Early Detection of Network Intrusions, One-Class Classifier

Santhoshkumar P
Jayam college of engineering and technology

Abstract- *Early detection of network intrusions is a very important factor in network security. However, most studies of network intrusion detection systems utilize features for full sessions, making it difficult to detect intrusions before a session ends. To solve this problem, the proposed method uses packet data for features to determine if packets are malicious traffic. Such an approach inevitably increases the probability of falsely detecting normal packets as an intrusion or an intrusion as normal traffic for the initial session. As a solution, the proposed method learns the patterns of packets that are unhelpful in order to classify network intrusions and benign sessions. To this end, a new training dataset for Generative Adversarial Network (GAN) is created using misclassified data from an original training dataset by the LSTM-DNN model trained using the original one. The GAN trained with this dataset has ability to determine whether the currently received packet can be accurately classified in the LSTM-DNN. If the GAN determines that the packet cannot be classified correctly, the detection process is canceled and will be tried again when the next packet is received. Meticulously designed classification algorithm based on LSTM-DNN and validation model using GAN enable the proposed algorithm to accurately perform network intrusion detection in real time without session termination or delay time for collecting a certain number of packets. Various experiments confirm that the proposed method can detect intrusions very early (before the end of the session) while maintaining detection performance at a level similar to that of the existing methods.*

Keywords- Intrusion Detection, Generative Adversarial Network, Early Detection, Real-Time Detection

I. INTRODUCTION

Network intrusion detection and prevention systems utilize machine learning for accuracy that exceeds the limits of existing rule-based methods [1-5]. Elaborate and sophisticated machine learning algorithms and powerful hardware accelerators are among the most important elements of today’s intrusion detection system—intrusion prevention system (IDS/IPS) [6][7]. As machine learning models evolve, they require higher processing power, and accordingly, hardware accelerators with higher computational power are being released. In this way, it is possible to classify high-capacity traffic in each session, and detect network intrusions with high accuracy.

When a network intrusion occurs, it is important to detect it without delay and block it to prevent further damage. Nevertheless, a current machine-learning-based IDS/IPS determines whether an intrusion has occurred within each session only after the session has terminated, and traffic is differentiated with a 5-tuple, i.e. <source IP, destination IP, source port, destination port, protocol>, which are key to identifying one session. In traffic divided into sessions, the statistical values of the traffic transmitted and received from the beginning of the session until the end become the features of the session. The machine learning model is trained using such features, and maliciousness is determined on a per-session basis. It means a network intrusion is detected after session termination, which also means it is detected only after the intrusion terminates. Therefore, this approach has limitations in safely protecting the network.

There have been studies to overcome the limitations of a method using session-based features, and directly using the packets to determine features is one representative method. However, since this method also needs to collect a fixed number of packets for each session to determine whether there is an intrusion, it cannot immediately detect one although the detection delay is shorter than that of NIDS using session-based features.

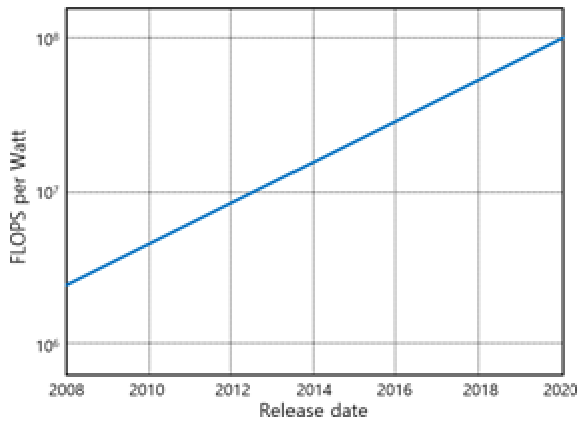


FIGURE 1. GPU performance trends in FLOPS per watt

In order to detect an intrusion as soon as it occurs, several issues must be addressed technically. First of all, the IDS/IPS must be able to inspect every packet received. As hardware technology continues to advance, machine learning computation speed is increasing at an exponential rate, and packet inspection can fully utilize the advantages of parallel processing. So, even if network traffic volume is high, it is expected that intrusion investigation of every packet will be possible, as shown in Figure 1 [8].

