^{n} \alpha_i x_i.$$

2) SELF-ATTENTION

Self-attention is a kind of attention mechanism that captures the relatedness between different positions of a single sequence and represents each position with an expressive vector. That is, a self-attention network takes a sequence of vectors $\mathrm{x} = (x_1, \ldots, x_n)$ as an input and captures interactions between x_i and other vectors of the input sequence. Then, a self-attention network outputs a sequence of vectors $\mathrm{z} = (z_1, \ldots, z_n)$ that contains contextual information.

3) SCALED DOT-PRODUCT ATTENTION

An attention mechanism can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is generated using a weighted sum of values, where each weight is obtained by an alignment model that takes the query

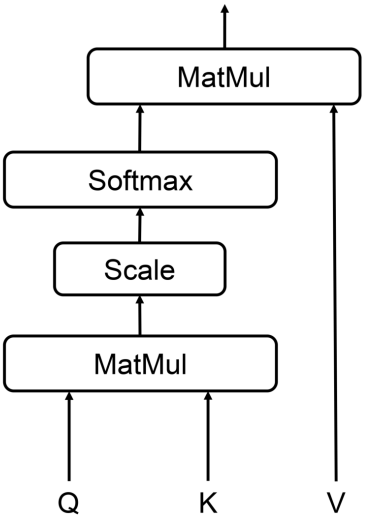


FIGURE 2. The schematic of the scaled dot-product attention. The output is computed by a weighted sum of the values, where the weights are determined by the dot-product of the query with all the keys.

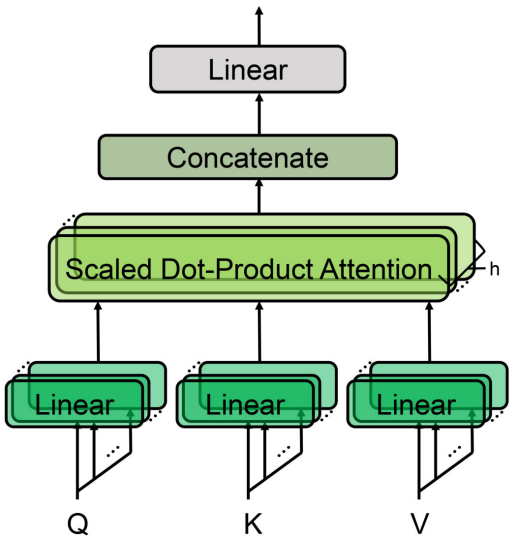


FIGURE 3. The schematic of the multi-head attention. The attention mechanisms are performed multiple times in parallel.

and the corresponding key as inputs. The attention weights represent the relative importance of each pair of key-value to the particular query.

In scaled dot-product attention, the dimensions of a query and a key are d_k, and the dimension of a value is d_v. As shown in Fig. 2, scaled dot-product attention is composed of four steps. In the first step, performing dot-product of a query and all keys, it estimates the significance of each key. The second step is dividing the results of dot-product by $\sqrt{d_k}$. The scaling prevents the arguments of the softmax function being too large, which will result in extremely small gradients. In the next step, the softmax function is applied to get attention weights. The last step multiplies each attention weight with the corresponding value. Scaled dot-product attention processes a set of queries simultaneously for efficiency. Thus, a set of queries is packed into the matrix Q. The keys and values are also packed into matrices K and V. Finally, the matrix of outputs is computed as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V.$$

4) MULTI-HEAD ATTENTION

Performing an attention mechanism multiple times on linearly projected queries, keys, and values is known to be more effective than a single attention mechanism [22]. Fig. 3 shows such multi-head attention linearly projects the queries, keys, and values h times using different, learned linear projections. In this way, d_{model}-dimensional queries, keys, and values are mapped into d_k, d_k, and d_v dimensional vector spaces, respectively. Then, the attention mechanism is performed on each of the linearly projected queries, keys, and values in parallel. Finally, the outputs of the attention mechanisms are concatenated and projected. The multi-head attention is formulated as follows:

$$\begin{aligned}\text{MultiHead}(Q, K, V) &= \text{Concat}(H_1, \ldots, H_h)W^O,\\ H_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V),\end{aligned}$$

where $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$, and $W^O \in \mathbb{R}^{hd_v \times d_{model}}$ are the projection matrices. Here H_i represents the output of a single attention function that takes linearly projected queries, keys, and values as inputs. By performing a single attention function h times on different projected versions of queries, keys, and values, multi-head attention can analyze the input from various aspects. We hypothesize that this construction will be useful in detecting small traces of attacks scattered in network packets.

5) POSITIONAL ENCODING

Positional encoding incorporates the positional information into sequence data. Models such as multi-head attention networks, which do not have recurrent and convolutional architecture, cannot take order into account. Therefore, it is necessary to inject relative or absolute position information into the sequence data for the models to make use of the order of the sequence. Sinusoidal positional encoding is one of the methods that solve this problem. Sinusoidal positional encoding adds a vector composed of sine or cosine values to each element of sequence data based on its position. The vectors for sinusoidal positional encoding are computed as follows:

$$\begin{aligned}\text{PE}_{(pos,2i)} &= \sin(pos/10000^{2i/d}),\\ \text{PE}_{(pos,2i+1)} &= \cos(pos/10000^{2i/d}),\end{aligned}$$

where pos is the position of each element, i is the position of the dimension, and d is the dimension of the element. This

way, the model can infer the relative position of each element of sequence data.

