IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 15, 2021, accepted January 4, 2022, date of publication January 20, 2022, date of current version January 28, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3145002

Robust Network Intrusion Detection System Based on Machine-Learning With Early Classification

TAEHOON KIM AND WOOGUIL PAK, (Member, IEEE)
Department of Information and Communication Engineering, Yeungnam University, Gyeongsan 38541, South Korea

Corresponding author: Wooguil Pak (wooguilpak@yu.ac.kr)

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea Government [Ministry of Science and Information and Communications Technology (ICT)] under Grant NRF-2019R1F1A1062320.

ABSTRACT Network Intrusion Detection Systems (NIDSs) using pattern matching have a fatal weakness in that they cannot detect new attacks because they only learn existing patterns and use them to detect those attacks. To solve this problem, a machine learning-based NIDS (ML-NIDS) that detects anomalies through ML algorithms by analyzing behaviors of protocols. However, the ML-NIDS learns the characteristics of attack traffic based on training data, so it, too, is inevitably vulnerable to attacks that have not been learned, just like pattern-matching machine learning. Therefore, in this study, by analyzing the characteristics of learning using representative features, we show that network intrusion outside the scope of the learned data in the feature space can bypass the ML-NIDS. To prevent this, designing the active session to be classified early, before it goes outside the detection range of the training dataset of the ML-NIDS, can effectively prevent bypassing the ML-NIDS. Various experiments confirmed that the proposed method can detect intrusion sessions early (before sessions terminate) significantly improving the robustness of the existing ML-NIDS. The proposed approach can provide more robust and more accurate classification with the same classification datasets compared to existing approaches, so we expect it will be used as one of feasible solutions to overcome weakness and limitation of existing ML-NIDSs.

INDEX TERMS Network intrusion detection, early classification, robust classification, adversarial attack, machine-learning.

I. INTRODUCTION

It is very important to detect a network intrusion quickly and accurately for stable operation of the network. For this purpose, a dedicated security device called the Network Intrusion Detection System (NIDS) was proposed [1], [2]. The initial NIDS generated patterns from existing attacks and detected intrusions very quickly and accurately through pattern matching with the received packets [3]–[5]. However, a method that relies on patterns for existing attacks has a disadvantage in that it is impossible to detect an attack that is not known beforehand, and the network is easily penetrated by a variant of an existing attack.

To solve this problem, various methods have been proposed and applied to the NIDS [6], [7]. The machine learning-based NIDS (ML-NIDS), which has recently received the most attention, was evaluated as an alternative that can significantly improve the shortcomings of the pattern matching NIDS (PM-NIDS). The ML-NIDS analyzes the characteristics of existing network intrusions using ML and detects the intrusions using overall behavioral characteristics. Therefore, while the PM-NIDS can be penetrated by simply changing the pattern of an intrusion session, the ML-NIDS can detect intrusions even if some characteristics are changed as long as the overall behavior is maintained. Therefore, it is well known through the results of several studies that the ML-NIDS provides higher robustness for intrusion detection than the PM-NIDS.

However, the ML-NIDS learns the overall behavior of an intrusion using a training dataset, so just like the PM-NIDS, it strongly depends on having the pattern of the existing attack, and its detection ability depends on the training

The associate editor coordinating the review of this manuscript and approving it for publication was Hayder Al-Hraishawi.

dataset. In other words, like the PM-NIDS, the ML-NIDS can have a very low probability of detecting an intrusion that does not exist in the training data. Nevertheless, research on such limitations is not being conducted much. Instead, various methods of avoiding the ML-NIDS by modifying features in the feature space are in progress, and recently, studies to supplement the robustness of training datasets through a Generative adversarial network (GAN) and other deep learning have been proposed [21]–[23]. However, these studies do not directly analyze the dependence of the ML-NIDS on the learning dataset, so there are limitations in understanding the characteristics of that dependence.

In this paper, we directly analyze these characteristics and propose a method to provide robustness to the ML-NIDS training dataset without increasing its size. The proposed method analyzes the characteristics of the training dataset for the ML-NIDS and uses discovered characteristics to significantly improve intrusion detection performance without major changes in the system. To that end, the method proposed in this paper increases the detection range of the training dataset by analyzing the existing session-based dataset.

The main contributions of this study are as follows.

First, it is proven that ML-NIDS is vulnerable for detecting existing intrusion with some behavioral modification by adding some packets. Through analyzing ML-NIDS datasets, it is found that dependence on the training dataset is very high, so weaknesses similar to the PM-NIDS exist. In particular, it shows that the influence of the dependency can be quite different based on the ML algorithm.

Second, to alleviate strong dependency on the training dataset in terms of packet count, we present a method for selecting when the ML-NIDS optimally detects an intrusion. Through this, even too short or too long sessions that cannot be detected by the existing ML-NIDS can be detected with very high accuracy. In particular, compared to the existing PM-NIDS, early attack detection is possible on a similar hardware platform, so it is advantageous in keeping the network safe.

Third, since the proposed method is so light to be implemented on the existing ML-NIDS platform instead of a high-cost, high-performance hardware platform, the proposed approach is feasible in economic terms.

II. PREVIOUS WORK

The types of ML-NIDS are packet-based methods that use packet data directly as features, and session-based methods that use statistical data for a logical group called a session instead of packets as features. The packet-based method can be classified in two ways: one detection method uses a single packet to detect a pattern for malicious data in every packet received, and the other detection method uses multiple packets, storing and combining packets belonging to the same session into one dataset that is used for detection [2]–[5], [9], [35], [36]. Both the single-packet detection method and the multi-packet detection method search for malicious code or patterns in the packet payloads [10]. Owing to the high accuracy of the pattern-matching algorithm, it can detect malicious traffic while maintaining a very low false positive rate (FPR). However, attacks exploiting normal packets, like a Distributed denial-of-service (DDoS), are hard to detect with the packet-based method, and the pattern-matching algorithm can easily be bypassed by adding random data to the payload. Therefore, only the pattern-based method is not used alone.

In order to solve the shortcomings of the packet-based method, studies have been proposed to extract session features and detect an intrusion through them [13]–[18]. When using session features, it is impossible to bypass the NIDS just by adding some dummy data. In addition, regardless of the packet size or the length of the session, the size of the entire feature is always the same, so the session-based method is more advantageous than the packet-based method for handling large volumes of traffic.

The NIDS using session features mostly uses machine learning algorithms to classify the received traffic. So far, various ML-NIDSs have been developed and are expected to overcome the weaknesses of the PM-NIDS. Inevitably, malicious users are developing various methods to bypass the ML-NIDS (largely divided into white box, gray box, and black box methods), depending on what information can be used. The white box method is a way to bypass the NIDS when the attacker knows all information about it [19]–[23]. This is an ideal and unrealistic case, because information about the dataset, the machine learning model, and the feature set used for learning has to be available. On the other hand, the gray box method is where a malicious user knows minimal information, such as the algorithm for extracting features [24], [25]. The black box method, however, finds a way to bypass the NIDS without having any information in advance [26]. Therefore, although it is the most realistic, it is technically quite difficult to implement, because it is necessary to actively collect the necessary information and indirectly identify the characteristics of the NIDS.

Although various approaches and many related studies exist, we can see that the accuracy of a commonly learned classification model is greatly affected by a small number of features [19], [25]. Therefore, in the white box method, the corresponding features can easily be found, and the NIDS can easily be bypassed by generating an attack that exceeds the learning range of these features. Of course, research to alleviate these weaknesses is also being conducted. Various methods, such as removing some of the most influential features and training a classification model, have been proposed. However, the method of removing some features is not a fundamental solution in that it can inevitably affect the performance of the ML model. In the end, it is urgent to make a robust ML model by lowering the dependency on, and sensitivity to, features that affect the learning model while maintaining classification accuracy at the same time [26], [27].

Fundamentally, it is unreasonable to assume that the NIDS equipped with a model created using pre-built training data can learn all the characteristics of all sessions received from an actual network in advance. In fact, the training dataset

size is limited, so it is possible for a session to exist where the values for some specific features exceed the range of the learning values from the training dataset. If the corresponding feature is one that greatly affects the performance of the learning model described above, the sessions cannot be accurately classified by ML-NIDS. Therefore, this is a problem that must be solved in order to develop a system to defend against attacks. Nevertheless, research on this is not being conducted. Table 1 summarizes the pros and cons of each type of ML-NIDS, including the proposed approach.

TABLE 1. Strengths and weaknesses of each type of ML-NIDS.