Second, it is essential to know exactly when an intrusion occurs within a session but to the best of our knowledge, there are few studies on this. If the entire traffic for one session is investigated simultaneously after the session ends, the characteristics of the intrusion can be identified more accurately, but in order to reduce detection delay, it is necessary to examine the packets before the session ends. However, even for a session with a network intrusion, it is difficult to detect the intrusion accurately because its initial traffic tends not to be significantly different from normal traffic in many cases. Ultimately, to detect an intrusion immediately, the problem of how to determine when accurate detection is possible must be solved.

To solve such problems, this study proposes a new solution combining a Generative Adversarial Network (GAN) with a Long Short-term Memory (LSTM) classifier [9][10]. The GAN learns the characteristics of a given dataset, and can numerically express how likely a data belongs to the class with respect to the training dataset. Therefore, a GAN is very useful when implementing a one- class classifier. In the proposed study, a GAN-based one- class classifier is a decision maker determining the right time for intrusion detection. The classifier learns all packets belonging to misclassified sessions, so it can distinguish between undetectable-intrusion and detectable-intrusion packets. Detectable-intrusion packets are forwarded while they are classified through a multi-class classifier to detect the intrusion class.

This study makes the following contributions.

I. Using a GAN, the one-class classifier identifies packets that cannot be classified exactly This is the first time a one-class classifier has been proposed to determine whether intrusion detection is impossible/possible by using a GAN, as far as we know. A classifier using a GAN can classify with high accuracy any packets from which intrusion detection is impossible.
II. The packet-based classifier supports immediate network intrusion detectionUnlike an existing packet-based classifier, classification time is accurately determined by the GAN-based one-class classifier, and it does not require all (or even a fixed number of) packets belonging to the session. Therefore, when an intrusion packet reaches the IDS/IPS, it is immediately detected as an intrusion.

This paper has the following structure. Section II describes the existing research. Section III introduces the structure of the proposed method in detail. In Section IV, performance comparison results are shown for the proposed method and existing methods. Finally, Section V concludes this paper.

PREVIOUS WORK

In order to detect network intrusions through machine learning, network traffic must be expressed in a specific data format (called features that enable the machine learning model understand the network traffic). Since the performance of machine learning is determined by the features, research on how to generate good features from network traffic is ongoing [11][12]. Currently, most of the machine-learning studies of IDS/IPSs classify traffic into units called a session, and they generate features from characteristic of overall session behavior like statistical values for each session. For example, the total number of packets or the total traffic size transmitted until the end of one session belongs to the features. Such features created for each session in this way are called 'session features'. A session is a logical grouping of traffic, and 5-tuple values <source IP, destination IP, source port, destination port, protocol> are used to distinguish each session. For protocols such as UDP rather than TCP, the session concept is not available, but traffic can be distinguished on a per-session basis using the 5-tuple values. However, in protocols other than TCP, it is difficult to clearly determine the end of a session. Therefore, if a packet having a new 5-tuple (different from those of the existing sessions) is received, the IDS/IPS handles the traffic as a new session, and if a packet belonging to the session is not received for a certain period of time (i.e. a session times out), the session is treated as terminated. This means that in creating a session feature after the session ends, it is difficult to create it without delay caused by time-out. Also, if an intrusion is attempted only for a specific period within a session, the characteristics of the intrusion may be diluted by traffic outside of the intrusion period. This characteristic becomes more pronounced with longer sessions or higher traffic volumes, so exploiting the characteristics of this approach can easily bypass a IDS/IPS that uses the session feature [13][14].