C. ENSEMBLE METHOD

The ensemble is a practical technique to build a predictive model by combining multiple models, so to improve the performance and robustness of classifiers [51]. For simplicity, we consider the voting ensembles based on the outputs of selected models. Depending on the basis for the prediction, voting ensembles can be largely categorized into hard voting and soft voting ensembles.

1) HARD VOTING

Hard voting, also known as majority voting, predicts the class $\hat{y}$ that is chosen by a majority of classifiers. The hard voting is described as follows:

$$\hat{y} = \text{mode}(C_1(x), \ldots, C_n(x)).$$

Here, $C_i(x)$ represents a class predicted by the individual classifier i, and n is the number of classifiers.

2) SOFT VOTING

In soft voting, the predicted probabilities from individual classifiers for each class are summed. Then, soft voting predicts the class with the largest value. Soft voting is defined as follows:

$$\hat{y} = \arg\max_j \sum_{i=1}^{n} C_i^j,$$

where C_i^j represents the predicted probability of the ith classifier for the jth class.

In our experiment, soft voting was possible since all models provided probabilities of being benign or malicious traffic, and soft voting generated ensemble models with better prediction performance than hard voting.

3) ENSEMBLE OF ATTACK-TYPE BEST PREDICTION MODELS

Ensembles can be constructed using all available models; however, it may weaken the strength of each model by merging too many opinions. When some base prediction models are good at detecting certain attack types, it would be better to mix only a few selected base classifiers having strength on perhaps non-overlapping attack types.

IV. EXPERIMENTS

This section evaluates the detection performance of the multi-head self-attention (MHSA) model and the state-of-the-art models on our new experiment setting, which is designed to reflect a real-world environment where unknown attacks are constantly emerging. We also carry out further analysis on the number of misclassified samples by attack types to figure out the strong points and weaknesses of each model. Finally, we compare the detection performance of our proposed MHSA model with that of ensemble models that integrate the strengths of individual detectors.

TABLE 3. Statistics of the IDS-2012 data set.

Flow Type	Count	Percentage
Benign	2,769,358	97.95%
Infiltration	9,920	0.35%
HTTP DoS	2,768	0.10%
DDoS	37,303	1.32%
SSH Brute Force	7,856	0.28%
Total	2,827,205	100%

TABLE 4. Statistics of the training set (a random subset of IDS-2012) used for training and validation.

Flow Type	Train		Validation	
	Count	Percentage	Count	Percentage
Benign	19,385	32.37%	8,308	32.37%
Infiltration	6,944	11.60%	2,976	11.60%
HTTP DoS	1,937	3.24%	831	3.24%
DDoS	26,112	43.61%	11,191	43.61%
SSH Brute Force	5,499	9.18%	2,357	9.18%
Total	59,877	100%	25,663	100%

We use the PyTorch 1.7.1 as the experimental framework on Ubuntu 18.04 64 bit OS with data parallelism across 8 GeForce RTX 2080 Ti GPUs. In experiments, we train our models using the mini-batch size of 16 and the learning rate of 0.01 in stochastic gradient descent.

A. DATA SETS

Our experiments use two distinct data sets for training and testing, respectively, where test data may include unseen attack types that are not contained in training data, so to perform a more realistic evaluation than the existing literature where training and testing are done using data from a single data set. In this work, we use the IDS-2012 data set for training and the IDS-2017 data set for testing, where the latter includes new attack types such as botnet and web attacks.

1) IDS-2012 DATA SET

The IDS-2012 data set [29] is a public data set published by the Information Security Centre of Excellence at the University of New Brunswick in 2012. This data set consists of seven days of network activities, including benign network traffic and four types of malicious network traffic: network infiltration, HTTP DoS, DDoS, and the SSH brute force attacks. The data has been collected by simulating normal users' behaviors and carrying out various multi-stage attack scenarios to generate realistic network traffic.

We have preprocessed the raw data of IDS-2012, composed of PCAP (packet capture) files and CSV files containing a 5-tuple (source IP address, source port, destination IP address, destination port, transport layer protocol), a timestamp, and a label for each flow. The preprocessing procedure can be summarized as follows. 1) Search for the packets with the same 5-tuple value of a single row from a CSV file. 2) Select the packets between the start time and the end time, where the start time corresponds to the timestamp value

TABLE 5. Statistics of the test set (IDS-2017).

Flow Type		Count	Percentage
Benign		2,103,377	82.93%
Brute Force	FTP Patator	4,579	0.18%
	SSH Patator	3,576	0.14%
DoS	Slowloris	5,764	0.23%
	Slowhttp	5,156	0.20%
	Hulk	157,696	6.22%
	GoldenEye	8,149	0.32%
Heartbleed		11	0.00%
Infiltration		35	0.00%
Web Attack	Brute Force	1,418	0.06%
	XSS	646	0.03%
	SQL Injection	12	0.00%
DDoS		85,217	3.36%
Port Scan		158,740	6.26%
BotNet		1,932	0.07%
Total		2,536,308	100%

in the row and the end time is the sum of start time and the flow duration value of the row. 3) Combine the selected packets and the label from the row to construct flow data. The resulting statistics of the IDS-2012 data set are shown in Table 3.

The preprocessing results of the IDS-2012 data set show that benign data constitutes 97.95 percent of the entire data set. Class imbalance can cause deep learning models to be biased towards the majority class and negatively impact classification performance. Therefore, we perform random subsampling to balance the numbers: the statistics of resulting training data based on IDS-2012 are shown in Table 4.

