Lightweight Convolutional Neural Network Based Intrusion Detection System

Vinh Pham, Eunil Seo, and Tai-Myoung Chung
Sungkyunkwan University, Suwon and 16419, South Korea
Email: vinhpham@g.skku.edu; seoei2@skku.edu; tmchung@skku.edu

Abstract —Identifying threats contained within encrypted network traffic poses a great challenge to Intrusion Detection Systems (IDS). Because traditional approaches like deep packet inspection could not operate on encrypted network traffic, machine learning-based IDS is a promising solution. However, machine learning-based IDS requires enormous amounts of statistical data based on network traffic flow as input data and also demands high computing power for processing, but is slow in detecting intrusions. We propose a lightweight IDS that transforms raw network traffic into representation images. We begin by inspecting the characteristics of malicious network traffic of the CSE-CIC-IDS2018 dataset. We then adapt methods for effectively representing those characteristics into image data. A Convolutional Neural Network (CNN) based detection model is used to identify malicious traffic underlying within image data. To demonstrate the feasibility of the proposed lightweight IDS, we conduct three simulations on two datasets that contain encrypted traffic with current network attack scenarios. The experiment results show that our proposed IDS is capable of achieving 95% accuracy with a reasonable detection time while requiring relatively small size training data.

Index Terms—Intrusion detection, machine learning, convolutional neural network

I. Introduction

The rapid development of technology not only makes life more comfortable but also reveals many security problems. The growing number of various cyber threats endanger today's Internet. Hackers are inventing new techniques daily to bypass security layers and avoid detection. Therefore, IDS play an indispensable role in defending against intrusions and malicious activities. IDS can be classified into Network Intrusion Detection Systems (NIDS) and Host-based Intrusion Detection Systems (Host-IDS). NIDS centrally monitors and analyzes network traffic, while Host-IDS monitor specific system in the network [1].

The continuous development of new cyber attacks necessitates the rapid extension of new communication protocols that not only encrypt user payload data but also hide the original information in the packet header (e.g., dynamic port). Hence, traditional approaches such as deep packet inspection and signatures could not be applied to encrypted traffic [2]. To cope with this challenge, IDS uses the machine learning algorithm to catch up on the problematic situation.

Machine learning algorithms require a large amount of training data to detect threats with high accuracy. These algorithms typically prefer statistical data as their training and input data. An IDS based on machine learning (e.g., [3]-[6]) would require sufficiently powerful hardware to process such big size statistical data. In practice, collecting statistic data from network traffic takes some time and might result in delayed intrusion detection. These limitations have made previous machine learning-based IDS impractical and unrealistic for deployment in real-world situations.

In this paper, we propose a lightweight CNN-based IDS, which only requires a small amount of data but can still detect threats with high accuracy. It should also be noted that related works have had to cope with imbalanced datasets in building machine learning IDS (e.g., [6], [7]). The lightweight CNN-based IDS described here will help us avoid the problem of an imbalanced dataset as we can easily construct a balanced dataset with minimal training data obtained. The proposed lightweight CNN-based IDS uses raw network traffic as the input data. This feature of our proposal is realistic, as the first part of an attack that can be observed is its raw traffic. With this practical feature, the proposed IDS has quicker detection time, facilitates real-time detection in a feasible manner, and is easily implemented and deployed in practice. We employ a method described in [8] for transforming raw traffic into an image. We also update this method to an enhanced method that works better on malicious traffic. In summary, we make the following contributions:

1) We propose a lightweight CNN-based IDS, which can be easily implemented and deployed into both real-time NIDS and Host-IDS, and that requires a comparatively small dataset while achieving a reasonable accuracy (e.g., 95%) and detection time.
2) We evaluate several performance metrics (such as accuracy, detection time, and training data size) of the proposed CNN-based IDS, which are

Manuscript received April 25, 2019; revised October 13, 2020.
This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2020-0-00990 Platform Development and Proof of High Trust & Low Latency Processing for Heterogeneous • Atypical • Large Scaled Data in 5G-IoT Environment) and by Healthcare AI Convergence Research & Development Program through the National IT Industry Promotion Agency of Korea (NIPA) funded by the Ministry of Science and ICT (No. S1601-20-1041)
Corresponding author email: tmchung@skku.edu.
doi:10.12720/jcm.15.11.808-817

trained by two CIC-IDS2018 and CIC-VPN2016 datasets in up-to-date network attack scenarios (e.g., Botnet, BruteForce, DoS, etc.).

3) The performance of the two methods transforming raw traffic into image are compared to determine which method is preferred and will be suggested for implementation in our proposal.

The paper is organized as follows. Section 2 provides a detailed introduction of our primary references and a short survey of the related research works, while Section 3 described the two datasets. The paper will go on to illustrate our proposed system, components, and functions in Section 4. Section 5 presents our experimental simulations and interprets our obtained results in detail. Finally, Section 6 summarizes the paper and provides our vision for future work.

II. Related Work

Representation learning [9] is a new machine learning approach that involves automatically learning features from raw data without the process of hand-designing features. Wang *et al.* [10] proposed combining a machine learning-based classifier with representation learning to identify the type of malware by the raw traffic generated. They transformed the payload of a data packet to image data, then trained image data on a one-dimension CNN model; they achieved an average accuracy of 99.41%. With this success in malware traffic classification, they applied the same method in an encrypted application traffic classification task [11] and achieved up to 99.9% accuracy. Our proposed IDS employs their representation learning approach to accept raw network traffic as the input data.

Shapira *et al.* [8] stated that a packet payload will change according to different encryption methods and encryption keys; and that packet payload should therefore not be employed to represent the characteristic of encrypted network traffic, like Wang *et al.* did in [10], [11]. Instead, they proposed the Payload Size Distribution (PSD) Histogram that utilizes time-related and size-related features to represent encrypted raw network traffic into the representation form of an image. This work in [8] also demonstrated that the PSD Histogram is "immune to encryption techniques". This means that the machine learning model can effectively classify sets of Internet traffic encrypted by different encryption techniques if they are transformed into image data through the PSD Histogram method. We employ the PSD Histogram method and attempt to classify the kinds of attack software that generated the malicious traffic that was sent to the victim computer in a network.