Nevertheless, the reason many IDS/IPSs still use the session feature is that it always has a constant size, regardless of the length of the session, the length of the packets, or the amount of traffic. Such a fixed size of session features makes designing machine learning models easy. Also, since it has low spatial complexity, it is more advantageous for learning larger-capacity traffic. However, in order to perform intrusion detection, it is necessary to classify the received packets by session, and analyse all packets in the session after the session is terminated in order to create a session feature. Therefore, it requires a lot of memory or a high amount of computational power, which is a big obstacle in real-time detection [15]-[21]. When session feature created from single session is used, it becomes difficult to detect an intrusion that leverages multiple simultaneous sessions to increase damage. To solve this problem, it is helpful to use features created from multiple sessions rather than features created from one session. Good examples of such features are the total number of sessions that occurred during a specific time period, or the total amount of traffic received by a specific server from multiple clients. To distinguish this session feature from the existing single-session feature, let us call it an inter-session feature. The inter-session feature is a method that can compensate for the shortcomings of the session feature. However, the inter-session feature requires more complex processing and more storage space than the existing session feature [22][23].

Although the session feature is most commonly used, research using packet data as a feature is also underway to solve the shortcomings of the session feature [24][25]. To directly use packet data as features, features are simply created by applying one-hot encoding to each byte value of the packet data [24]. Therefore, in this case, the total feature size is larger than the packet size. To distinguish a feature created in this way from a session feature, let us call it a packet feature to discriminate it from a session feature. Time- consuming calculations, such as statistical calculations for creating session features, are unnecessary when generating packet features.

When packet feature is used, it is difficult to determine a normal session from an intrusive session using the first packet alone. Since information stored in one packet is quite limited, features built from packets in one session should be collected and used for classification; however, such a procedure is similar to generating a session feature. Basically, a packet feature can be created by collecting data of the same size from the first N packets in a session, or it can be created by collecting all packet data up to a predetermined size, L. Therefore, there is no need to wait until the session ends unlike session feature, but the packet feature also needs time to collect N packets or L bytes for each session. In the end, intrusion detection is delayed if not enough packets are collected, even when a specific packet that attempts an intrusion is received.

If one-hot encoding is applied to each byte to generate packet features, the number of features increases significantly, which is a major obstacle for a machine learning model trying to learn or classify at a high speed. In order to solve this problem, even if accuracy is low, a method of using packet data directly as a feature without one-hot encoding is also being tried. In this case, it helps to speed up the classification, but in order to collect a certain amount of packet data, it is still inevitable that intrusion detection will be delayed.

Session features and packet features represent characteristics about traffic in different ways. Therefore, a method of combining the session feature and the packet feature to create synergy has also been proposed. In this case, it requires a lot of resources to create both features simultaneously, so even with high-performance hardware, it can be too difficult to handle large-capacity traffic. In some studies exploiting two types of features, the overhead from simultaneously creating and maintaining both features can be avoided by using only the packet feature at the beginning of the session and the session feature thereafter separately, instead of using the packet feature and the session feature at the same time. However, real-time intrusion detection cannot be supported, because this method also has to wait until the end of the session to create the session feature if it fails to classify the session only using the packet feature.

So far, existing studies of IDS/IPSs in which machine learning is applied have been reviewed according to the characteristics of features. In order to detect a network intrusion in real time without delay, the existing session-feature-based or packet-feature-based approaches cannot be the solution. For real-time network intrusion detection, an innovative approach is required to eliminate weaknesses in the existing approaches.

III. THE PROPOSED APPROACH

To achieve real-time detection, it is important to use packet data directly as a feature in order to eliminate waiting for the end of the session and shorten the time required to generate feature of the session. At the same time, it is necessary to find a packet that can accurately discriminate whether an intrusion has occurred, and to detect the network intrusion based on it. Existing research does not provide this capability. Therefore, in this study, the following new method is proposed to solve the problem.

A. *SYSTEM ARCHITECTURE*

The proposed method is implemented based on LSTM so that intrusions can be detected on a per-session basis while directly using packet data as features. The received packets are determined to be an intrusion or not through the proposed classifier. If there is no intrusion, the classification result is temporarily stored and used as a feature together with packet data when the next packet belonging to the same session is received. When network intrusions are detected using only packet data, there is a high probability that detection will fail due to fragmented information for the entire session.