2) IDS-2017 DATA SET

The IDS-2017 data set [30] is designed to cover a variety of network attacks, including benign and malicious network traffic from realistic simulation of user behavior and execution of various attack scenarios. In particular, IDS-2017 is purposed to include more up-to-date network traffic and attacks than IDS-2012, providing traffic diversity and volumes for research. The raw data of IDS-2017 has been preprocessed in the same way as the case of IDS-2012 above. We have used the entire IDS-2017 data set for evaluation: the statistics are shown in Table 5.

3) FURTHER PREPROCESSING

In general, flow data has a variable length because the number of packets constituting the flow and the size of packets is different from each other. In this work, to deal with variable-length inputs, we transformed the flow data into the fixed-size data by fixing the number of packets per flow and each packet size to the specific values, where the values are determined by the average flow length and average packet length of the data sets. According to statistics, the average flow length of the IDS-2012 data set and the IDS-2017 data set are 10.05 packets and 10.84 packets, respectively. In addition, the average packet size of those data sets is 81.09 bytes and 167.09 bytes, respectively. We set the length of flows to 20 packets and the packet size to 800 bytes to minimize the input size while maintaining the packet payloads as much as possible. If a packet is longer than 800 bytes, we truncate the trailing bytes to fit it in 800 bytes. On the contrary, if a packet is shorter than 800 bytes, we pad it with zeroes. We also randomize the internet protocol (IP) address and media access control (MAC) address to remove the influence of IP address and MAC address on detecting malicious network traffic.

In summary, the preprocessing is composed of two main steps: 1) combine the packets matching the values of a row in a CSV file with a label value of the row to construct flow data, and then 2) zero-pad or truncate the variable-length flow data to fit it into fixed-size. Besides the above steps, random subsampling and randomization of network addresses are performed before and after the second step, respectively. In addition to the methods described above, there is no further data preprocessing such as removing redundant features.

B. EVALUATION METRICS

We evaluate the network intrusion detection (NID) models by means of the receiver operating characteristic (ROC) curve and the area under the curve (AUC). A ROC curve shows the true positive rate (TPR) against the false positive rate (FPR) at different classification threshold values. ROC analysis is critical since one can check 1) if a model can achieve the desired level of TPR, 2) the level of FPR at the desired point of TPR, 3) the AUC value, which tells the overall performance characteristic of the classifier under different threshold values.

The TPR and the FPR are defined as follows:

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \quad \text{FPR} = \frac{\text{FP}}{\text{TN} + \text{FP}}.$$

Here, true positive (TP) indicates the number of samples correctly classified as positive, and false positive (FP) means the number of samples incorrectly classified as positive. In addition, true negative (TN) is the number of samples correctly classified as negative, and FN indicates the number of samples incorrectly classified as negative.

Both TPR and FPR are essential indicators that represent the performance of NID systems [40]. High TPR means a high detection rate, which is the first priority in intrusion detection systems. At the same time, low FPR is also an essential factor in intrusion detection since high FPR implies that many of the detected attacks can be false alarms. Therefore if the detection system has a high FPR, it will be necessary for security professionals to examine all the alarms to distinguish true positives – this will be complicated and time-consuming, limiting our defense readiness to respond immediately to attacks. The accuracy rate used in many NID literature is a less appealing measure for evaluation since it represents the ratio of correct answers to all test cases, which can look very high if some of the major classes (e.g., benign) are classified correctly, even though there are many false positives and false negatives.

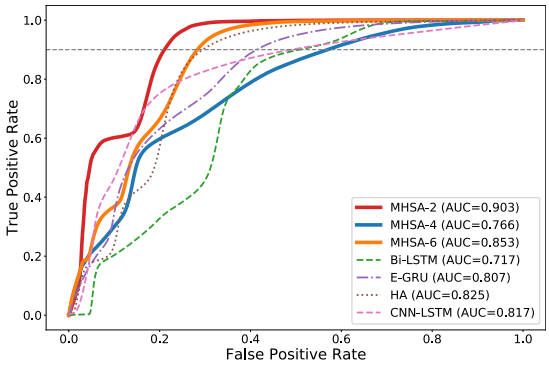


FIGURE 4. The ROC curves of the individual models. The horizontal dashed line indicates the TPR value of 0.9.

C. RESULTS

1) OVERALL DETECTION PERFORMANCE OF INDIVIDUAL MODELS

We compare the detection performance of our proposed MHSA model with that of the deep learning-based state-of-the-art malicious traffic detectors: bidirectional long short-term memory (denoted by Bi-LSTM), encoded gated recurrent unit (E-GRU) [39], hierarchical attention (HA) [17], and convolutional neural network long short-term memory (CNN-LSTM) [20]. Bi-LSTM is essentially a recurrent neural network (RNN) that distinguishes malicious and benign network traffic. E-GRU consists of an encoder and GRU networks. The encoder performs feature extraction on network flow data, whose outputs are fed into the GRU networks to classify attacks from network traffic. E-GRU shows improved detection capability through learning representations of network flows using the encoder from an auto-encoder. HA uses bidirectional GRU-based RNNs and attention connections to learn the hierarchical representation structure of network traffic. It progressively builds a flow embedding vector by aggregating the critical bytes into packet embedding vectors. HA integrates significant packets into the flow vector in a similar fashion, where the attention mechanism determines the importance of bytes and packets. Then, HA predicts whether the flow is benign or malicious with the flow vector. Finally, CNN-LSTM integrates CNN layers into an LSTM network to capture spatial and temporal patterns from network flows simultaneously. The resulting architecture is known to be fast compared to other state-of-the-art methods.