Type	Strength	Weakness
Packet-based	- Fast detection. - Low false positive rate for existing intrusion	Vulnerable to attacks exploiting normal packets or adding random data to the payload.
Session-based	- High detection rate for attacks based on normal packets or randomly increased payload. - High scalability in terms of traffic rate and volume	Vulnerable to adversarial attack targeting some specific features
Proposed	Robustness against adversarial attack to any specific features.	Real-time monitoring overhead for each session.

III. THE PROPOSED APPROACH

We present a new method to improve the ML-NIDS in order to handle intrusion sessions with feature values that exceed the range of learned values. Therefore, an ML model combined with our proposed approach can detect intrusions that exceed the classification range of the training dataset in the feature space with high probability, so it is expected to not only make the ML-NIDS robust, but will also help prevent existing adversarial attacks.

A. MOTIVATION

Since the training dataset determines the performance of the ML-NIDS, it is most important to implement a training dataset including a large amount of rich data on network intrusions without redundancy as much as possible. However, since the size of the training dataset is finite, the area of the feature space learned with the training dataset is inevitably limited. In order to confirm this in more detail, it is necessary to analyze the effect when the learning range of the training dataset and the range of the test dataset do not overlap in the feature space. For further explanation, let us define some notations as follow:

A session S is defined by $S = \{P_1, P_2, \ldots, P_k\}$, where it consists of k packets. Let us define $\mathrm{src}(P_i)$, $\mathrm{size}(P_i)$, $\mathrm{rtime}(P_i)$ by source IP of P_i, size of P_i, reception time of P_i, respectively. Then the forward packet count and the total data rate are defined by $|S_{\mathrm{forward}}|$ and $\frac{\sum_{i=1}^{k} \mathrm{size}(P_i)}{\mathrm{rtime}(P_k) - \mathrm{rtime}(P_1)}$, where $S_{\mathrm{forward}} = \{P_h \mid P_h \in S, \mathrm{src}(P_h) = \mathrm{src}(P_1)\}$.

In previous studies, the forward packet count and the total data rate are known to be very important features in the ML-NIDS [19], [25]. According to the value of the forward packet count feature, in this experiment, a training dataset consisting of sessions with values smaller than a threshold value and a test dataset consisting of sessions with larger than a threshold value were created, and an experiment was conducted to measure classification performance using them. Since the influence of the forward packet count feature may be different for each class, the following experiment was conducted to analyze it. With the data for the i-th class among the entire dataset, only sessions with a forward packet count value less than or equal to a threshold value (the maximum forward packet count value for class i, θ_i) were selected to create the training dataset, and only sessions where the forward packet count value was greater than θ_i were used to construct the test dataset. Here, θ_i sets the data configuration ratio at 7:3 for the corresponding class. For other classes, the training and test datasets were randomly configured, regardless of the θ_i value.

θ_i is set to a value close to the ratio of 7:3 because sufficient training and test dataset size are required to obtain accurate classification performance. That is, if θ_i is too large, the test dataset becomes too small to accurately measure classification performance. On the contrary, if θ_i is too small, the training dataset becomes too small compared to train the ML model, causing degraded classification performance. Figure 1 shows f1-scores of some selected classes according to the dataset ratio. From the figure, we can see that the ratio should not be too small or too large according to the class. Thus, each θ_i was set as close to 7:3 as possible.

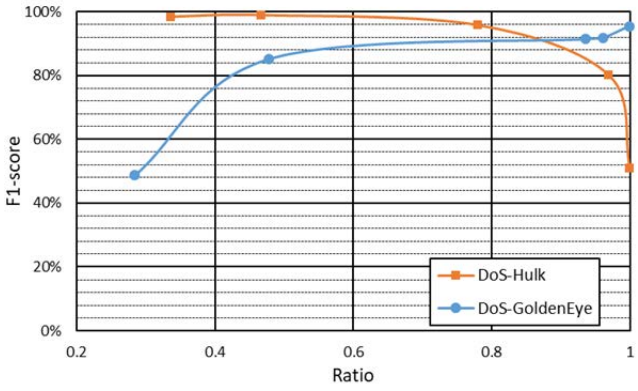


FIGURE 1. F1-scores of some selected classes according to the ratio of training dataset size to test dataset size when CIC-IDS 2017 dataset is used.

Tables 2 to 5 show the experimental results from configuring the training dataset and the test dataset based on θ_i for Brute Force-SSH, DDoS, DoS-HTTP, and Infiltration using the ISCX2012 dataset. For Brute Force-SSH, when the ML model is trained only with forward packet count values smaller than θ_i (as shown in Table 2) the class cannot be detected at all using the model. On the other hand, as shown in Tables 3 to 5, when the training and test datasets are randomly configured for Brute Force-SSH, more than 98.5% were detected. As for the detection rates of the other classes (DDoS, DoS-HTTP, and Infiltration) shown in Tables 3, 4,

TABLE 2. Confusion matrix where the training dataset and test dataset, respectively, are composed of sessions with small forward packet count values and sessions with large forward packet count values for Brute Force-SSH, whereas training and test datasets for other classes are randomly composed. Columns and rows of the matrix represent instances of actual and predicted classes, respectively.

	Brute Force-SSH	DDoS	DoS-HTTP	Infiltration	Normal
Brute Force-SSH	0	0	0	0	3
DDoS	0	41,290	2	0	1,890
DoS-HTTP	0	0	1,097	33	4
Infiltration	0	0	160	2,313	5
Normal	3,010	3,514	49	93	185,219

TABLE 3. Confusion matrix where the training dataset and test dataset, respectively, are composed of sessions with small forward packet count values and sessions with large forward packet count values for DDoS, whereas training and test datasets for other classes are randomly composed. Columns and rows of the matrix represent instances of actual and predicted classes, respectively.

	Brute Force-SSH	DDoS	DoS-HTTP	Infiltration	Normal
Brute Force-SSH	2,966	0	0	0	1
DDoS	0	6	3	0	2,094
DoS-HTTP	0	0	1,104	30	7
Infiltration	0	0	164	2,316	5
Normal	44	44,798	37	93	185,014

TABLE 4. Confusion matrix where the training dataset and test dataset, respectively, are composed of sessions with small forward packet count values and sessions with large forward packet count values for DoS-HTTP, whereas training and test datasets for other classes are randomly composed. Columns and rows of the matrix represent instances of actual and predicted classes, respectively.

	Brute Force-SSH	DDoS	DoS-HTTP	Infiltration	Normal
Brute Force-SSH	2,971	0	0	0	2
DDoS	0	41,039	6	0	1,616
DoS-HTTP	0	0	68	48	7
Infiltration	0	0	34	2,308	4
Normal	39	3,765	1,200	83	185,492

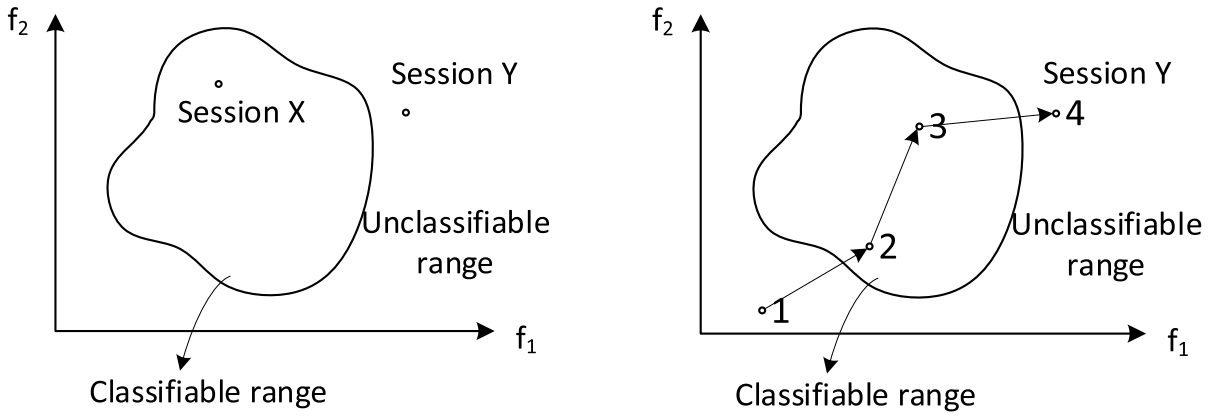
TABLE 5. Confusion matrix where the training dataset and test dataset, respectively, are composed of sessions with small forward packet count values and sessions with large forward packet count values for the Infiltration class, whereas training and test datasets for other classes are randomly composed. Columns and rows of the matrix represent instances of actual and predicted classes, respectively.