Helmer *et al.* [12] proposed a Multi-agent IDS that combined multiple lightweight agents running on individual computers to secure a distributed system. Zhengbing *et al.* [13] proposed an intelligent lightweight IDS that employed a data mining and watermark technique to effectively and efficiently detect intrusions in real-time. Zaman *et al.* [14] might be considered one of the pioneers who applied the features selection concept to reduce the amount of data needed, thus improving IDS scalability and extendibility. There have also been various other essential studies [15], [16] that attempted to deploy machine learning-based lightweight IDS on IoT devices with limited resources.

Table I presents the previous machine learning-based IDS that were tested on the CSE-CIC-IDS2018 dataset to evaluate their performance.

TABLE I: Previous Machine Learning IDS.

Name	Accuracy	Data size	Remark
Kim *et al.* [17]	99% DoS	1040548	13% Infiltration
Lin *et al.* [6]	96.2% average	2172045	17.12% Infiltration
Roy *et al.* [7]	88.5% average	164536	40% Infiltration
Zhou *et al.* [18]	96% average	whole	57% Malicious only
Catillo *et al.* [5]	99.2% Botnet	whole	95.8% BruteForce

Because there are substantial differences in data interpretation, data preparation and strategies were applied to the dataset in the research highlighted above, and we could not acquire consistent numbers of training data between them. Some studies, such as [5], [18], did not mention a specific number but referred to using the entire dataset. However, the common limitation between these proposed IDS is that the training data size typically exceeded one million units, but not all the attack scenarios were detectable; i.e., the accuracies for Infiltration were below 50% [6], [7], [17].

III. Dataset

In this work, we employed two widely used datasets to evaluate our proposed IDS: CSE-CIC-IDS2018 (IDS2018) [19] and ISCX-VPN2016 (VPN2016) [20].

A. CSE-CIC-IDS2018

The IDS2018 was released by the Canadian Institute of Cybersecurity (CIC) for intrusion detection research in 2018. This dataset includes network traffics captured from an infrastructure that includes fifty hosts, and the victim organization has five departments with four hundred twenty hosts and thirty servers with different operating systems (e.g., Windows, Ubuntu, and MACOS X) and network devices (e.g., modems, switches, firewall, and routers) used. This dataset contains the most popular and up-to-date network attack scenarios which are Brute-force, Botnet, DoS, DDoS, Web attacks, and Infiltration, resulting in a total of 14 types of intrusions (e.g., FTP-BruteForce, XSS, and DoS-Slowloris).

The IDS2018 is provided in two data formats: The first data format is the raw record of network traffic (PCAP) and the event logs of all computers. The second data format (CSV) is fully labeled and describes the 83 statistical features of the traffic (e.g., flow duration, number of packets, average size of the data packet, standard deviation time between two data packets sent in

the backward direction, etc.). Most related research has preferred the labeled format because it is widely believed that statistics based on network flow can provide more useful information for intrusion detection [21]. Instead, we employ the raw network traffic (i.e., PCAP) as a target object for our IDS to detect intrusions in this paper.

B. ISCX-VPN2016

CIC also released the VPN2016 to generate a representative dataset of real-world traffic that is sufficiently rich in both diversity and quantity. This dataset is provided in two formats, like IDS2018, and contains seven categories of application network traffic (e.g., Browsing, Email, Chat, Streaming, File Transfer, VoIP, and P2P). A VPN encryption method was used to encrypt all traffic at the protocol layer (TLS, IPsec, etc.). This dataset has been popular in many studies researching topics related to the classification of application traffic [8], [11].

Classifying application traffic involves identifying the type (e.g., Streaming, File Transfer, VoIP, etc.) of a certain set of network traffic based on their behaviors. Intrusion detection, which involves analyzing the characteristic of the generating tool, proceeds under the same principle as application traffic classification. These attack tools generate some kinds of malicious encrypted traffic that will be sent to the network. Each piece of malicious traffic is expected to carry the characteristic of the generating tool within it. We assume that if our proposed lightweight IDS can classify the correct type of encrypted traffic belonging to the VPN2016 dataset, then it should also be able to detect encrypted malicious traffic.

IV. Design of the Proposed Lightweight CNN-Based IDS

In this section, we will discuss the architecture of the lightweight CNN-based IDS. The proposed IDS consists of two components: a data transformation process and a CNN-based detection model. We also propose a multiple classifiers combination method for deployment. A schematic diagram of the proposed lightweight IDS is shown in Fig. 1.

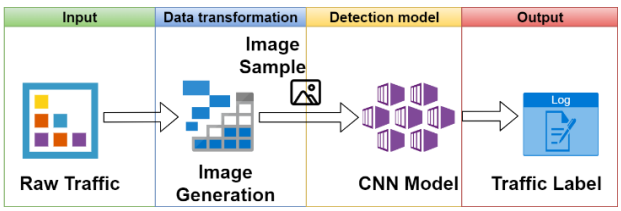

Fig. 1. Lightweight CNN-based IDS.

A. Data Transformation

The primary function of the data transformation is to represent the raw network traffic's characteristics as an image that can be analyzed by a CNN model to determine whether it represents normal traffic or malicious traffic generated by attack tools.

According to [14], a lightweight IDS is a small, flexible, and highly capable system that accomplishes essential tasks using minimal data, and that is dynamically updatable and upgradable. We refer to a portion of monitored network traffic within a given time window as a block. This block is the data unit that will be analyzed by the detection model and that will also be used for training data. If we employ statistical data based on network traffic as the input data for IDS, it would require some data preparation (e.g., feature selection, cleaning the data, outlier removal, etc.) steps. These extra workloads would take time, thus resulting in a delay in making observations in the network traffic (which involves constructing a block in our case) that would be analyzed thereafter, hence interfering with the real-time detection of IDS. By contrast, by transforming raw network traffic into small image data, we can minimize the workload of both constructing the training data and quickly making blocks.