Figure 4 shows the ROC curves of the network intrusion detection models, where MHSA-2, MHSA-4, and MHSA-6 represent the variants of the proposed MHSA model with different numbers of MHSA blocks in the byte encoder. From the ROC curves, we can read how large FPR values will be from different methods to achieve the desired TPR level of 0.9. The best performing model is one of our proposed models, MHSA-2, whose FPR value is the smallest (0.210) among the models at the TPR value of 0.9. On the other hand, the FPR values of the deep learning-based state-of-the-art models: Bi-LSTM, E-GRU, HA, and CNN-LSTM are 0.508, 0.414, 0.295, and 0.490, respectively. As a result, MHSA-2 improves the detection performance compared to the state-of-the-art models by about 29% in FPR. In addition, MHSA-2 achieves performance enhancement by 9% in terms of the AUC score.

2) DETECTION PERFORMANCE OF INDIVIDUAL MODELS BY FLOW TYPES

Here we investigate the detection capability of classification models in terms of different flow types in the test data. We conjecture that each detection method will have the strength on different types of input patterns due to their differences in design. For instance, CNN-based models will better capture multiple localized patterns, while RNN-based models will better utilize (potentially long-term) temporal information. We also hypothesize that our proposed MHSA model will be better at detecting attacks such as DoS and DDoS since the multi-head self-attention mechanism will suit detecting scattered traces of attacks in multiple forms.

Table 6 shows the number of misclassified test instances for different flow types. The numbers in boldface indicate the best performance in each flow type. Here, we can observe that one of our proposed models, MHSA-2, achieves the best detection performance on DoS, DDoS, web attack, and port scan as we expected. Especially, the model shows remarkable performance on the DDoS and the port scan attacks with detection rates of 99.96% and 99.80%, respectively. MHSA-2 also has competent performance on other flow types, except for the benign and botnet types. Still, its performance for the benign and botnet attacks is not the worst among the competitors. Other models also achieve good performance on certain attack types, but they fail to identify other flow types well compared to those few. For example, while CNN-LSTM detects benign and port scan types well, the model has the worst performance on brute force, heartbleed, infiltration, and DDoS attacks.

3) COMPARISON WITH ENSEMBLE MODELS

In the previous experiment, we demonstrated the strength of our model MHSA-2 that it achieves the best overall prediction performance compared to the existing methods and is excellent at detecting DoS, DDoS, web attack, and port scan attacks that seem to be not adequately handled by the other methods. Here we compare the detection performance of our MHSA-2 model and ensemble models constructed by joining the state-of-the-art models, which is a typical approach to improve the performance of base classifiers.

We use the soft voting scheme for building ensemble models as discussed in Section III-C. Table 7 shows the prediction performance of all ensemble combinations of two state-of-the-art classifiers with their AUC scores on the test set, along with the scores of the single models for comparison. Here we can find that our MHSA-2 (the S1 model in the table) model surpasses the best ensemble model E6 in terms of the accuracy, F1 score, and AUC score. The MHSA-2 model has the AUC value of 0.903, which is better than that of the best ensemble model E6, 0.889. We also

TABLE 6. The number of misclassified instances of individual models and the best ensemble model to each attack type. The boldface numbers indicate the least number of misclassified instances for each attack type.

Model	Benign	Brute Force	DoS	Heartbleed	Infiltration	Web Attack	DDoS	Port Scan	BotNet
MHSA-2	405,394	2,960	**61,155**	2	15	**100**	**33**	**304**	1,390
MHSA-4	311,351	5,263	167,616	11	18	1,979	18,548	20,869	1,355
MHSA-6	308,140	1,807	160,436	11	32	1,573	5,440	20,323	1,716
Bi-LSTM	685,647	4,181	73,717	1	**0**	1,966	54,177	43,237	**335**
E-GRU	379,841	5,033	142,759	4	19	1,017	4,767	16,593	1,416
HA	315,571	**1,311**	160,559	**0**	10	450	83,273	3,132	1,538
CNN-LSTM	202,048	5,682	149,065	11	34	1,804	71,360	7,513	1,699
MHSA-2 + CNN-LSTM	**197,394**	4,135	147,224	11	34	1,535	45,661	3,278	1,752
Total no. of instances	2,103,377	8,155	176,765	11	35	2,076	85,217	158,740	1,932

TABLE 7. Comparison of the AUC scores, accuracy, and F1 scores between individual models and ensemble models, where the accuracy and F1 scores are computed at the TPR value of 0.9. The results show that our MHSA-2 outperforms the best ensemble model E6, consisting of MHSA-2 and CNN-LSTM. The boldface numbers indicate the best AUC scores, accuracy, F1 scores of individual and ensemble models.