	Brute Force-SSH	DDoS	DoS-HTTP	Infiltration	Normal
Brute Force-SSH	2,972	0	0	22	1
DDoS	0	40,844	4	0	1,652
DoS-HTTP	0	0	1,087	22	8
Infiltration	0	0	166	74	9
Normal	38	3,960	51	2,321	185,451

and 5, only 0.01%, 5.2%, and 3% were detected, respectively, and all classes showed similar results. In the end, it was confirmed that when the range of the value of the forward packet count configured in the training dataset and the range of the value of the forward packet count configured in the test dataset were different, the classification accuracy was significantly affected.

As shown in Tables 2 to 5, sessions with a count value greater than the number of forward packets in the training dataset are hardly ever detected, regardless of the class type. In an actual network, the forward packet count value simply increases as the attack continues. That is, it is relatively easy to make it outside the range of the forward packet count of the training data (compared to other features), but the impact on the existing ML-NIDS is very high.

One of the solutions for this is to collect various sessions, including sessions with from a very small forward packet count to a very large count. However, this method not only makes the training dataset too large, but also makes it quite difficult to obtain sufficient training data without any missing value because the range of the forward packet count values is large. In addition, when the size of the training dataset increases, training time greatly increases due to the bigger dataset, and detection speed may decrease as the complexity of the training model increases. Therefore, increasing the size of the training dataset in order to increase the forward packet count value cannot be a fundamental solution. In the end, malicious users can disarm the existing ML-NIDS by performing an attack to increase the forward packet count, and they can easily bypass detection regardless of the dataset



(a) Session X within the classifiable area, and session Y within the unclassifiable area in the feature space

(b) Path trace of each packet for session Y in the feature space

FIGURE 2. Classifiable and unclassifiable regions of sessions in the feature space.

size, but an effective method to prevent this has not yet been presented.

This study tries to effectively solve this problem by adjusting the detection timing, instead of expanding the detection range of the training dataset in the feature space. Figure 2 is a conceptual diagram showing two ranges within which the ML model can and cannot classify sessions when the model is trained on a dataset consisting of two features. According to Figure 2 (a), session X can be classified, so the ML classifier can determine whether it is an intrusion or a benign session. On the other hand, it is impossible to classify session Y using the ML classifier because it is located in an area that cannot be classified.

Now let us assume that session Y consists of four packets. We also assume that whenever each packet is received, the NIDS cumulatively creates from the first packet an intermediate session feature using the currently received packet, and plots it in the feature space as shown in Figure 2 (b). The number on the path shown in Figure 2 (b) represents the number of packets used to create the feature. For example, 2 in Figure 2 (b) indicates a session feature created using the first and second packets.

In Figure 2 (b), the features when the first and fourth packets are received are located in the unclassifiable range in the feature space, whereas the features when the second and third packets are received are located within the classifiable range. Therefore, if we find the right timing at which the corresponding session can be accurately classified, instead of classifying it when the session is terminated, we can classify the session correctly before the end of the session. Now let us discuss in detail how this idea can be implemented.

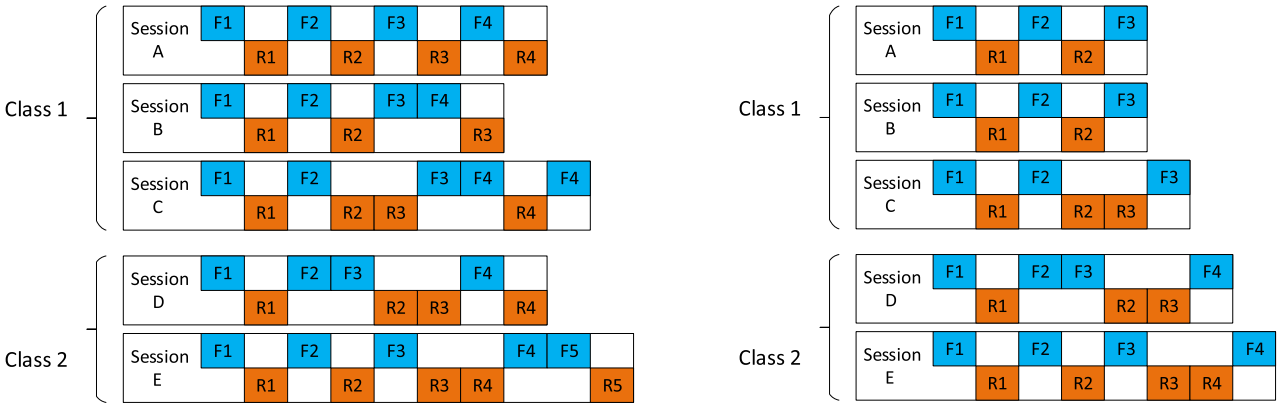
B. THE PROPOSED ALGORITHM

The algorithm should classify a session when the intermediate session feature exists in the classifiable area even before the session is terminated. However, determining if the session feature is in an unclassifiable area for the currently on-going session is difficult. Of course, if intermediate session features are created, and if classification is performed using them on every received packet, it may be possible to classify the session when the intermediate session feature exists in a classifiable region in the feature space even before the session is terminated. However, this incurs very high calculation and memory costs, which means it requires a very expensive, high-end platform that far exceeds the performance of the currently existing NIDS. As a result, in terms of cost, it is infeasible to classify every received packet.

To solve this problem, the proposed method uses the following approach. For a specific feature, a range of values that can be well classified for each class is determined in advance, and classification is attempted only when the intermediate session feature for the currently received session is included within the range. Here, the range of the feature for each class is determined, because the range of the training data for the corresponding feature may be different for each class. In this case, it is advantageous to select a feature type that can be easily calculated and that has great influence on classification accuracy. In this paper, we chose the forward packet count as the feature for the decision, since it meets all conditions.

In the proposed method, the learning process is the same as that of the existing ML-NIDS. Also, it is the same when the session ends in that the corresponding session feature is created to perform classification. However, the classification process differs from that of the existing classification in the following aspect.

By analyzing the training dataset, the maximum forward packet count value (θ_i) for each class except benign class is precalculated. Thus, we will have $N - 1$ values at most due to duplication, when N is the total number of classes. If the forward packet count of the currently received packet matches one of these values, an intermediate session feature of the session the packet belongs to is created and classified. At this time, if the classification result is a class in which



(a) Original dataset

(b) Modified dataset for early classification when Θ_1 and Θ_2 are set to 3 and 4.

FIGURE 3. The comparison original and modified datasets with five sessions belonging to two classes where F_k and R_k stand for the k-th forward and backward packets.

the maximum forward packet count has the same value as the packet count of the currently received packet, the packet will be processed according to the classification result. For example, the packet will be dropped and the result will be logged or notified to an administrator Otherwise, the current classification result is ignored, and the next classification is reperformed whenever the packet number matches one of the maximum forward packet counts again, or when the session ends.

Figure 3 shows how to obtain the training dataset from the original dataset using precalculated θ_i. For training dataset, each session of a class i is normalized according to θ_i. Such a normalized dataset is greatly helpful to avoid classification on the unclassifiable range of the class.

In this method, each class undergoes classification within the classifiable feature values, while at the same time adjusting the classification timing so that each class is best classified. Here, since the intrusion class tends to be classified as benign, if the result of each classification is benign, the result is ignored. Only when the session is finished and classified as benign, the packet is processed as benign. The detailed operation of the proposed method is in Algorithm 1. The classification is only performed when P is the last packet of the session or $n(P) \in \Theta$, the computational complexity of Algorithm 1 is $O(|\Theta| + 1) = O(N + 1) = O(N)$, where N is the class number. Usually, the class number is smaller than the session length. It means that the algorithm has lower complexity than the per packet detection approach.

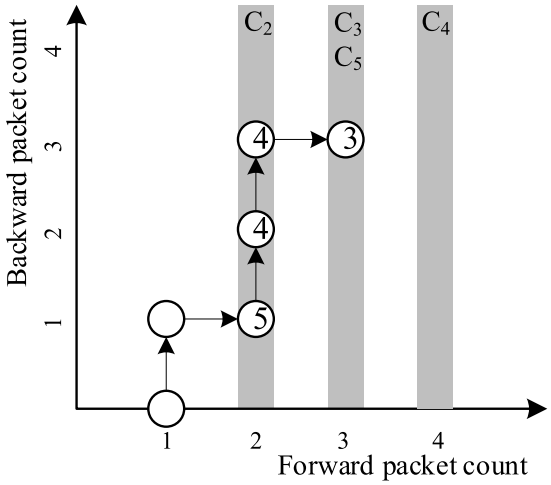
To make it easier to understand the operation of the proposed algorithm, Figure 4 illustrates two cases in which the proposed method finally obtains the classification result. As shown in the figure, classification is performed only when $\theta(C_i)$ and forward packet count are the same, thus reducing the overall classification overhead and increasing the possibility of completing classification before the forward packet count becomes too large. By doing so, the proposed approach improves classification speed and classification accuracy simultaneously.