In this way, the classification performance from a single packet can be greatly improved, but the remaining unresolved problem is which packet classification in the session will be selected as the final result. In general, determining whether an intrusion has occurred using a packet from the beginning of the session increases the possibility of misclassification. On the other hand, if intrusion detection is performed using a packet close to the end of the session, detection is too late, as mentioned above. Ultimately, it is necessary to determine the packet that can make the fastest determination most accurately. The proposed method attempts to solve this problem by using a GAN. The GAN discriminator is trained using only the packets misclassified by the LSTM classifier for the training dataset, so the characteristics of the packets that are highly likely to be misclassified are accurately learned by the GAN generator.

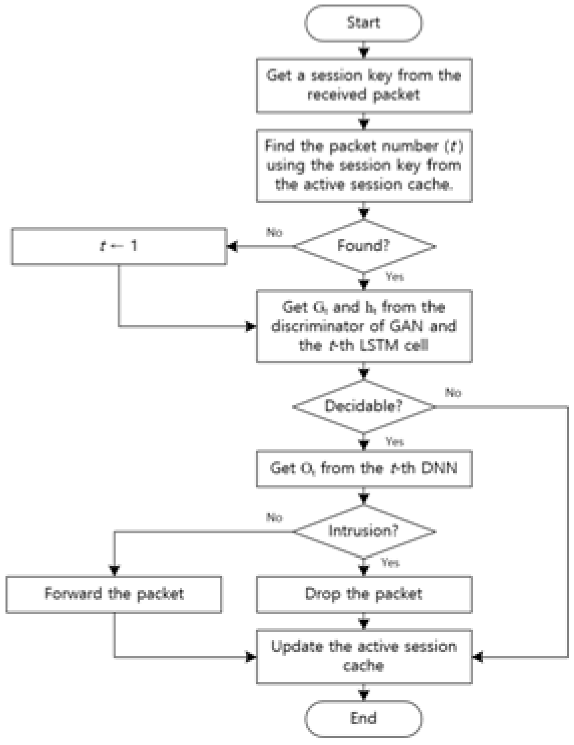

Start
Get a session key from the received packet
Find the packet number (t) using the session key from the active session cache.
Found?
No
t ← 1
Yes
Get G_t and h_t from the discriminator of GAN and the t-th LSTM cell
Decidable?
No
Yes
Get O_t from the t-th DNN
Intrusion?
No
Yes
Forward the packet
Drop the packet
Update the active session cache
End

FIGURE 2. Procedures in the proposed algorithm

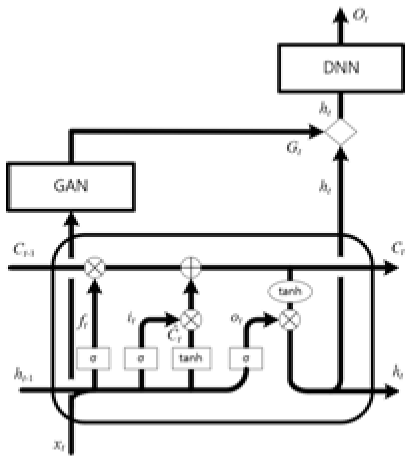


FIGURE 3. The partial structure of the proposed classifier for *t*-th packet

Whenever a packet is received by NIDS/NIPS, the LSTM classifier is used to determine whether the session including the packet is malicious or not, and the result is verified using GAN. If it is determined through the GAN that the reliability of the classification result is low, the classification result is ignored. If it is determined that the reliability is high, the session is processed according to the classification result. The overall flow and structure are as seen in Figures 2 and 3. Each part of the entire system is described in detail below.