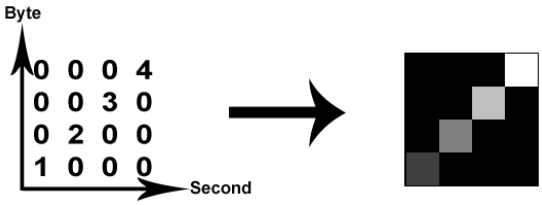

Fig. 2. Data transformation illustration.

A simple example might be the best way to explain the data transformation process. Let us say we have monitored ten packets that were sent to our computer. There was one 1-byte packet that arrived at the 1st second, two 2-byte packets at the 2nd second, three 3-byte packets at the 3rd second, and finally four 4-byte packets at the 4th second. We can then initialize a two-dimension array data structure that represents a 2D image, where the Y-axis stands for the packet length and the X-axis stands for the arrival time of the packet.

The value of a cell that has its index represents the length, and the arrival time of the packet will be set equal to the number of the respective packet. To clarify, the [0, 0] cell will be 1, the [1, 1] cell will be 2, and so on. Then, that array data structure can be saved into a 2D image (e.g., PNG format) with the value of each cell representing the brightness of the respective pixel. One image is one block. That means the brighter the pixel, the more packets of equivalent length and arrival time there are within that block. Fig. 2 shows a simple illustration of this process. This explanation is the principle of the PSD Histogram method, which was introduced in [8] for the application traffic classification task.

By inspecting the IDS2018 dataset, we realize that in most attack scenarios, the malicious traffic contains TCP control packets (i.e., a packet with flags such as SYN, ACK, PSH, etc.) which have small length values that are always below 100 bytes. In addition, there is a notable aspect of malicious traffic generated by attack tools, which we call a sequence pattern of packet length. For

example, in the malicious raw network traffic of the Botnet attack scenario, two [ACK] 54-byte packets and one [SYN, ACK] 66-byte packet always came after one [PSH, ACK] 71-byte packet, resulting in a 71-54-54-66 sequence pattern that was repeated many times. This behavior might be worth exploiting to identify malicious traffic.

Therefore, we modify the original PSD method to exploit this new finding by replacing the Y-axis of the representation image with a new metric that we term Sequence Pattern (SP). This new metric is calculated as follows:

- First, multiply the packet length by its normalized arrival time. This multiplication helps us discriminate between two packets that have the same length but different arrival times, then distribute them into different portions of the representation image.
- Secondly, modulo the result of the above step to the predefined size of the representation image. The purpose of this second step is to calculate a valid value for the SP metric because it must be smaller than the size of the image.

The pseudocode for calculating the index of a specific cell belonging to the two-dimension array data structure that represents its respective pixels is shown in Algorithm 1. Whenever the IDS observes an incoming packet within a time window, it will calculate the index of the representative cell for that packet length and arrival time, then increase that cell by one. When the time window is exceeded, the two-dimension array is ready to be saved as image data, then reset for the next time window.

Algorithm 1: Cell index calculation for PSD & SP.

```
Result: xPSD, yPSD, xSP, ySP
t: arrival time of the packet;
l: length of the packet;
PNGSIZE: Image size;
INTERVAL: Time window;
MAXLEN: Maximum length of the packet;
base: first packet appears within a time window;
i: i-th packet after the first one;
xPSD = (t[i]-t[base]) * PNGSIZE / INTERVAL;
yPSD = l[i] * PNGSIZE / MAXLEN;
xSP = xPSD;
ySP = modulo(l[i]*xSP, PNGSIZE);
```

(a) Botnet PSD (b) Botnet SP

Fig. 3. Pattern visual comparison of PSD & SP image data

Fig. 3 shows the pattern visual comparison between two image data generated from Botnet malicious traffic by PSD and our new proposed SP method. By replacing the packet length with the SP metric for the Y-axis, the SP image data seems to be disordered. However, the disorder is distinct between different kinds of malicious network traffic, which may imply latent patterns that can be identified by the detection model, resulting in higher accuracy detection. By contrast, PSD image data share a common regular pattern consisting of horizontal dashed or dotted lines, so they are easily confused by the detection model, resulting in lower accuracy detection.

B. CNN-based Detection Model

Because we applied the representation learning approach by transforming raw traffic into image data, the intrusion detection problem is changed to an image classification problem. Classifying small-sized images will require less computing and data processing power, while the training time and time needed to classify new representation images will still be relatively short. Returning to our intrusion detection problem, if we employ a simple but effective machine learning model for image classification, then the training time and detection time will be substantially enhanced compared to employing a complex machine learning model for statistical data.

CNN [22] has been proven effective and highly accurate in image classification tasks. We use CNN as the core detection model for the proposed IDS. Moreover, the openness in the CNN architecture design unlocks many fine-tuning possibilities aiming for different targets, such as classifying time, accuracy, and acceleration.

C. Multiple Classifiers Combination

Hackers continually try to invent new attack methods to bypass the latest security solutions. Alongside malicious traffic, there are also many kinds of normal application traffic. The number of classification classes must increase to adapt to the increasing number of new attack methods as well as applications. As a result, the architecture of a detection model must be evolved and more complex to cope with newly appearing classes. The more complex the architecture, the longer the training and detection times. There is no single perfect solution for all problems; in other words, there is no perfect classifier for all classes.