Figure 5 shows the entire procedure of the proposed algorithm. It calculates the maximum forward packet count for each class, build the training set using the counts. It then tries to classify the received packet to infer the class of the session that the packet belongs to.

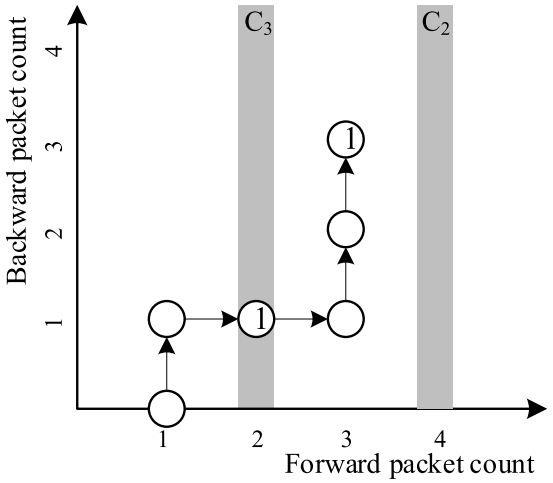
Algorithm 1 Proposed NIDS Classification

Input: $C = \{C_1, C_2, \ldots, C_N\}$ where N is the total number of classes, and C_1 denotes a benign class.
$\theta(C_i)$: the maximum packet count value for C_i in the training dataset. i.e., θ_i. $\Theta = \{\theta(C_2), \theta(C_3), \ldots, \theta(C_N)\}$.
P: the current received packet.
n(P): the maximum forward packet count of P in the session.
F(P): the intermediate session features created from the first to P packets.
Output: Class ID if found
None, otherwise

```
IF P is the last packet of the session THEN
    C_est = classifier(F(P))
    Return C_est
ELSE
    IF n(P) ∈ Θ THEN
        C_est = classifier(F(P))
        IF θ(C_est) == n(P) THEN
            Return C_est
        ELSE
            Postpone the decision until the
            next packet is received.
        ENDIF
    ENDIF
ENDIF
```



(a) Case 1: When C_i matching the classification result for the forward packet count: lines 7-8 of Algorithm 1, where $\theta(C_2)$=2, $\theta(C_3)$ =3, $\theta(C_4)$=4, and $\theta(C_5)$=3.



(b) Case 2: When the session finished before find C_i matching the classification result for the forward packet count: lines 1-3 of Algorithm 1, where $\theta(C_2)$=4 and $\theta(C_3)$=2.

FIGURE 4. Two cases that the proposed algorithm classifies an incoming. Each circle represents each packet of the session. The empty and numbered circles denote no classification and classification result, i.e., class ID, respectively.

IV. PERFORMANCE EVALUATION

In order to accurately evaluate and analyze the proposed method, various datasets and several classification algorithms were used to analyze its performance in various environments. For the evaluation, six algorithms were selected: Random forest [28], Adaboost decision tree [29], XGBoost [30], Extreme learning machine (ELM) [31], Deep neural network (DNN) [32], and Convolutional neural network (CNN) [17]. By including from the deep learning to the decision tree–based method, we compare how the proposed method affects performance when applied to various algorithms.

A. EVALUATION ENVIRONMENT

It is important to use multiple datasets, because characteristics within the same class may differ, depending on the network environment in which data are collected. In this experiment, three datasets were used: ISCX2012, CIC-IDS2017, and CSE-CIC-IDS2018 [33], [34]. Here, minor classes were excluded. In addition, classes having only one forward packet count value were excluded because there is no need to apply the proposed method. For example, in PortScan from CIC-IDS2017, there is no need to apply the proposed method because sessions with one forward packet count comprise 99.5% of the total data. For the same reason, FTP-Brute Force and DoS-SlowHTTPTest with only one forward packet count value were excluded from CSE-CIC-IDS2018. The total numbers of classes of ISCX2012, CIC-IDS2017, and CSE-CIC-IDS2018 are 6, 9, and 8 respectively.

To measure the performance of the proposed method, it is necessary to create a training dataset consisting of small forward packet counts and a test dataset consisting of large counts. To this end, in the distribution according to the forward packet count size for each class, all the session data were divided at a ratio of 7:3 to create training and test datasets. Exceptionally, if the distribution of forward packet counts is U-shaped, training and test datasets were built by dividing all the data based on the minimum point. Only the benign class created training and test datasets by randomly dividing the data at 7:3, regardless of the forward packet count value. The data sizes for the classes are shown in Tables 6 to 8. To evaluate each classification models, the following metrics were used:

- Accuracy $= \frac{TP+TN}{TP+TN+FP+FN}$
- Precision $= \frac{TP}{TP+FP}$
- Recall $= \frac{TP}{TP+FN}$
- F1-score $= \frac{2}{1/Precision+1/Recall}$
- Error rate $= \frac{FP+FN}{TP+TN+FP+FN}$

where TP, TN, FP and FN stand for true positive, true negative, false positive and false negative.

TABLE 6. The ISCX2012 dataset.

Class	θ_i	Training	Test
Benign	-	436,615	187,121
Brute Force-SSH	7	6,367	3,665
DDoS	57	113,407	35,937
HTTPDoS	4	3,558	799
Infiltration	4	7,674	453
Total	-	567,621	227,975

B. DETECTION RATE

The results of applying the proposed method to the three datasets and six machine learning algorithms are shown in Figures 6 to 8. Comparing the results of applying and not

TABLE 7. The CIC-IDS2017 dataset.

Class	θ_i	Training	Test
Benign	-	318,221	136,381
Bot	23	1,933	33
DDoS	5	103,709	24,318
DoS GoldenEye	9	9,632	661
DoS Hulk	4	107,932	123,032
DoS SlowHttpTest	14	5,228	201
DoS Slowloris	9	3,978	1,815
FTP-Patator	6	3,979	3,959
SSH-Patator	14	2,953	2,944
Total	-	557,565	293,344

TABLE 8. The CSE-CIC-IDS2018.

Class	θ_i	Training	Test
Benign	-	355,169	152,216
Bot	24	143,000	95
Brute Force-SSH	19	46,668	47,126
Brute Force-WEB	38	476	135
Brute Force-XSS	83	129	101
DoS-GoldenEye	4	31,001	10,507
DoS-Hulk	2	187,318	43,511
DoS-Slowloris	4	6,007	4,983
Total	-	769,768	258,674

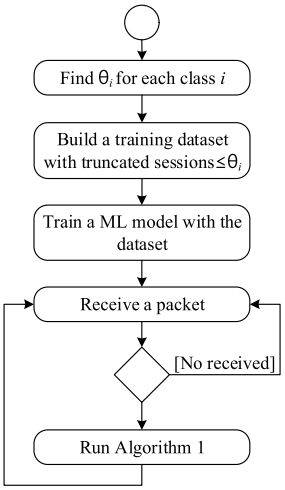


FIGURE 5. The overall procedure of the proposed approach.

applying the proposed method to each algorithm, we see that most of the performances improved when the proposed method was applied. Among a total of 18 test cases, the only case where the performance did not improve (based on the F1-score) was the DNN with the proposed algorithm using the CSE-CIC-IDS2018 dataset. Also, in Figures 6 to 8, we found that the overall sensitivity of the deep learning–based algorithm to the forward packet count feature was higher than DT-based algorithms. Even if the proposed method was applied, the ELM, DNN, and CNN all had F1-scores of 30% to 50%, whereas RF, Adaboost DT, and XGBoost show F1-scores higher than 75%. Therefore, it can be argued that deep learning algorithms have great difficulty in improving performance if sufficient training datasets are not collected.