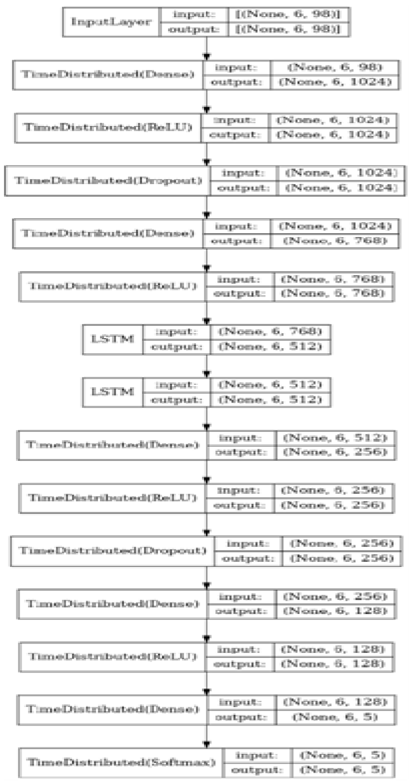

InputLayer | input: [(None, 6, 98)] | output: [(None, 6, 98)]
TimeDistributed(Dense) | input: (None, 6, 98) | output: (None, 6, 1024)
TimeDistributed(ReLU) | input: (None, 6, 1024) | output: (None, 6, 1024)
TimeDistributed(Dropout) | input: (None, 6, 1024) | output: (None, 6, 1024)
TimeDistributed(Dense) | input: (None, 6, 1024) | output: (None, 6, 768)
TimeDistributed(ReLU) | input: (None, 6, 768) | output: (None, 6, 768)
LSTM | input: (None, 6, 768) | output: (None, 6, 512)
LSTM | input: (None, 6, 512) | output: (None, 6, 512)
TimeDistributed(Dense) | input: (None, 6, 512) | output: (None, 6, 256)
TimeDistributed(ReLU) | input: (None, 6, 256) | output: (None, 6, 256)
TimeDistributed(Dropout) | input: (None, 6, 256) | output: (None, 6, 256)
TimeDistributed(Dense) | input: (None, 6, 256) | output: (None, 6, 128)
TimeDistributed(ReLU) | input: (None, 6, 128) | output: (None, 6, 128)
TimeDistributed(Dense) | input: (None, 6, 128) | output: (None, 6, 5)
TimeDistributed(Softmax) | input: (None, 6, 5) | output: (None, 6, 5)

FIGURE 4. The LSTM-DNN-based classifier using packet features

B. *ACTIVE SESSION CACHE*

The active session cache is used to determine whether a received packet is an existing-session packet or a new session packet. Also, packets from the existing session are used to find the total number of received packets in the session and the classification results for previous packets. Here, the classification result is the output of the LSTM cell or the final intrusion-detection result. For example, if the result is intrusion, the all consecutive packets belonging to the session is immediately dropped when it is received. This can greatly improve the performance of the classifier by avoiding unnecessary packet classification.

C. *FEATURE GENERATION*

The first *L* bytes of the IP packet are used to create a feature for classification from the packet data. A typical value for *L* in the experiment is 98 bytes. As *L* is increasing, classifier tends to achieve higher detection rate since it can leverage more meaningful features but it will suffer from larger overhead in terms of memory consumption and classification speed. From the preliminary experiment, the detection rate was measured according to *L* as shown in Table I. By considering detection rate and classification overhead simultaneously, 98 is chosen for the value of *L*.

TABLE 1.
F1-SCORE ACCORDING TO *L* FROM THE PRELIMINARY EXPERIMENT RESULT USING ISCX2012 TRAINING DATASET. *L* IS SET TO EACH VALUE SATISFYING CONSTRAINT OF IMAGE TRANSFORMATION.

L (byte)	50	98	162	242	392	512
F1-score (%)	92.30	92.83	91.41	92.17	91.82	92.92

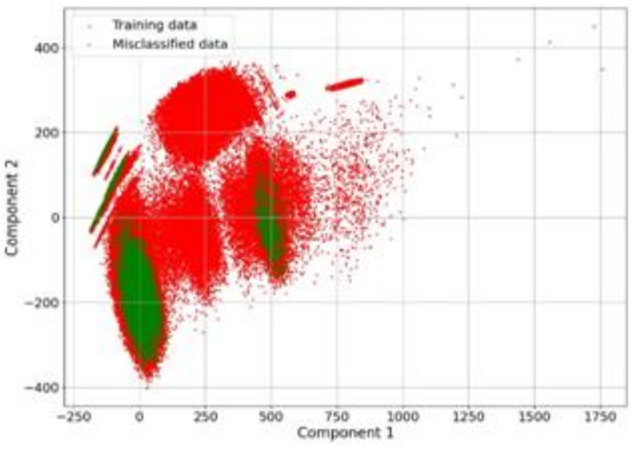
In order to remove dependency on a specific session, the source IP, destination IP, source port, and IP identification (a total of 12 bytes) are excluded from the packet feature. Thus, if the packet size is smaller than *L*+12 bytes, zeros pad the values at the end of the data. In general, packet data create features through one-hot encoding, but there are fields (e.g. length) where it is advantageous for values in the packet data to be processed as numbers. Therefore, one- hot encoding is not applied in this proposed method.