In this section, we propose a deployment concept that combines two classifiers, which are:

- One PSD classifier (e.g., [8]) that classifies the raw network traffic first. If that traffic belongs to the well-known category of application (e.g., Chrome web browsing, Facebook video streaming, etc.), they will pass through the network.
- By contrast, if one piece of traffic is classified as unknown, it is considered potentially malicious traffic. This unknown traffic should be redirected to the proposed lightweight IDS, which is an SP classifier. The SP classifier will then detect whether this unknown traffic belongs to an attack tool or not.

An illustration of the deployment concept is shown in Fig. 4. Finally, the update scenario of our CNN-based IDS should be carefully considered. Suppose we deploy one NIDS at the server and other Host-IDS at particular

computers to secure our network. The training process of a CNN model might require some processing capability for long-term usage when the data volume increases while operating day by day. Then, it might not be a good choice for each Host-IDS to train its CNN-based detection model to update itself to cope with newly-appearing malicious traffic.

The main NIDS should be responsible for training the CNN model because the server may be the most powerful computer in the network. Then, the updated CNN model will be distributed to other Host-IDS. This update scenario should save time and improve the overall performance of all IDS within the network while keeping up with new attack methods.

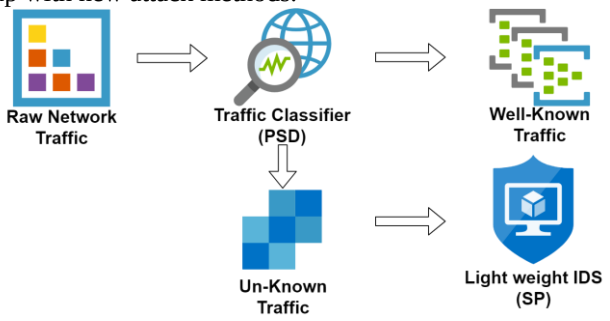

Fig. 4. Multiple classifiers combination.

V. Experiment & Result

The whole process of our simulation can be summarized as follows:

1) We filtered all TCP packets that were sent to the victim computer that we selected for the deployment of the IDS. We employed the WireShark application to read the PCAP files provided by the dataset, filter only two features of each packet (arrival time and packet length), and store the whole list in CSV format files.
2) We group all the packets within a pre-defined time window based on the arrival time of each packet, as a group of packets belongs to one block. Then, we generate two image data, one by the PSD method and the other by the SP method. Both of the image data represent the corresponding block.
3) Finally, we use the generated image data for training and testing a CNN model to obtain the results.

We employed the Python version of the Pillow (PIL) library to generate the image data. We built and ran the CNN model with Keras API [23] on top of the machine learning platform TensorFlow [24]. The training was done by optimizing the *categorical cross entropy* [25] cost function. The *Adam* [26] optimizer was used for the optimization process. In all simulations, the CNN model reached convergence after around 30 epochs.

A. Experimental Setup

As stated in [27], the IDS2018 dataset is heavily imbalanced; the traffic of the DDoS+PortScan attack scenario dominated the traffic from other attack scenarios. There was also a huge gap to each of the other classes in terms of size. If we employ a completely intact dataset, it will significantly decrease the performance, as the CNN-based detection model tends to classify many samples into the dominance class.

To minimize the size of the training data set and maintain balance between the classification classes, we try generating the image data for all attack scenarios of the IDS2018 dataset. Then, we select only image data belonging to classes that are not much different in size. In the end, we include image data from five out of the six attack scenarios, which are Bruteforce (FTP and SSH), DoS (GoldenEye and Slowloris), Web attack (Bruteforce-Web and Bruteforce-XSS), Botnet, and Infiltration. The image dataset consists of eight classes from the IDS2018 dataset.

With the VPN2016 dataset, we select the traffic of four types of applications that are Chat (Facebook, Hangout), File transfer (FTPS, SFTP), Video streaming (Netflix, YouTube), and VoIP (Hangout, Skype). We employ these criteria for selection in an attempt to reproduce the work demonstrated in [8], to which we will compare our result. The image dataset also consists of eight classes from the VPN2016 dataset.

The name of each class and its number of blocks are displayed in Tables II and III for the two image datasets.

By the analysis of the IDS2018 dataset, we set up the time window of ten seconds, and the size of the image data was 30x30 pixels for the simulation on the IDS2018 dataset. These two parameters were decided based on the average number of packets and their packet length within a time window. If there are more packets with variety in the packet length within a time window, then we need to employ a larger size of the image to represent the characteristics of the traffic. However, large-sized image data requires more processing power and may stop the system from being lightweight. If the time window is too short, then there are too few packets to be able to represent the traffic's characteristics. However, a long time window but a small size image is useless, as there are few pixels to capture the characteristics of the traffic. The optimal parameters depend on the concrete situation and the hardware on which we deploy the IDS.

Table II: Labels and Number Blocks of Each Class in IDS2018.

Attack name	Label	Number
FTP-BruteForce	0	565
Infiltration	1	341
Botnet	2	490
SSH-Bruteforce	3	535
DoS-GoldenEye	4	92
DoS-Slowloris	5	237
BruteForce-Web	6	385
BruteForce-XSS	7	404
Total	8 classes	3049 blocks

TABLE III: LABELS AND NUMBER BLOCKS OF EACH CLASS IN VPN2016.

Application	Label	Number
Chat-Facebook	0	244
Chat-Hangout	1	270
File-FTPS	2	87
File-SFTP	3	78
Video-Netflix	4	260
Video-Youtube	5	234
VoIP-Hangout	6	250
VoIP-Skype	7	250
Total	8 classes	1673 blocks

For VPN2016, normal applications tend to generate traffic containing packets with various lengths. For example, video streaming applications typically generate packets over 1000 bytes in size, while the packet length of a chat application is variable depending on the content of the message. The number of packets within a time window is also relatively large, so we have to define the time window as six seconds and the image size as 300x300 pixels to effectively capture the characteristics of the traffic of these applications.

As mentioned above, there are many types of fine-tuning we can apply on the CNN architecture for different purposes (e.g., maximizing accuracy, minimizing the detection time or the required processing power, etc.). In these simulations, we only use a sparse CNN architecture, as shown in Table IV, so we can evaluate the performance in the most general situation.