	Model	MHSA-2	MHSA-4	MHSA-6	Bi-LSTM	E-GRU	HA	CNN-LSTM	AUC score	Accuracy	F1 score
Single	S1	•							**0.903**	**80.84%**	**0.616**
	S2		•						0.766	51.06%	0.386
	S3			•					0.853	74.84%	0.550
	S4				•				0.717	56.19%	0.412
	S5					•			0.807	63.97%	0.460
	S6						•		0.825	73.84%	0.540
	S7							•	0.817	57.64%	0.420
Ensemble	E1	•	•						0.866	74.49%	0.546
	E2	•		•					0.880	75.64%	0.558
	E3	•			•				0.845	67.43%	0.485
	E4	•				•			0.875	73.38%	0.536
	E5	•					•		0.865	78.27%	0.586
	E6	•						•	**0.889**	**78.56%**	**0.589**
	E7		•	•					0.830	68.23%	0.492
	E8		•		•				0.765	59.31%	0.430
	E9		•			•			0.806	59.35%	0.430
	E10		•				•		0.812	67.35%	0.485
	E11		•					•	0.800	57.65%	0.420
	E12			•	•				0.801	63.47%	0.457
	E13			•		•			0.841	66.90%	0.481
	E14			•			•		0.839	71.74%	0.521
	E15			•				•	0.858	73.22%	0.534
	E16				•	•			0.793	62.39%	0.450
	E17				•		•		0.777	60.86%	0.440
	E18				•			•	0.785	60.36%	0.437
	E19					•	•		0.826	67.61%	0.487
	E20					•		•	0.827	64.68%	0.465
	E21						•	•	0.839	74.20%	0.544

compared our MHSA-2 model to the ensembles joining three single models: still, our MHSA-2 model was the best (results are not shown for the sake of simplicity). Furthermore, Table 6 shows that the MHSA-2 model outperforms the best ensemble model E6 in detection rates for all attack types. We think these results come from the weaknesses of the state-of-the-art models in detecting particular attack types such as DoS, DDoS, port scan, and web attacks; the ensemble combinations of our MHSA-2 model and the existing models have weakened the strength of the best model MHSA-2.

The ROC plots of MHSA-2 and ensemble models are illustrated in Fig. 5. Here we present only the ROC curves of ensemble models based on MHSA-2 since the combinations of MHSA-2 and other models have achieved the best performance among all constructions. We also include the ROC curve of the state-of-the-art ensemble model [58] (we denote this model as the ET-ELM Ensemble) for comparison with our models. The ET-ELM Ensemble model is composed of extreme learning machine classifiers, each of which detects a specific type of attack using a different subset of features chosen by the extremely randomized trees-based feature selection method [61]. The ET-ELM Ensemble model combines the outputs of each classifier to identify the network intrusion. The figure shows that MHSA-2 model has achieved the best performance in ROC, which represents the overall detection performance in realistic evaluation. More specifically, our MHSA-2 model achieved the lowest FPR (0.210) at the TPR level of 0.9, which is a significant improvement of about 12% and 40% compared to the E6 model and ET-ELM Ensemble model (their FPR values are around 0.238 and 0.3516, respectively).

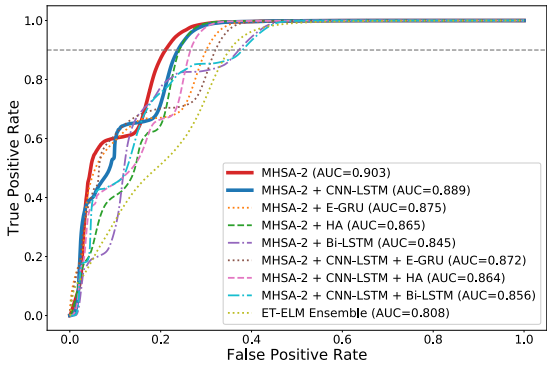


FIGURE 5. The ROC curves of MHSA-2 and ensemble models. The horizontal dashed line indicates the TPR value of 0.9.

V. CONCLUSION

In this paper, we present a new evaluation strategy to reflect better the real-world where cybersecurity threats are evolving continuously, and therefore unseen attacks are constantly emerging. We propose a novel neural detector based on multi-head self-attention, called MHSA, which can capture packet information from multiple perspectives to alleviate the limitations of the state-of-the-art models. The MHSA model improves the detection performance compared to the existing models, especially excellent at detecting DoS, DDoS, port scan, and web attacks. Moreover, the proposed MHSA model outperforms the best ensemble model constructed by combining the state-of-the-art models in detection rate for each attack type and consequently shows better overall detection performance. Although our MHSA model has achieved significant improvement over the existing methods, several aspects remain to be investigated for real-world intrusion detection system (IDS) development.

A. DETECTION OF UNSEEN ATTACKS

The purpose of applying AI technology in developing intrusion detection systems can be two-folded: reducing expert effort and detecting unseen attacks. Controlling false alarms through careful calibration by the ROC analysis of detectors will be essential to achieve the first goal. The second goal of detecting unseen attacks is why increasing numbers of developers incorporate AI technology into IDS development. However, the theoretical basis of machine learning assumes that the probability distribution of training and testing data be the same: detection of entirely new attacks will be very challenging, if not impossible. Therefore, the goal of detecting 'unseen' attacks should be elaborated as detecting new attacks that share specific characteristics to the known attacks used in training. For evaluation, one would need to find good training and test data that fit the purpose, such as IDS-2012 and IDS-2017 used in our experiments. However, more careful analysis will be needed to check if some public data sets fit the purpose of real-world deployment.

B. ON-SITE FINE-TUNING AND RETRAINING

In some IDS development scenarios, on-site fine-tuning and real-time retraining might be possible. However, both will require the collection of new site-specific data, where it is hard to determine how long we need to collect new data until we get a sufficient amount of information regarding unseen attacks. Further, we would need the intervention of human experts to detect and adequately label the attacks to use them for fine-tuning or retraining, where the speed of labeling can bottleneck the entire process. AI techniques such as transfer learning [62], [63] can be helpful to use new data from other sites if such data use is allowed by affected organizations. However, one would need to investigate the similarity of attacks among the sites to apply the techniques effectively. The retraining cost of the detection AI model will also be a critical factor.