D. *LSTM-BASED CLASSIFIER*

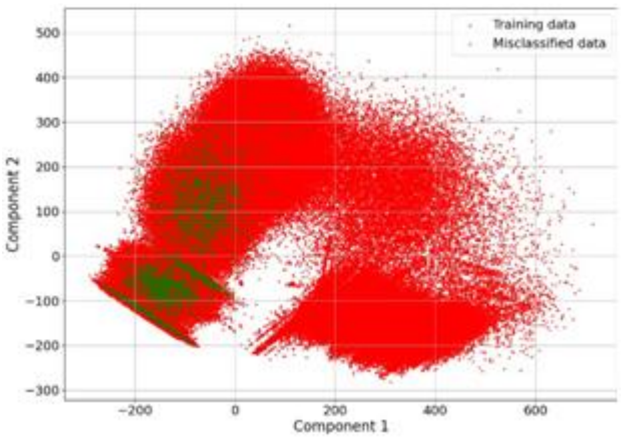
The $t(\leq N)$-th cells of LSTM classify the *t*-th packets. The value for *k* is obtained from the active session cache, as described above. Figure 4 shows the structure of the LSTM classifier when *N*=6 and *L*=98. The output of each LSTM cell has a 512 X 2 dimension as input to the DNN to determine whether a network intrusion occurs. In addition, the output of the cell is stored in the active session cache and serves as a session feature when classifying the next packet. Figure 4 shows the detailed structure of the packet- feature-based classifier.

The (*t*-1)-th cell of the LSTM needs x_{t-1}, h_{t-1}, and C_{t-1} as an input for the cell, and outputs h_t and C_t, which are used as input to the next cell. In order to classify the current packet x_t, only h_t and C_t are required instead of the previous packet, x_{t-1}. Noting that x_{t-1} has a variable size, it is possible to classify x_t by storing the fixed sizes of h_{t-1} and C_{t-1} from the previous classification. This approach has two advantages. First, the classifier can guarantee that it consumes only a fixed size of memory for classification so it can support a high scalability in terms of the concurrent session number.

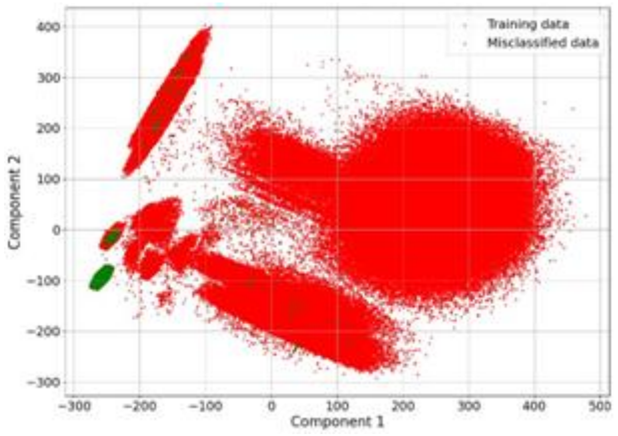
Second, the fixed number of features makes the classifier design easy.



a. ISCX2012



b. CIC2017



c. CSE2018

FIGURE 5. The training dataset (in red) and misclassified data (in green) in the classifier projected on a two-dimensional plane by PCA for each dataset.

E. *THE GAN-BASED ONE-CLASS CLASSIFIER*

Since the result of the LSTM-based classifier determines whether to allow or discard all packets belonging to the session, it is important to verify the reliability of that result. In order to identify and classify packets with a high probability of being false positives, the proposed method uses a GAN. In the GAN for the proposed method with the given training data, the generator creates several sets of fake data similar to the training data, and the discriminator is trained to distinguish real data from the fake data generated. Therefore, the discriminator can be used as a one-class classifier that distinguishes real data. Using these characteristics, a dataset is constructed with data misclassified by a packet-based classifier, and the GAN learns by using it; then, the received packet is classified with a discriminator.