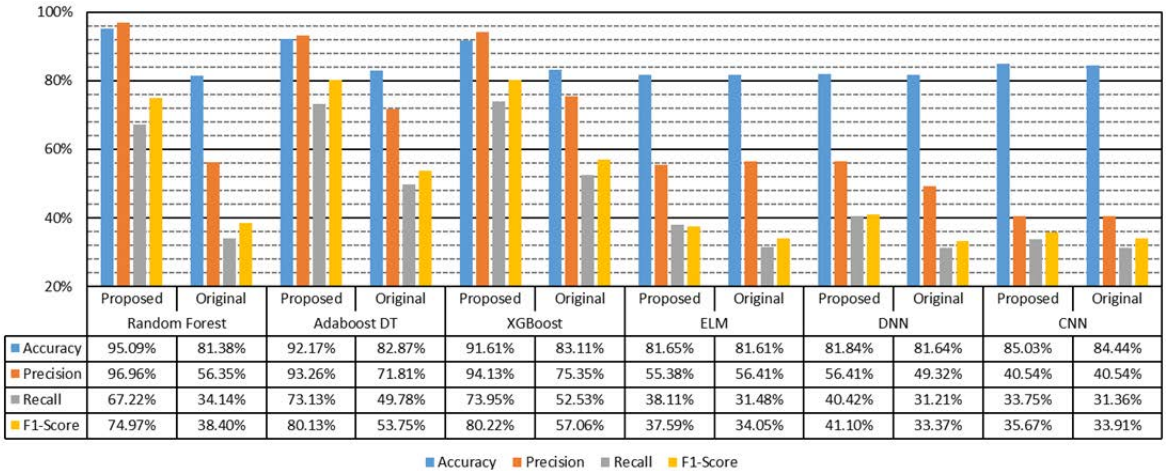
RF, Adaboost DT, and XGBoost also had very low F1-scores if the proposed method was not applied. In applying XGBoost to CSE-CIC-IDS2018, as an exception, it shows a high F1-score of 86.15% even without using the proposed method. However, in other datasets, the F1-score is only about 10% higher than deep learning.

When the proposed method was applied to DT-based algorithms the F1-score improved by up to 32%. The highest F1-score was obtained by applying the proposed method to XGBoost with ISCX2012, which achieved 80.22%. With CIC-IDS2017, applying the proposed method to XGBoost achieved 94.21%, and with CSE-CIC-IDS2018, the F1-score reached 90.49% when applying the proposed method to RF. Depending on the dataset, the optimal algorithm may be different, but we confirmed it is essential to apply the proposed method.

Additionally, Figure 9 shows the error rates of the proposed and original approaches. For most cases, our algorithm shows smaller error rate than the original one regardless of types of ML algorithm and dataset.

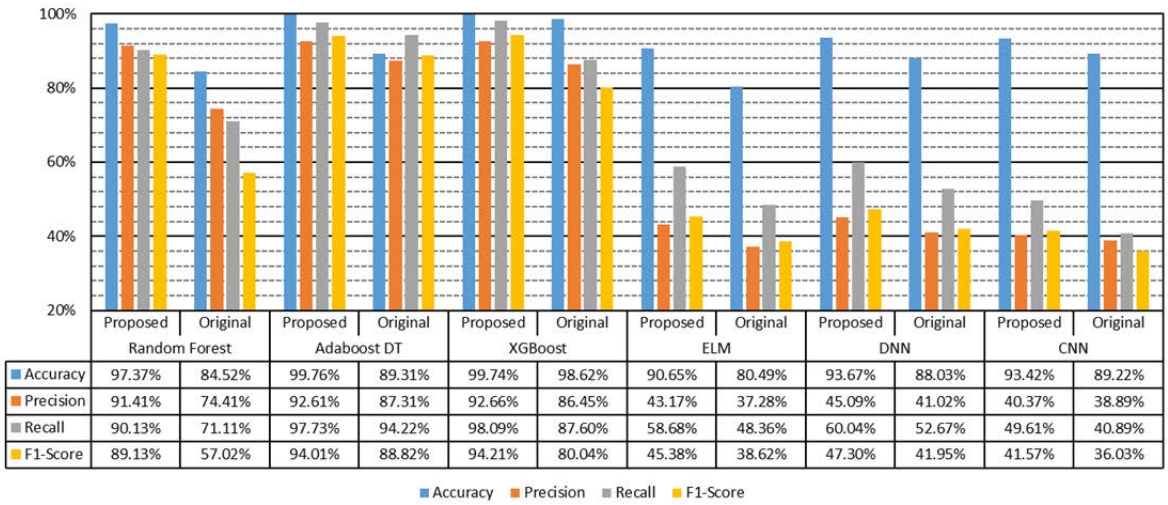
For more detail, it is necessary to analyze the performance of each class when the proposed method is applied. We chose the case where the proposed method is applied to RF with the CSE-CIC-IDS2018 dataset for detailed analysis. According to Figure 10, the F1-score improved, or was at least maintained, after applying the proposed method to all classes except for Brute Force-XSS. In particular, Brute Force-WEB, DoS-GoldenEye, and DoS-Slowloris show that extremely low F1-scores of 7%, 0%, and 1.6% significantly improved to 95.7%, 64.7%, and 96.1% after applying the proposed method.

Table 9 shows the confusion matrix from Figure 10 for a more accurate analysis. If the proposed method is not applied



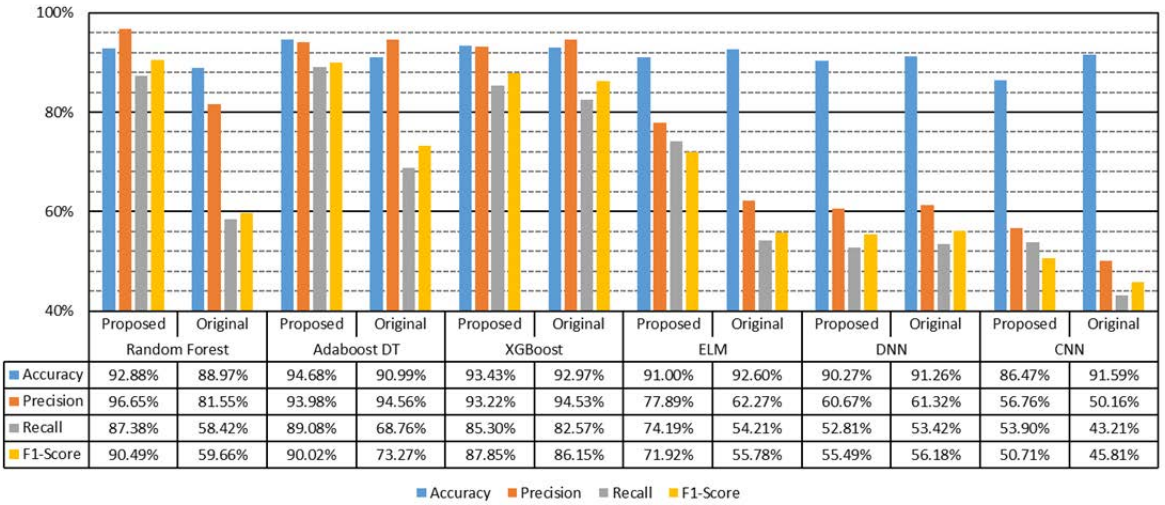
	Random Forest		Adaboost DT		XGBoost		ELM		DNN		CNN	
	Proposed	Original	Proposed	Original	Proposed	Original	Proposed	Original	Proposed	Original	Proposed	Original
Accuracy	95.09%	81.38%	92.17%	82.87%	91.61%	83.11%	81.65%	81.61%	81.84%	81.64%	85.03%	84.44%
Precision	96.96%	56.35%	93.26%	71.81%	94.13%	75.35%	55.38%	56.41%	56.41%	49.32%	40.54%	40.54%
Recall	67.22%	34.14%	73.13%	49.78%	73.95%	52.53%	38.11%	31.48%	40.42%	31.21%	33.75%	31.36%
F1-Score	74.97%	38.40%	80.13%	53.75%	80.22%	57.06%	37.59%	34.05%	41.10%	33.37%	35.67%	33.91%

FIGURE 6. The comparison of detection rates for each classification algorithm with the ISCX2012 dataset.