In some studies testing the IDS on the IDS2018 dataset, including [6], [7], [17], they usually include normal (benign) traffic as one classification class. In the multiple classifier combination concept, we have included the PSD application traffic classifier to filter out normal traffic. Those pieces of traffic that would be re-directed to IDS are considered potential highly malicious traffic. Therefore, we did not include normal traffic as a classification class in our simulation.

TABLE IV: THE ARCHITECTURE OF THE CNN DETECTION MODEL

Layer (type)	Output Shape	Param #
Conv2D	(None, 30, 30, 64)	640
MaxPooling2D	(None, 15, 15, 64)	0
Conv2D	(None, 13, 13, 128)	73856
MaxPooling2D	(None, 6, 6, 128)	0
Conv2D	(None, 4, 4, 256)	295168
MaxPooling2D	(None, 2, 2, 256)	0
Conv2D	(None, 1, 1, 256)	262400
Flatten	(None, 256)	0
Dense	(None, 256)	65792
Dense	(None, 7)	1799
Total params:		699655

B. Evaluation Metrics

To evaluate the proposed model, we employ three metrics: precision, recall, and f1-score. The f1-score is the weighted average of precision and recall, so the f1-score is preferable when we need to balance between precision and recall. In this paper, we will refer to the f1-score for accuracy rather than the original accuracy metric.

Although we did not apply the k-Fold Cross-Validation for the testing process, we obtain the testing result with a similar workflow. After generating the image data, we randomly split them up into a training set, validation set, and test set with a respective ratio of 60%/20%/20%. We repeat this workflow five times. In addition, because the training process may be affected by random seeds which are automatically set up by the TensorFlow platform, in each repeated time we train and test the CNN model five times. This setting might be equivalent to a 5-fold Cross-Validation procedure for the best results.

The average result of testing times is typically used to evaluate the performance of the proposed IDS. However, we employ only a minimal training data to train the detection model. Over time, our IDS performance will be improved because it will be trained with updated training data. We choose to use the best result between testing times for the evaluation because it is more suitable for evaluating the potential of our proposed IDS.

C. IDS Overall Performance (IDS2018)

We generated the representation image data using both the PSD and SP methods, trained and tested according to the pre-described workflow, then compared the results between the two methods. The best results (precision, recall, and f1-score) of each class for each method and their summaries are listed in Table V. The Support column indicates how many blocks of each class were reserved for testing (20% of the total generated blocks).

The f1-scores of both the PSD and SP methods are around 90% and up to 95%. In our opinion, 90% may be a reasonable threshold for an IDS. With a minimal dataset (1831 blocks for training and 609 blocks for validation), we were able to meet this threshold. With more data, we may be able to improve the detection accuracy further. The results suggest that the combination of representation learning and machine learning is a potential approach for building future IDS.

TABLE V: IDS2018 SIMULATION RESULT

Label	**precision**		**recall**		**f1-score**		**Support**
	PSD	SP	PSD	SP	PSD	SP	
0	1.00	0.99	1.00	1.00	1.00	1.00	113
1	0.93	0.88	0.56	0.75	0.70	0.81	68
2	0.95	0.88	0.90	0.92	0.92	0.90	98
3	1.00	1.00	1.00	1.00	1.00	1.00	107
4	1.00	0.90	0.94	1.00	0.97	0.95	18
5	0.98	0.96	1.00	0.91	0.99	0.93	47
6	0.78	0.91	0.74	0.96	0.76	0.94	77
7	0.62	0.99	0.90	1.00	0.74	0.99	81
accuracy					0.89	**0.95**	609
macro avg	0.91	0.94	0.88	0.94	0.88	0.94	609
weighted avg	0.90	0.95	0.89	0.95	0.89	0.95	609

We can also observe that the three classes that the lower accuracy of PSD can be most attributed to are classes 1, 6, and 7 (with respective f1-scores of 70%, 76%, and 74%). Compared to the SP method's accuracy (81%, 94%, and 99%, respectively), we can see that the SP method achieved some improvements over the PSD method of around 5% in terms of overall f1-score.

Class	0	1	2	3	4	5	6	7
0	113	0	0	0	0	0	0	0
1	0	38	5	0	0	0	2	**23**
2	0	0	88	0	0	0	8	2
3	0	0	0	107	0	0	0	0
4	0	0	0	0	17	1	0	0
5	0	0	0	0	0	47	0	0
6	0	1	0	0	0	0	57	**19**
7	0	2	0	0	0	0	6	73

Fig. 5. PSD confusion matrix.

Examining the Confusion Matrix (Fig. 5), we can observe that 23 blocks from class 1 and 19 blocks from class 6 were misclassified as class 7. This is why the f1-score of the PSD method for these three classes is lower than that of the SP method. Because the image data generated by the PSD method belonging to these three classes are quite similar to each other, they are easily misclassified, resulting in lower overall accuracy of the PSD method.

We examine misclassified blocks belonging to three classes above that achieved low accuracy in IDS2018 simulation (Infiltration, BruteForceWeb, and BruteForce-XSS) generated by the PSD method versus their corresponding blocks generated by the SP method (Fig. 6). Visually, we can observe that the PSD blocks are identical to each other. On the other hand, in SP blocks, bright pixels are distributed in different portions of the image. The similarity of PSD blocks might cause trouble for the CNN model in classification, resulting in underperformance compared to the SP blocks.