For example, Figure 5 shows the entire training dataset (in red) and misclassified data (in green) in the classifier on a two-dimensional plane through principal component analysis (PCA) [26] for each dataset, ISCX2012, CIC2017, and CSE2018. As shown in each sub-figure, the misclassified data are concentrated in a specific area, and in other cases, they are spread out regardless of the dataset type. In this case, the general GAN is prone to fail during training. Therefore, the proposed method uses the Wasserstein GAN with a gradient penalty (shortly, WGAN- GP) in which the gradient penalty is applied to effectively prevent training failure [27].

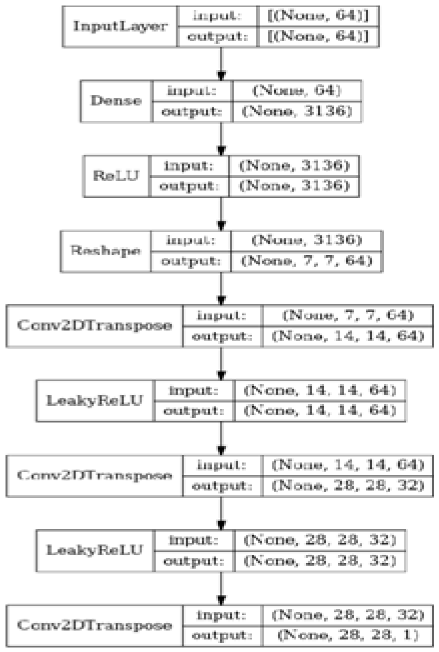

InputLayer | input: [(None, 64)] | output: [(None, 64)]
Dense | input: (None, 64) | output: (None, 3136)
ReLU | input: (None, 3136) | output: (None, 3136)
Reshape | input: (None, 3136) | output: (None, 7, 7, 64)
Conv2DTranspose | input: (None, 7, 7, 64) | output: (None, 14, 14, 64)
LeakyReLU | input: (None, 14, 14, 64) | output: (None, 14, 14, 64)
Conv2DTranspose | input: (None, 14, 14, 64) | output: (None, 28, 28, 32)
LeakyReLU | input: (None, 28, 28, 32) | output: (None, 28, 28, 32)
Conv2DTranspose | input: (None, 28, 28, 32) | output: (None, 28, 28, 1)

FIGURE 6. The WGAN-GP generator structure

Figures 6 and 7 show the model structures of the generator and the discriminator of the WGAN-GP used in the proposed method. All convolution layers in both models use a Leaky ReLU with an alpha value of 0.3 as an activation function and L2 regulation with parameters of 2.5 X 10^{-5} applied [28]. The last convolution transpose layer of the constructor uses a sigmoid activation function, and the dropout of the discriminator is set to 0.3.

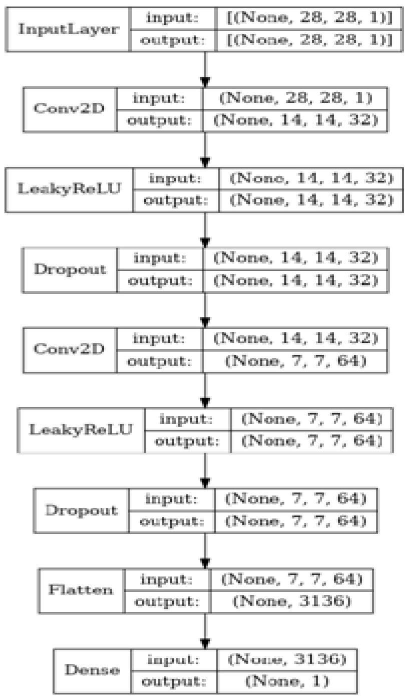

InputLayer | input: [(None, 28, 28, 1)] | output: [(None, 28, 28, 1)]
Conv2D | input: (None, 28, 28, 1) | output: (None, 14, 14, 32)
LeakyReLU | input: (None, 14, 14, 32) | output: (None, 14, 14, 32)
Dropout | input: (None, 14, 14, 32) | output: (None, 14, 14, 32)
Conv2D | input: (None, 14, 14, 32) | output: (None, 7, 7, 64)
LeakyReLU | input: (None, 7, 7, 64) | output: (None, 7, 7, 64)
Dropout | input: (None, 7, 7, 64) | output: (None, 7, 7, 64)
Flatten | input: (None, 7, 7, 64) | output: (None, 3136)
Dense | input: (None, 3136) | output: (None, 1)