	Random Forest		Adaboost DT		XGBoost		ELM		DNN		CNN	
	Proposed	Original	Proposed	Original	Proposed	Original	Proposed	Original	Proposed	Original	Proposed	Original
Accuracy	97.37%	84.52%	99.76%	89.31%	99.74%	98.62%	90.65%	80.49%	93.67%	88.03%	93.42%	89.22%
Precision	91.41%	74.41%	92.61%	87.31%	92.66%	86.45%	43.17%	37.28%	45.09%	41.02%	40.37%	38.89%
Recall	90.13%	71.11%	97.73%	94.22%	98.09%	87.60%	58.68%	48.36%	60.04%	52.67%	49.61%	40.89%
F1-Score	89.13%	57.02%	94.01%	88.82%	94.21%	80.04%	45.38%	38.62%	47.30%	41.95%	41.57%	36.03%

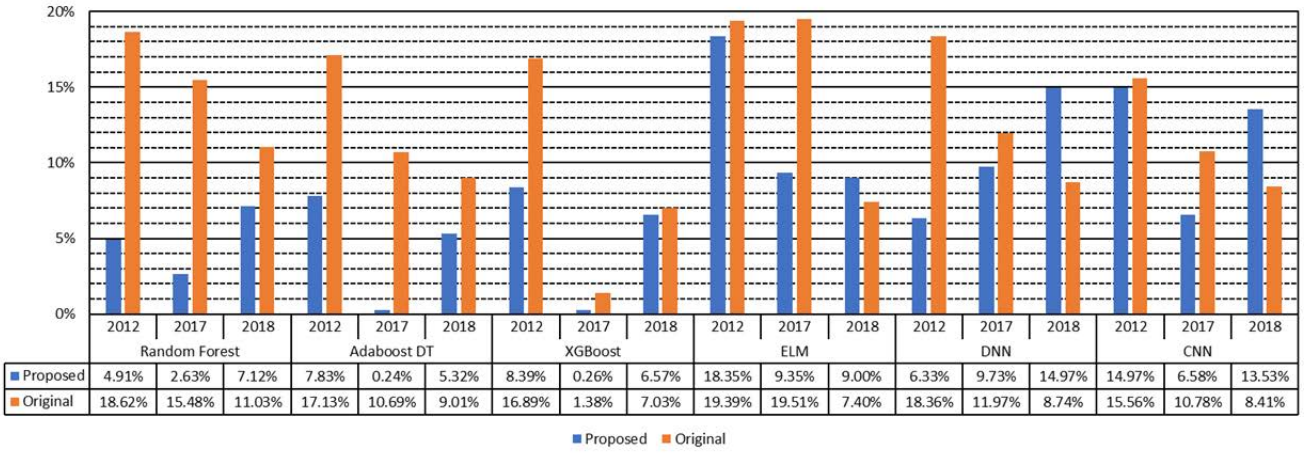
FIGURE 7. Comparison of detection rates for each classification algorithm with the CIC-IDS2017 dataset.



	Random Forest		Adaboost DT		XGBoost		ELM		DNN		CNN	
	Proposed	Original	Proposed	Original	Proposed	Original	Proposed	Original	Proposed	Original	Proposed	Original
Accuracy	92.88%	88.97%	94.68%	90.99%	93.43%	92.97%	91.00%	92.60%	90.27%	91.26%	86.47%	91.59%
Precision	96.65%	81.55%	93.98%	94.56%	93.22%	94.53%	77.89%	62.27%	60.67%	61.32%	56.76%	50.16%
Recall	87.38%	58.42%	89.08%	68.76%	85.30%	82.57%	74.19%	54.21%	52.81%	53.42%	53.90%	43.21%
F1-Score	90.49%	59.66%	90.02%	73.27%	87.85%	86.15%	71.92%	55.78%	55.49%	56.18%	50.71%	45.81%

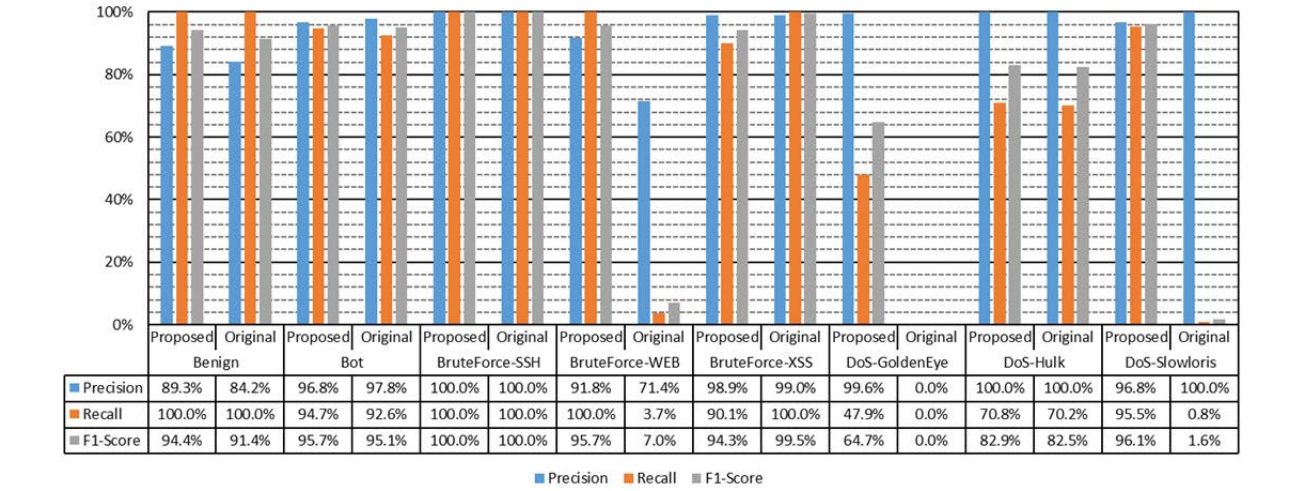
FIGURE 8. Comparison of detection rates for each classification algorithm with the CSE-CIC-IDS2018 dataset.

in the confusion matrix, intrusions are often falsely detected as benign. On the other hand, when the proposed method is used as shown in Table 10, the number of cases in which intrusions are falsely detected as benign was greatly reduced. For example, 96% of the Brute Force-Web class was falsely detected originally, but when the proposed method was



	Random Forest			Adaboost DT			XGBoost			ELM			DNN			CNN		
	2012	2017	2018	2012	2017	2018	2012	2017	2018	2012	2017	2018	2012	2017	2018	2012	2017	2018
Proposed	4.91%	2.63%	7.12%	7.83%	0.24%	5.32%	8.39%	0.26%	6.57%	18.35%	9.35%	9.00%	6.33%	9.73%	14.97%	14.97%	6.58%	13.53%
Original	18.62%	15.48%	11.03%	17.13%	10.69%	9.01%	16.89%	1.38%	7.03%	19.39%	19.51%	7.40%	18.36%	11.97%	8.74%	15.56%	10.78%	8.41%

FIGURE 9. The comparison of error rate for each classification algorithm with ISCX2012, CIC-IDS2017 and CSE-CIC-IDS2018 datasets.



	Benign		Bot		BruteForce-SSH		BruteForce-WEB		BruteForce-XSS		DoS-GoldenEye		DoS-Hulk		DoS-Slowloris	
	Proposed	Original	Proposed	Original	Proposed	Original	Proposed	Original	Proposed	Original	Proposed	Original	Proposed	Original	Proposed	Original
Precision	89.3%	84.2%	96.8%	97.8%	100.0%	100.0%	91.8%	71.4%	98.9%	99.0%	99.6%	0.0%	100.0%	100.0%	96.8%	100.0%
Recall	100.0%	100.0%	94.7%	92.6%	100.0%	100.0%	100.0%	3.7%	90.1%	100.0%	47.9%	0.0%	70.8%	70.2%	95.5%	0.8%
F1-Score	94.4%	91.4%	95.7%	95.1%	100.0%	100.0%	95.7%	7.0%	94.3%	99.5%	64.7%	0.0%	82.9%	82.5%	96.1%	1.6%

FIGURE 10. Comparison of detection rates for each class with the CSE-CIC-IDS2018 dataset.

TABLE 9. Confusion matrix for RF and the proposed algorithm with the CSE-CIC-IDS2018 dataset. Columns and rows represent instances of actual and predicted classes, respectively.

	Benign	Bot	Brute Force-SSH	Brute Force-WEB	Brute Force-XSS	DoS-GoldenEye	DoS-Hulk	DoS-Slowloris
Benign	152,208	5	0	0	0	5,321	12,663	224
Bot	3	90	0	0	0	0	0	0
Brute Force-SSH	0	0	47,126	0	0	0	0	0
Brute Force-WEB	2	0	0	135	10	0	0	0
Brute Force-XSS	1	0	0	0	91	0	0	0
DoS-GoldenEye	2	0	0	0	0	5,033	20	0
DoS-Hulk	0	0	0	0	0	0	30,823	0
DoS-Slowloris	0	0	0	0	0	153	5	4,759

applied, Brute Force-Web was detected with 100% accuracy. In addition, after applying the proposed method, the F1-score decreased by 5.2% for Brute Force-XSS, but looking at the confusion matrix, 10% of Brute Force-XSS sessions were falsely detected as Brute Force-WEB, so the intrusion was successfully detected as an intrusion. Therefore, the proposed

TABLE 10. Confusion matrix for RF without the proposed algorithm using the CSE-CIC-IDS2018 dataset. Columns and rows represent instances of actual and predicted classes, respectively.