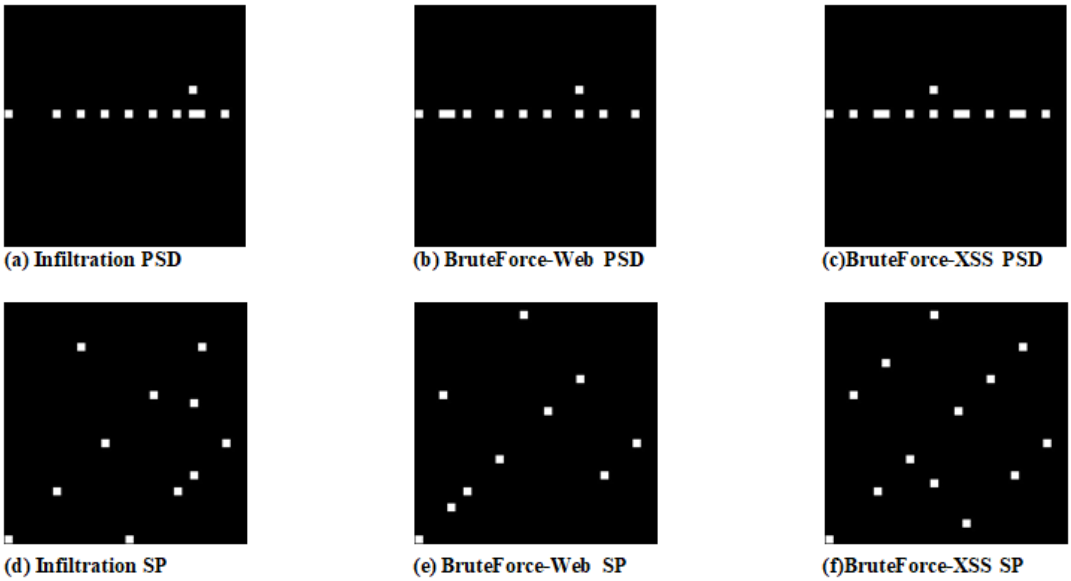

Fig. 6. PSD & SP Visual Comparison (For illustration purposes, white pixels represent any value between 1 and 255, in real image data these pixels are visually different in brightness).

D. IDS Performance on Encrypted Traffic (VPN2016)

In this simulation, we aim to obtain the performance of the proposed IDS, specifically on encrypted network traffic. There are two reasons for us to conduct the simulation on the VPN2016 dataset:

First, we want to check whether or not our simulation workflow is correct and robust by trying to reproduce the results of [8] that claimed that the PSD method could achieve accuracy up to 99% in encrypted traffic classification on the same dataset. If we can reproduce the equivalent result, this would guarantee the correctness of our simulation, and our obtained results (on both IDS2018 and VPN2016) could be considered trustworthy for evaluation purposes. Second, as in the previous simulation, we also want to determine which method (PSD versus SP) is better for filtering normal application traffic.

TABLE VI: VPN2016 BI-DIRECTION SIMULATION RESULT

Label	precision		recall		f1-score		Support
	PSD	SP	PSD	SP	PSD	SP	
0	0.89	0.74	0.96	0.82	0.92	0.78	49
1	0.88	0.72	0.93	0.61	0.90	0.66	54
2	0.58	0.32	0.65	0.41	0.61	0.36	17
3	0.86	0.36	0.38	0.25	0.52	0.30	16
4	0.94	0.80	0.94	0.92	0.94	0.86	52
5	0.79	0.81	0.79	0.72	0.79	0.76	47
6	1.00	0.98	1.00	1.00	1.00	0.99	50
7	1.00	1.00	1.00	0.98	1.00	0.99	50
accuracy					**0.90**	0.79	335
macro avg	0.87	0.72	0.83	0.71	0.84	0.71	335
weighted avg	0.90	0.79	0.90	0.79	0.89	0.79	335

In the first simulation in this section, we filter both TCP packets that were sent to and sent from the deployed IDS computer. This approach was employed in the

experiment in [8] for their obtained result. We named this simulation as the bi-direction simulation because it contains packets coming from both the in and out directions of the computer. The result of the first simulation is presented in Table VI. With the PSD method, we can achieve accuracy of around 90% compared to the claimed result of the original work [8], which is 99%.

The 10% difference might come from the difference in the dataset, as [8] stated that they also captured their own traffic dataset and merged it into VPN2016 to produce a combined dataset in an attempt to obtain the best result. Moreover, they monitored the traffic with a longer time window and generated image data with a bigger size, which are 60 seconds and 1500x1500 pixels, respectively, while in our case they are 6 seconds and 300x300 pixels, respectively. Because of this, we consider the 10% difference to be acceptable and can therefore guarantee that our simulation is correct and robust.

In addition, we observe that the SP method achieves only 79% accuracy in terms of f1-score. The PSD method is more suitable for the application traffic classification than the SP method. The PSD method should be employed to filter normal traffic out of malicious traffic, as suggested in the multiple classifier combination concept. However, is the PSD method more suitable for detecting encrypted malicious traffic than the SP method? This is the question explored in the second simulation in this section.

TABLE VII: VPN2016 UNI-DIRECTION SIMULATION RESULT

Label	precision		recall		f1-score		Support
	PSD	SP	PSD	SP	PSD	SP	
0	0.00	0.65	0.00	0.79	0.00	0.71	47
1	0.00	0.69	0.00	0.66	0.00	0.67	53
2	0.50	0.50	0.50	0.14	0.50	0.22	14
3	0.50	0.44	0.20	0.47	0.29	0.45	15
4	1.00	0.74	0.98	0.96	0.99	0.84	51
5	0.91	0.91	0.81	0.57	0.86	0.70	37
6	0.24	1.00	1.00	1.00	0.39	1.00	50
7	0.00	1.00	0.00	1.00	0.00	1.00	50
accuracy					0.44	**0.79**	317
macro avg	0.39	0.74	0.44	0.70	0.38	0.70	317
weighted avg	0.35	0.80	0.44	0.79	0.36	0.78	317

In contrast to the first simulation, in the second simulation, we filter only TCP packets that were sent to the deployed IDS computer. This approach supposes that the IDS should be able to detect the malicious traffic of an attack quickly enough when that traffic enters the IDS computer without needing to wait for the reply traffic from the victim computer. This approach is the same with the simulation on IDS2018. Because of the data transformation implementation, there is a small change in the numbers of generated image data between the first (335 blocks for test set) and second (317 blocks for test set) simulations. However, this difference in the number of blocks is not significant, so it will not affect the result. We named the second simulation as the uni-direction simulation.