C. BEST DETECTION MODELS FOR SPECIFIC ATTACK TYPES

In this paper, we have suggested the MHSA detector with a conjecture that its multi-head self-attention construction will help to capture scattered pieces of evidence in network attacks, and Table 6 shows that our detector MHSA-2 is the best at detecting DoS, DDoS, web attack, and port scan attack types. Among these, the detection performance improvement on DDoS attacks is very significant: about $\times 144$ compared to the second best, the E-GRU model. We believe this success indicates that there is still good room for improvement for detecting a specific type of attack, where the optimal models may use different architecture to better capture distinct characteristics of the attack: this will be a good direction for follow-up research.

REFERENCES

[1] S. T. Ikram and A. K. Cherukuri, "Improving accuracy of intrusion detection model using PCA and optimized SVM," *J. Comput. Inf. Technol.*, vol. 24, no. 2, pp. 133–148, Jun. 2016.
[2] R. Vijayanand, D. Devaraj, and B. Kannapiran, "Intrusion detection system for wireless mesh network using multiple support vector machine classifiers with genetic-algorithm-based feature selection," *Comput. Secur.*, vol. 77, pp. 304–314, Aug. 2018.
[3] Y. Tian, M. Mirzabagheri, S. M. H. Bamakan, H. Wang, and Q. Qu, "Ramp loss one-class support vector machine; a robust and effective approach to anomaly detection problems," *Neurocomputing*, vol. 310, pp. 223–235, Oct. 2018.
[4] K. A. P. Costa, L. A. M. Pereira, R. Y. M. Nakamura, C. R. Pereira, J. P. Papa, and A. X. Falcão, "A nature-inspired approach to speed up optimum-path forest clustering and its application to intrusion detection in computer networks," *Inf. Sci.*, vol. 294, pp. 95–108, Feb. 2015.
[5] H. Shapoorifard and P. Shamsinejad, "Intrusion detection using a novel hybrid method incorporating an improved KNN," *Int. J. Comput. Appl.*, vol. 173, no. 1, pp. 5–9, Sep. 2017.
[6] B. Ingre, A. Yadav, and A. K. Soni, "Decision tree based intrusion detection system for NSL-KDD dataset," in *Proc. Int. Conf. Inf. and Commun. Technol. Intell. Syst.*, 2017, pp. 207–218.
[7] A. Vladutu, D. Comanecii, and C. Dobre, "Internet traffic classification based on flows' statistical properties with machine learning," *Int. J. Netw. Manage.*, vol. 27, no. 3, pp. 19–27, May 2017.
[8] B. Li, S. Zhang, and K. Li, "Towards a multi-layers anomaly detection framework for analyzing network traffic," *Concurrency Comput., Pract. Exper.*, vol. 29, no. 14, p. e3955, Jul. 2017.
[9] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 1, pp. 41–50, Feb. 2018.
[10] F. Farahnakian and J. Heikkonen, "A deep auto-encoder based approach for intrusion detection system," in *Proc. 20th Int. Conf. Adv. Commun. Technol. (ICACT)*, Feb. 2018, pp. 178–183.