Class	Benign	Bot	Brute Force-SSH	Brute Force-WEB	Brute Force-XSS	DoS-GoldenEye	DoS-Hulk	DoS-Slowloris
Benign	152,210	7	0	130	0	10,507	12,951	4,942
Bot	2	88	0	0	0	0	0	0
Brute Force-SSH	0	0	47,126	0	0	0	0	0
Brute Force-WEB	2	0	0	5	0	0	0	0
Brute Force-XSS	1	0	0	0	101	0	0	0
DoS-GoldenEye	1	0	0	0	0	0	0	0
DoS-Hulk	0	0	0	0	0	0	30,560	0
DoS-Slowloris	0	0	0	0	0	0	0	41

TABLE 11. Average detection length using random forest with the ISCX2012 dataset.

Label	Average session length	Average detection length (packets)	Average detection length (sessions)	Average total detection length
Benign	11.0	6.9	11.0	10.9
Brute Force-SSH	10.8	7.0	10.9	9.5
DDoS	119.5	57.0	134.5	70.1
DoS-HTTP	20.9	4.1	186.0	18.6
Infiltration	178.2	4.0	229.5	172.2
Total average	28.5	50.4	15.3	20.6

TABLE 12. Average detection length using random forest with the CIC-IDS2017 dataset.

Label	Average session length	Average detection length (packets)	Average detection length (sessions)	Average total detection length
Benign	13.1	2.9	13.1	13.1
Bot	29.8	23.0	30.3	25.0
DDoS	8.0	5.0	7.2	5.1
DoS-GoldenEye	11.8	7.6	15.5	8.8
DoS-Hulk	7.4	5.0	6.6	5.0
DoS-SlowHTTPTest	17.4	13.8	23.0	13.8
DoS-Slowloris	14.7	9.0	15.0	9.0
FTP-Patator	9.0	8.0	9.0	8.3
SSH-Patator	21.2	14.0	24.2	14.2
Total average	10.3	5.3	12.9	9.0

method not only improved the average F1-score, but also significantly increased the intrusion detection rate for each session.

C. AVERAGE SESSION LENGTH REQUIRED FOR DETECTION

The most important performance metric in the NIDS is detection rate. In addition, detection speed is also a very important metric—the faster the NIDS detects network intrusions, the more it helps keep the network safe. The proposed method is not designed to increase detection speed; nevertheless, it is important to check whether the speed is increased or decreased by it. Therefore, in this experiment, we analyzed detection speed before and after using the proposed method. In order to measure detection speed, we measured how many packets were received in each session until detection was completed. The experimental results for each dataset are shown in Tables 11 to 13. As seen in Algorithm 1, the proposed method includes both packet-based detection and detection of session-based behaviors; therefore, we measured the number of packets required for both. With ISCX2012, the proposed method detected intrusions slightly faster than the existing method. For a DDoS, the proposed method can reduce the number of packets required for detection by 40%, and for the entire classes, it can reduce the number by 28% on average. Although the proposed method is not designed to focus on improving detection speed, it significantly improved speed, thus proving it is of great help in improving the performance of an existing NIDS.

With the CIC-IDS2017 dataset, there was a significant improvement for some classes, such as a 36% performance

TABLE 13. Average detection length using random forest with the CSE-CIC-IDS2018 dataset.

Label	Average session length	Average detection length (packets)	Average detection length (sessions)	Average total detection length
Benign	6.9	5.5	6.9	6.9
Bot	53.0	21.0	73.0	26.5
Brute Force-SSH	22.4	19.0	22.1	19.2
Brute Force-WEB	151.2	38.0	0.0	38.0
Brute Force-XSS	202.7	78.5	0.0	78.5
DoS-GoldenEye	5.7	4.0	6.2	5.1
DoS-Hulk	3.2	2.0	3.7	2.5
DoS-Slowloris	14.4	4.0	14.9	4.5
Total average	9.3	11.2	6.9	8.3

TABLE 14. The average number of classifications for each class from using the ISCX2012 dataset.

Class	Benign	Brute Force-SSH	DDoS	HTTPDoS	Infiltration	Total
C/F number	1.33	2.65	3	1.1	2.24	1.58

TABLE 15. The average number of classifications for each class from using the CIC-IDS2017 dataset.

Class	Benign	Bot	DDoS	DoS GoldenEye	DoS Hulk	DoS Slowhttptest	DoS Slowloris	FTP-Patator	SSH-Patator	Total
C/F number	1.47	5	1.04	2.62	1.03	3.97	3	2.32	4.01	1.29

TABLE 16. The average number of classifications for each class from using the CSE-CIC-IDS2018 dataset.

Class	Benign	Bot	Brute Force-SSH	Brute Force-WEB	Brute Force-XSS	DoS-GoldenEye	DoS-Hulk	DoS-Slowloris	Total
C/F number	2.76	4.13	3.15	5	5.9	2.51	1.34	2.04	2.57

improvement against a DDoS, and a 33% performance improvement against SSH-Patator.

Unlike the other two datasets, the CSE-CIC-IDS2018 dataset showed significant performance improvement for classes with a very long detection length. For example, if the proposed method is not used, detecting Brute Force-WEB and Brute Force-XSS required 151.2 and 202.7 packets, respectively, whereas using the proposed method, only 38 and 78.5 packets needed to be received, improving performance by 75% and 61%.

In the results from using the three datasets, detection speed for most classes improved, and only a few classes showed the same performance. In particular, the longer the session is, a greater improvement in detection speed can be a great advantage. Long sessions usually consume a lot of memory of NIDS, because NIDS require all the data for each packet to create a feature after the session ends. Therefore, the proposed method can classify such long sessions much before session termination, so it can significantly reduce the amount of memory for the sessions, and can improve detection performance at the same time.

D. TOTAL CLASSIFICATION NUMBER REQUIRED FOR DETECTION

Unlike the existing ML-NIDS, the proposed method may make multiple classifications until one session is successfully classified. Now, let the number of classifications be defined as "the number of classifications required until one session is classified." This is important because more classifications require more processing power to classify one session, so a hardware platform with better performance is required. In the end, the closer the number of classifications is to one, the higher the possibility of implementing the proposed algorithm on the existing session-based NIDS hardware platform. As shown in Tables 14 to 16, the average number of classifications for each dataset differed greatly for each class. This is because the total session length for each class and the size of θ_i for each class were different. However, in Tables 14 to 16, the average number of classifications for the entire dataset does not exceed 3. In particular, the average number of classifications with the CIC-IDS2017 and ISCX2012 datasets were 1.29 and 1.58, respectively, which is not a significant increase compared to the existing session-based classification. Therefore, even if the proposed method is implemented on the existing hardware platform, there is no significant performance degradation.

V. CONCLUSION

The most important thing in the ML-NIDS is the training dataset used to create the classifier model. However, it is impossible to obtain a training dataset including all network intrusions that occur in the wild. Rather, it is important to find a way to accurately detect an intrusion by utilizing an

existing dataset, even if the intrusion data it contains are insufficient. In this paper, a new approach to solve this problem is presented. Using various datasets, the proposed method has proven that the weaknesses of the existing ML-NIDS can be greatly improved. Of course, there is still much room for improvement in the proposed method. For example, it may not be sufficient to determine whether the learning range is exceeded using only the forward packet count feature. However, if multiple features are considered, the number of sessions that can be processed per second decreases. Also, for some classes, improvement of the detection rate is not big. Despite these weaknesses, it is a great advantage to be able to broadly expand the classification range in the feature space by using a dataset consisting of limited data. In addition, classification speed can also be improved, so it is expected that the proposed method, when installed in actual NIDS equipment, will be of great help in keeping large networks safe. As our future work, we will focus on how to extend this current result to support multiple features. If the solution is successfully found, ML-NIDS can maximize the classification detection rate without deteriorating the classification speed.

REFERENCES

[1] A. Borkar, A. Donode, and A. Kumari, "A survey on intrusion detection system (IDS) and internal intrusion detection and protection system (IIDPS)," in *Proc. Int. Conf. Inventive Comput. Informat. (ICICI)*, Nov. 2017, pp. 949–953, doi: 10.1109/ICICI.2017.8365277.