The result of the second simulation is presented in Table VII. It can clearly be seen that in the extreme condition which requires quick detection of only incoming traffic, the PSD method accuracy was dropped to 44%. By contrast, the SP method could still maintain its result from the first simulation, with 79% accuracy. A potential reason for this behavior of the PSD method is the shortage of packets that lead to the ineffective representation of the traffic's characteristics that can be discriminated between classes. The second simulation result draws a conclusion indicating that the SP method is more suitable for representing encrypted attack traffic as image data. Thus, SP should be applied for the data transformation in proposed IDS.

E. Detection Time

We can estimate the required time to detect an attack based on the sum of the data transformation time and the time needed to classify the image data. The data transformation time might depend on many aspects of the concrete implementation. While conducting simulations, we also measure the needed time for completing the testing on test sets (Fig. 7) to estimate the time for classifying one image data (Fig. 8).

Simply speaking, the image size is the factor that most affects the classification time. It costs around 3.42 seconds to complete classification of 317 300x300 image data, but only 1.97 seconds to complete classification of 609 30x30 image data. Based on this result, we can estimate that the needed time to classify a block will be around 3.23 microseconds (30x30) or 10.78 microseconds (300x300). This indicates that the detection time was minimized to the micro-second level, so our remaining challenge is implementing data transformation in such a way that minimizes the needed time to transform the raw traffic into image data.

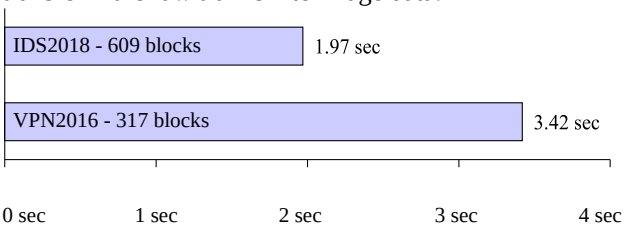

Fig. 7. Testing time.

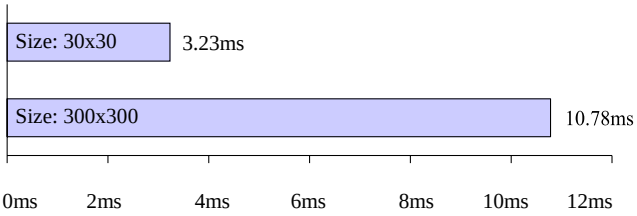

Fig. 8. Classification time on one block.

As a reference, one study [16] proposed a lightweight IDS implementation for a Raspberry Pi 3 (a single-board computer) that could classify 47693 instances in 8.75 seconds. This means it takes around 0.18 microseconds to classify one instance. However, this research employed a

machine learning algorithm which requires statistical data as input data, so there might have been other factors that needed to be considered, e.g., required time for collecting statistical data, data preparation, running feature selection algorithm, etc.

Moreover, that study [16] aimed to deploy IDS on IoT devices with limited resources, so they integrated some optimizations, e.g., sophisticated lightweight machine learning models and algorithms for feature selection. By contrast, our proposed IDS is aimed toward computer-based devices, and we implement our simulations in general ways and using general tools. Hence, in our opinion, our lightweight IDS detection time is relatively acceptable.

F. Training Data Size

As mentioned previously, a lightweight IDS should be able to accomplish its essential tasks with minimal amounts of data. Compared to Table I, our proposed IDS was trained with only 2440 images, partitioned into 1831 images in the training set and 609 images in the validation set, but it was able to achieve an acceptable accuracy (95%), and all attack scenarios were detectable (i.e., f1-scores over 90%) except for Infiltration (f1-score of 81%). Further, we employed only some portions of the IDS2018 dataset. We believe that the performance of our lightweight IDS will be improved over time as there will be more training data available.

VI. Conclusion

In this paper, we show how to implement a lightweight machine learning IDS with representation learning and how to leverage the CNN model to classify image data. Furthermore, the proposed CNN-based IDS shows highly accurate threat detection from the encrypted malicious raw network traffic. We propose methods that transform raw network traffic into representation images using only two features of raw traffic: packet length and arrival time. A CNN model was employed in constructing the IDS detection model to minimize the required training data and detection time. Three simulations were conducted on two popular datasets for evaluation. The experimental results demonstrate that the classification is capable of achieving accuracies of 95% in identifying malicious network traffic and of 79% in classifying encrypted traffic. We specifically note that the achieved accuracy (95%) only required a small amount of training data (2440 images), and the detection time was minimized to 3.23 microseconds, which is competitive relative to those of previously reported machine learning-based IDS. In future work, we plan to implement our proposed IDS and evaluate it with new attack scenarios in a real-world environment.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Vinh Pham conducted this research including formulating idea, performance evaluation to the final manuscript under the guidance of Eunil Seo. Tai Myoung-Chung is the corresponding author. All authors had approved the final version.

References

[1] D. Aksu, S. Üstebay, M. A. Aydin, and T. Atmaca, "Intrusion detection with comparative analysis of supervised learning techniques and fisher score feature selection algorithm," in *Proc. International Symposium on Computer and Information Sciences*, 2018, pp. 141–149.

[2] B. Anderson and D. McGrew, "Identifying encrypted malware traffic with contextual flow data," in *Proc. ACM Workshop on Artificial Intelligence and Security*, pp. 35–46, 2016.

[3] D. Pérez, S. Alonso, A. Morán, M. A. Prada, J. J. Fuertes, and M. Domínguez, "Comparison of network intrusion detection performance using feature representation," in *Proc. International Conference on Engineering Applications of Neural Networks*, 2019, pp. 463–475.