[11] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," in *Proc. IEEE Int. Conf. Intell. Secur. Informat. (ISI)*, Jul. 2017, pp. 43–48.
[12] H. Zhou, Y. Wang, X. Lei, and Y. Liu, "A method of improved CNN traffic classification," in *Proc. 13th Int. Conf. Comput. Intell. Secur. (CIS)*, Dec. 2017, pp. 177–181.
[13] M. Yeo, Y. Koo, Y. Yoon, T. Hwang, J. Ryu, J. Song, and C. Park, "Flow-based malware detection using convolutional neural network," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Jan. 2018, pp. 910–913.
[14] B. J. Radford, L. M. Apolonio, A. J. Trias, and J. A. Simpson, "Network traffic anomaly detection using recurrent neural networks," 2018, *arXiv:1803.10769*. [Online]. Available: http://arxiv.org/abs/1803.10769
[15] C. Xu, J. Shen, X. Du, and F. Zhang, "An intrusion detection system using a deep neural network with gated recurrent units," *IEEE Access*, vol. 6, pp. 48697–48707, 2018.
[16] B. Roy and H. Cheung, "A deep learning approach for intrusion detection in Internet of Things using bi-directional long short-term memory recurrent neural network," in *Proc. 28th Int. Telecommun. Netw. Appl. Conf. (ITNAC)*, Nov. 2018, pp. 1–6.
[17] L. Han, Y. Sheng, and X. Zeng, "A packet-length-adjustable attention model based on bytes embedding using flow-wgan for smart cybersecurity," *IEEE Access*, vol. 7, pp. 82913–82926, 2019.
[18] C. Liu, Y. Liu, Y. Yan, and J. Wang, "An intrusion detection model with hierarchical attention mechanism," *IEEE Access*, vol. 8, pp. 67542–67554, 2020.
[19] W. Wang, Y. Sheng, J. Wang, X. Zeng, X. Ye, Y. Huang, and M. Zhu, "HAST-IDS: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection," *IEEE Access*, vol. 6, pp. 1792–1806, 2018.
[20] Y. Zhang, X. Chen, L. Jin, X. Wang, and D. Guo, "Network intrusion detection: Based on deep hierarchical network and original flow data," *IEEE Access*, vol. 7, pp. 37004–37016, 2019.
[21] T. Su, H. Sun, J. Zhu, S. Wang, and Y. Li, "BAT: Deep learning methods on network intrusion detection using NSL-KDD dataset," *IEEE Access*, vol. 8, pp. 29575–29585, 2020.
[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Neural. Inf. Process. Syst. (NeurIPS)*, 2017, pp. 6000–6010.
[23] M. Al-Fawa'reh, A. Saif, M. T. Jafar, and A. Elhassan, "Malware detection by eating a whole APK," in *Proc. 15th Int. Conf. for Internet Technol. Secured Trans. (ICITST)*, Dec. 2020, pp. 268–276.
[24] Y. Chen, S. Wang, D. She, and S. Jana, "On training robust PDF malware classifiers," in *Proc. 29th USENIX Secur. Symp.*, 2020, pp. 2343–2360.
[25] Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le, "Self-training with noisy Student improves ImageNet classification," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 10687–10698.
[26] R.-H. Hwang, M.-C. Peng, V.-L. Nguyen, and Y.-L. Chang, "An LSTM-based deep learning approach for classifying malicious traffic at the packet level," *Appl. Sci.*, vol. 9, no. 16, p. 3414, Aug. 2019.
[27] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, 2016, pp. 1480–1489.
[28] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–15.
[29] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Comput. Secur.*, vol. 31, no. 3, pp. 357–374, May 2012.
[30] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. 4th Int. Conf. Inf. Syst. Secur. Privacy*, 2018, pp. 108–116.
[31] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symp. Comput. Intell. Secur. Defense Appl.*, Jul. 2009, pp. 1–6.
[32] J. Song, H. Takakura, and Y. Okabe. *Description of Kyoto University Benchmark Data* Accessed: Jan. 10, 2021. [Online]. Available: https://www.takakura.com/Kyoto_data/BenchmarkData-Description-New.pdf
[33] M. Garuba, C. Liu, and D. Fraites, "Intrusion techniques: Comparative study of network intrusion detection systems," in *Proc. 5th Int. Conf. Inf. Technol., New Generat. (ITNG)*, Apr. 2008, pp. 592–598.
[34] B. I. Santoso, M. R. S. Idrus, and I. P. Gunawan, "Designing network intrusion and detection system using signature-based method for protecting OpenStack private cloud," in *Proc. 6th Int. Annu. Eng. Seminar (InAES)*, Aug. 2016, pp. 61–66.
[35] P. Garcia-Teodoro, J. Diaz-Verdejo, G. Macia-Fernandez, and E. Vazquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *Comput. Secur.*, vol. 28, nos. 1–2, pp. 18–28, 2009.
[36] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.
[37] H. T. Kam, "Random decision forests," in *Proc. 3rd Int. Conf. Document Anal. Recognit.*, vol. 1, 1995, pp. 278–282.
[38] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *Amer. Statistician*, vol. 46, no. 3, pp. 175–185, Aug. 1992.
[39] Y. Hao, Y. Sheng, and J. Wang, "Variant gated recurrent units with encoders to preprocess packets for payload-aware intrusion detection," *IEEE Access*, vol. 7, pp. 49985–49998, 2019.
[40] G. C. Tjhai, M. Papadaki, S. M. Furnell, and N. L. Clarke, "Investigating the problem of IDS false alarms: An experimental study using Snort," in *Proc. 23rd Int. Inf. Secur. Conf. (IFIP TC)*, 2008, pp. 253–267.
[41] N. Farnaaz and M. A. Jabbar, "Random forest modeling for network intrusion detection system," *Procedia Comput. Sci.*, vol. 89, pp. 213–217, May 2016.
[42] P. A. A. Resende and A. C. Drummond, "A survey of random forest based methods for intrusion detection systems," *ACM Comput. Surveys*, vol. 51, no. 3, pp. 1–36, Jul. 2018.
[43] Y. Y. Aung and M. M. Min, "An analysis of random forest algorithm based network intrusion detection system," in *Proc. 18th IEEE/ACIS Int. Conf. Softw. Eng., Artif. Intell., Netw. Parallel/Distrib. Comput. (SNPD)*, Jun. 2017, pp. 127–132.
[44] Y. Chang, W. Li, and Z. Yang, "Network intrusion detection based on random forest and support vector machine," in *Proc. 7th IEEE Int. Conf. Comput. Sci. Eng. (CSE), IEEE Int. Conf. Embedded Ubiquitous Comput. (EUC)*, Jul. 2017, pp. 635–638.
[45] H. Benaddi, K. Ibrahimi, and A. Benslimane, "Improving the intrusion detection system for NSL-KDD dataset based on PCA-fuzzy clustering-KNN," in *Proc. 6th Int. Conf. Wireless Netw. Mobile Commun. (WINCOM)*, Oct. 2018, pp. 1–6.
[46] Y. Xiao, C. Xing, T. Zhang, and Z. Zhao, "An intrusion detection model based on feature reduction and convolutional neural networks," *IEEE Access*, vol. 7, pp. 42210–42219, 2019.
[47] H. Yang and F. Wang, "Wireless network intrusion detection based on improved convolutional neural network," *IEEE Access*, vol. 7, pp. 64366–64374, 2019.
[48] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware traffic classification using convolutional neural network for representation learning," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, 2017, pp. 712–717.
[49] N. Muraleedharan and B. Janet, "Behaviour analysis of HTTP based slow denial of service attack," in *Proc. Int. Conf. Wireless Commun., Signal Process. Netw. (WiSPNET)*, Mar. 2017, pp. 1851–1856.
[50] *SplitCap. (3.0.0.0). NETRESEC*. Accessed: Jan. 14, 2021. [Online]. Available: https://www.netresec.com/?page=SplitCap
[51] L. Rokach, "Ensemble-based classifiers," *Artif. Intell. Rev.*, vol. 33, nos. 1–2, pp. 1–39, 2010.
[52] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. NAACL-HLT*, 2019, pp. 4171–4186.
[53] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7132–7141.
[54] A. Moore, D. Zuev, and M. Crogan, "Discriminators for use in flow-based classification," Queen Mary Univ. London, London, U.K., Tech. Rep. RR-05-13, 2005.
[55] N. Moustafa, B. Turnbull, and K.-K.-R. Choo, "An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4815–4830, Jun. 2019.
[56] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. Mil. Commun. Inf. Syst. Conf. (MilCIS)*, Nov. 2015, pp. 1–6.
[57] F. Haddadi and A. N. Zincir-Heywood, "Benchmarking the effect of flow exporters and protocol filters on botnet traffic classification," *IEEE Syst. J.*, vol. 10, no. 4, pp. 1390–1401, Dec. 2016.