[2] Z. Zhou, C. Zhongwen, Z. Tiecheng, and G. Xiaohui, "The study on network intrusion detection system of snort," in *Proc. Int. Conf. Netw. Digit. Soc.*, May 2010, pp. 194–196, doi: 10.1109/ICNDS.2010.5479341.

[3] M. F. Zolkipli and A. Jantan, "A framework for malware detection using combination technique and signature generation," in *Proc. 2nd Int. Conf. Comput. Res. Develop.*, May 2010, pp. 196–199, doi: 10.1109/ICCRD.2010.25.

[4] H. Zhang, "Design of intrusion detection system based on a new pattern matching algorithm," in *Proc. Int. Conf. Comput. Eng. Technol.*, Jan. 2009, pp. 545–548, doi: 10.1109/ICCET.2009.244.

[5] V. Gupta, M. Singh, and V. K. Bhalla, "Pattern matching algorithms for intrusion detection and prevention system: A comparative analysis," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Sep. 2014, pp. 50–54, doi: 10.1109/ICACCI.2014.6968595.

[6] A. Halimaa A. and K. Sundarakantham, "Machine learning based intrusion detection system," in *Proc. 3rd Int. Conf. Trends Electron. Informat. (ICOEI)*, Apr. 2019, pp. 916–920, doi: 10.1109/ICOEI.2019.8862784.

[7] A. Phadke, M. Kulkarni, P. Bhawalkar, and R. Bhattad, "A review of machine learning methodologies for network intrusion detection," in *Proc. 3rd Int. Conf. Comput. Methodol. Commun. (ICCMC)*, Mar. 2019, pp. 272–275, doi: 10.1109/ICCMC.2019.8819748.

[8] L. Bondan, M. A. Marotta, M. Kist, L. R. Faganello, C. B. Both, J. Rochol, and L. Z. Granville, "Kitsune: A management system for cognitive radio networks based on spectrum sensing," in *Proc. IEEE Netw. Operations Manage. Symp. (NOMS)*, May 2014, pp. 1–9, doi: 10.1109/NOMS.2014.6838316.

[9] R. Gaddam and M. Nandhini, "An analysis of various snort based techniques to detect and prevent intrusions in networks proposal with code refactoring snort tool in kali Linux environment," in *Proc. Int. Conf. Inventive Commun. Comput. Technol. (ICICCT)*, Mar. 2017, pp. 10–15, doi: 10.1109/ICICCT.2017.7975177.

[10] 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.

[11] S. Soheily-Khah, P.-F. Marteau, and N. Bechet, "Intrusion detection in network systems through hybrid supervised and unsupervised machine learning process: A case study on the ISCX dataset," in *Proc. 1st Int. Conf. Data Intell. Secur. (ICDIS)*, Apr. 2018, pp. 219–226.

[12] Y. Yuan, L. Huo, and D. Hogrefe, "Two layers multi-class detection method for network intrusion detection system," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jul. 2017, pp. 767–772.

[13] I. Ahmad, M. Basheri, M. J. Iqbal, and A. Raheem, "Performance comparison of support vector machine, random forest, and extreme learning machine for intrusion detection," *IEEE Access*, vol. 6, pp. 33789–33795, 2018.

[14] Y. Cheong, K. Park, H. Kim, J. Kim, and S. Hyun, "Machine learning based intrusion detection systems for class imbalanced datasets," *J. Korea Inst. Inf. Secur. Cryptol.*, vol. 27, no. 6, pp. 1385–1395, Dec. 2017.

[15] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.

[16] K. Park, Y. Song, and Y.-G. Cheong, "Classification of attack types for intrusion detection systems using a machine learning algorithm," in *Proc. IEEE 4th Int. Conf. Big Data Comput. Service Appl. (BigDataService)*, Mar. 2018, pp. 282–286.

[17] W.-H. Lin, H.-C. Lin, P. Wang, B.-H. Wu, and J.-Y. Tsai, "Using convolutional neural networks to network intrusion detection for cyber threats," in *Proc. IEEE Int. Conf. Appl. Syst. Invention (ICASI)*, Apr. 2018, pp. 1107–1110.

[18] Y. Otoum, D. Liu, and A. Nayak, "DL-IDS: A deep learning–based intrusion detection framework for securing IoT," *Trans. Emerg. Telecommun. Technol.*, p. e3803, 2019, doi: 10.1002/ett.3803.

[19] Z. Wang, "Deep learning-based intrusion detection with adversaries," *IEEE Access*, vol. 6, pp. 38367–38384, 2018.

[20] J. Clements, Y. Yang, A. Sharma, H. Hu, and Y. Lao, "Rallying adversarial techniques against deep learning for network security," 2019, *arXiv:1903.11688*.

[21] O. Ibitoye, O. Shafiq, and A. Matrawy, "Analyzing adversarial attacks against deep learning for intrusion detection in IoT networks," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2019, pp. 1–6.

[22] A. Piplai, S. S. L. Chukkapalli, and A. Joshi, "NAttack! Adversarial attacks to bypass a GAN based classifier trained to detect network intrusion," in *Proc. IEEE IEEE 6th Int. Conf. Big Data Secur. Cloud (BigDataSecurity) Int. Conf. High Perform. Smart Comput., (HPSC) IEEE Int. Conf. Intell. Data Secur. (IDS)*, May 2020, pp. 49–54.

[23] D. Han, Z. Wang, Y. Zhong, W. Chen, J. Yang, S. Lu, X. Shi, and X. Yin, "Evaluating and improving adversarial robustness of machine learning-based network intrusion detectors," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 8, pp. 2632–2647, Aug. 2021.

[24] Z. Lin, Y. Shi, and Z. Xue, "IDSGAN: Generative adversarial networks for attack generation against intrusion detection," 2018, *arXiv:1809.02077*.

[25] X. Peng, W. Huang, and Z. Shi, "Adversarial attack against DoS intrusion detection: An improved boundary-based method," in *Proc. IEEE 31st Int. Conf. Tools with Artif. Intell. (ICTAI)*, Nov. 2019, pp. 1288–1295.

[26] G. Apruzzese, M. Colajanni, and M. Marchetti, "Evaluating the effectiveness of adversarial attacks against botnet detectors," in *Proc. IEEE 18th Int. Symp. Netw. Comput. Appl. (NCA)*, Sep. 2019, pp. 1–8.

[27] M. J. Hashemi and E. Keller, "Enhancing robustness against adversarial examples in network intrusion detection systems," in *Proc. IEEE Conf. Netw. Function Virtualization Softw. Defined Netw. (NFV-SDN)*, Nov. 2020, pp. 37–43, doi: 10.1109/NFV-SDN50289.2020.9289869.

[28] V. Svetnik, A. Liaw, C. Tong, J. Culberson, R. Sheridan, and B. Feuston, "Random forest: A classification and regression tool for compound classification and QSAR modeling," *J. Chem. Inf. Comput. Sci.*, vol. 43, pp. 1947–1958, Nov. 2003.

[29] Y. Coadou, "Boosted decision trees and applications," in *Proc. EPJ Web Conf.*, vol. 55, 2013, p. 02004.

[30] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, San Francisco, CA, USA, Aug. 2016, pp. 785–794.

[31] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, pp. 489–501, Dec. 2006.

[32] Y. Bengio, "Learning deep architectures for AI," *Found. Trends Mach. Learn.*, vol. 2, pp. 1–127, Jan. 2007.

[33] 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, doi: 10.1016/j.cose.2011.12.012.
[34] I. Sharafaldin, A. Habibi 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, doi: 10.5220/0006639801080116.
[35] G. Bovenzi, G. Aceto, D. Ciuonzo, V. Persico, and A. Pescape, "A hierarchical hybrid intrusion detection approach in IoT scenarios," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2020, pp. 1–7.
[36] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 2, pp. 445–458, Feb. 2019.

TAEHOON KIM received the B.S. degree in information and communication engineering from Yeungnam University, in 2018, where he is currently pursuing the M.S. degree. His current research interests include high-speed network intrusion detection and prevention based on machine-learning.

WOOGUIL PAK (Member, IEEE) received the B.S. and M.S. degrees in electrical engineering and the Ph.D. degree in electrical engineering and computer science from Seoul National University, in 1999, 2001, and 2009, respectively. In 2010, he joined the Jangwee Research Institute for National Defence as a Research Professor, and Keimyung University, Daegu, South Korea, in 2013. He is currently an Associate Professor at Yeungnam University, Gyeongsan, South Korea. His current research interests include network and system security, blockchain, user behavior analytics based on machine learning, and network security for high speed networks.