[4] O. Faker and E. Dogdu, "Intrusion detection using big data and deep learning techniques," in *Proc. ACM Southeast Conference*, 2019, pp. 86–93.

[5] M. Catillo, M. Rak, and U. Villano, "2l-zed-ids: A two-level anomaly detector for multiple attack classes," in *Proc. Workshops of the International Conference on Advanced Information Networking and Applications*, 2020, pp. 687–696.

[6] P. Lin, K. Ye, and C. Z. Xu, "Dynamic network anomaly detection system by using deep learning techniques," in *Proc. International Conference on Cloud Computing*, 2019, pp. 161–176.

[7] D. D. Roy and D. Shin, "Network intrusion detection in smart grids for imbalanced attack types using machine learning models," in *Proc. International Conference on Information and Communication Technology Convergence (ICTC)*, 2019.

[8] T. Shapira and Y. Shavitt, "Flowpic: Encrypted internet traffic classification is as easy as image recognition," in *Proc. IEEE INFOCOM 2019-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2019, pp. 680–687.

[9] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[10] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware traffic classification using convolutional neural network for representation learning," in *Proc. International Conference on Information Networking (ICOIN)*, 2017, pp. 712–717.

[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*

International Conference on Intelligence and Security Informatics (ISI), 2017, pp. 43–48.

[12] G. Helmer, J. S. Wong, V. Honavar, L. Miller, and Y. Wang, "Lightweight agents for intrusion detection," *Journal of Systems and Software*, vol. 67, no. 2, pp. 109–122, 2003.

[13] H. Zhengbing, S. Jun, and V. Shirochin, "An intelligent lightweight intrusion detection system with forensics technique," in *Proc. 4th IEEE Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications*, 2007, pp. 647–651.

[14] S. Zaman and F. Karray, "Lightweight ids based on features selection and ids classification scheme," in *Proc. International Conference on Computational Science and Engineering*, 2009, vol. 3, pp. 365–370.

[15] N. U. Sheikh, H. Rahman, S. Vikram, and H. AlQahtani, "A lightweight signaturebased ids for iot environment," *arXiv preprint arXiv:1811.04582*, 2018.

[16] Y. N. Soe, Y. Feng, P. I. Santosa, R. Hartanto, and K. Sakurai, "Implementing lightweight iot-ids on raspberry pi using correlation-based feature selection and its performance evaluation," in *Proc. International Conference on Advanced Information Networking and Applications*, 2019, pp. 458–469.

[17] J. Kim, Y. Shin, E. Choi, *et al.*, "An intrusion detection model based on a convolutional neural network," *Journal of Multimedia Information System*, vol. 6, no. 4, pp. 165–172, 2019.

[18] Q. Zhou and D. Pezaros, "Evaluation of machine learning classifiers for zeroday intrusion detection–an analysis on cic-aws-2018 dataset," *arXiv preprint arXiv:1905.03685*, 2019.

[19] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *ICISSP*, pp. 108–116, 2018.

[20] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of encrypted and vpn traffic using time-related," in *Proc. 2nd International Conference on Information Systems Security and Privacy (ICISSP)*, 2016, pp. 407–414.

[21] Z. Zhang, J. Li, C. Manikopoulos, J. Jorgenson, and J. Ucles, "Hide: a hierarchical network intrusion detection system using statistical preprocessing and neural network classification," in *Proc. IEEE Workshop on Information Assurance and Security*, 2001, pp. 85–90.

[22] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[23] F. C. *et al.*, "Keras," 2015.

[24] M. A. *et al.*, "Tensorflow," 2015.

[25] R. A. Dunne and N. A. Campbell, "On the pairing of the softmax activation and cross-entropy penalty functions and the derivation of the softmax activation function," in *Proc. 8th Aust. Conf. on the Neural Networks, Melbourne*, 1997, vol. 181, p. 185.

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

[27] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

Vinh Pham received the B.S. degree in Computer Science from the Ho Chi Minh City University of Technology - VNU-HCM (HCMUT), Vietnam in 2018. He is currently pursuing the M.S. degree with the Department of Computer Science and Engineering, Sungkyunkwan University, Korea. His research interests include network security, data mining and machine learning.

Dr. Eunil Seo received a B.S. degree from the Sungkyunkwan University, Korea, in 1997, and an M.S. degree from the University of Southern California (USC), Los Angeles, USA 2002. Over 16 years, he carried out the 17 patents, drafts, and technical specifications regarding the connectivity of devices for Samsung Advanced Institute of Technology (SAIT) and International Thermonuclear Experimental Reactor (ITER). He received a Ph.D. degree in Computer Engineering from Sungkyunkwan University, Korea, in 2019. He is currently a post-doctoral researcher at the computing science of Umea University, Sweden. His research interests are data connectivity using machine learning, Traffic Engineering (T.E.), LISP-based SDWN, and SDN&NFV-based Mobility Management (MM).

Dr. Tai-Myoung Chung received his first B.S. degree in Electrical Engineering from Yonsei University, Korea in 1981 and his second B.S. degree in Computer Science from University of Illinois, Chicago, USA in 1984. He received his M.S. degree in Computer Engineering from the University of Illinois in 1987 and his Ph.D. degree in Computer Engineering from Purdue University, W. Lafayette, USA in 1995. He is currently a professor of Information and Communications Engineering at Sungkyunkwan University, Korea. He is now a vice-chair of the Working Party on IS & Privacy, OECD, and a senior member of IEEE. He also serves as a Presidential committee member of the Korean e-government, the chair of the information resource management committee of the e-government. He is an expert member of Presidential Advisory Committee on Science & Technology of Korea, and is chair of the Consortium of Computer Emergency Response Teams (CERTs). His research interests are in information security, network, information management, and protocols on the next-generation networks such as active networks, grid networks, and mobile networks.