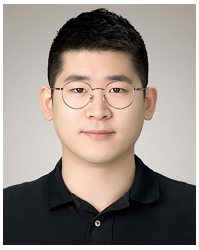
[58] J. Sharma, C. Giri, O.-C. Granmo, and M. Goodwin, "Multi-layer intrusion detection system with ExtraTrees feature selection, extreme learning machine ensemble, and softmax aggregation," *EURASIP J. Inf. Secur.*, vol. 2019, no. 1, pp. 1–16, Dec. 2019.
[59] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," in *Proc. Eur. Conf. Comput. Learn. Theory*, 1995, pp. 23–27.
[60] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: A new learning scheme of feedforward neural networks," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, vol. 2, Jul. 2004, pp. 985–990.
[61] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Mach. Learn.*, vol. 63, no. 1, pp. 3–42, 2006.
[62] S. Bozinovski and A. Fulgosi, "The influence of pattern similarity and transfer learning upon the training of a base perceptron B2. (Original in Croatian)," in *Proc. Symp. Inf.*, May 1976, pp. 3–121.
[63] L. Y. Pratt, "Discriminability-based transfer between neural networks," in *Advances in Neural Information Processing Systems 5*. San Mateo, CA, USA: Morgan Kaufmann, 1993, pp. 204–211.

SEONGYUN SEO received the B.S. degree in cyber defense from Korea University, Seoul, South Korea, in 2018, where he is currently pursuing the M.S. degree with the School of Cybersecurity.

His research interests include secure AI, such as adversarial robustness and backdoor robustness, as well as AI for security, including malware detection and network intrusion detection.

SUNGMIN HAN received the B.S. degree in computer engineering from Kwangwoon University, Seoul, South Korea, in 2020, where he is currently pursuing the M.S. degree with the School of Cybersecurity.

His main research interests include signal processing, generative models, anomaly detection, and adversarial robustness.



JANGHYEON PARK received the B.S. degree from Korea University, Seoul, South Korea, in 2016, where he is currently pursuing the M.S. degree with the School of Cybersecurity.

He is working on cybersecurity as an Officer in the military, and is doing research on AI security at the AI Research Laboratory, Korea University Graduate School. His research interests include the study of model stealing attacks to protect the copyright and privacy of AI model.

SHINWOO SHIM received the B.S. degree in computer science and engineering from POSTECH, in 2007, and the M.S. degree in information security from Korea University, in 2019.

He has been working with LIG NEX1 Corporation, Ltd., South Korea, since 2007. His research interests include command and control in cyber warfare, mission impact analysis, and course of action in cyber warfare.

HAN-EUL RYU received the B.S. and M.S. degrees in computer engineering from Korea Aerospace University, in 2009 and 2011, respectively.

He has been working with LIG NEX1 Corporation, Ltd., South Korea, since 2011. His research interests include cybersecurity, cyber training systems, and system/network virtualization.

BYOUNGMO CHO received the M.S. degree in computer science and engineering from Inha University, South Korea.

He is currently a Research Engineer of cybersecurity with LIGNex1 Corporation, Ltd. His research interests include machine learning and threat hunting.

SANGKYUN LEE received the B.S. and M.S. degrees in computer science from Seoul National University, Seoul, South Korea, in 2003 and 2005, respectively, and the M.S. and Ph.D. degrees in computer science from the University of Wisconsin-Madison, Madison, WI, USA, in 2008 and 2011, respectively.

From 2011 to 2014, he was a Postdoctoral Research Fellow with the Collaborative Research Center SFB876 within TU Dortmund University, Germany, and from 2015 to 2016, he served as the Project Leader of the Collaborative Research Center SFB876, leading the C1 Division. From 2017 to 2019, he was an Assistant Professor with the Division of Computer Science, College of Computing, Hanyang University ERICA Campus, Ansan-si, South Korea. Since 2020, he has been affiliated as an Assistant Professor with the School of Cybersecurity, Korea University, Seoul. His research interests include machine learning, including AI for security, secure AI, model compression, computer vision, time-series prediction, and distributed learning.