FIGURE 7. The WGAN-GP discriminator structure

F. *IMAGE FEATURE GENERATION*

The proposed method uses packet data as features of the LSTM classifier (as described previously). However, if each byte of the packet data is mapped to one feature and applied to the GAN, the GAN does not work properly. This can be explained from three aspects. First, the dimensional size of packet-based features is smaller than images used in deep learning in general. Second, in the image, the correlation between the pixel group composed of adjacent pixels and the class is higher than the correlation between the value of each pixel and the class. On the other hand, in the packet-based feature, the value of a specific feature (for example, the destination port or the total length of the packet) has a very high correlation with a specific class. Finally, the effect of noise on packet-based features is significantly greater than the effect of noise on specific features (i.e. pixel values) of the image. For example, even if the protocol field is changed by 1, the session is treated as a completely different protocol. In order to solve this problem, one-hot encoding can be applied, but this not only makes the number of features too large, but also burdens it by being able to determine whether the packet data features are numerical or categorical. Therefore, in the proposed method, an image is generated by converting existing packet-based features so they have image characteristics by applying a method similar to that of a PAC-GAN [29].

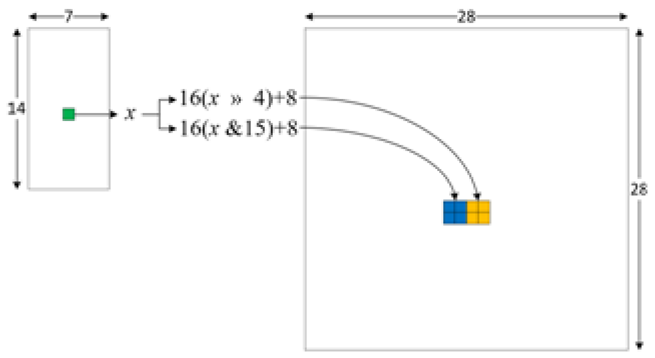


FIGURE 8. Image expansion from the original 7 X 14-dimensional data for the WGAN-GP; '»' and '&' are bitwise right-shift operator and bitwise AND operator, respectively.

The procedure for creating an image is as follows: 98 bytes of packet data used for learning about each packet are first stored in a 7 X 14–byte array. Now, the byte value of each array is divided into the upper four bits and the lower four bits, and each four-bit value, b, is converted into 16 X b+8 (that is, back to an eight-bit value). The transformed values are mapped to 2 X 2 pixels in the image. Finally, the 7 X 14 black-and-white image is expanded to a 28 X 28 X 1 image and is used as training data for the WGAN-GP. For a detailed explanation, see Figure 8.

IV. PERFORMANCE EVALUATION

TABLE II
IDS ALGORITHMS USED FOR PERFORMANCE EVALUATION

Feature classifierstype	Non-deep-learningclassifiers	Deep-learning
Session	Gradientboosting[30], AdaBoostdecisiontree[31]	DNN[32],Tree-CNN[33], 1D-CNN[34]
Packet	-	HAST-IDS

TABLE III
THE DATASETS USED IN THE PERFORMANCE EVALUATION

Name		ISCX2012[35]	CIC2017 [36]	CSE2018 [37]
Classsize		5	11	10
Training dataset	Session	477K	607K	982K
	Packet	15.742K	9,415K	7,685K
Validation dataset	Session	159K	202K	327K
	Packet	5,479K	3,351K	2,750K
Testing dataset	Session	159K	202K	327K
	Packet	5,356K	3,338K	2,750K

To compare accurate performance by the classifiers, detection accuracy and detection speed should be measured. In this study, accuracy was measured for various metrics, including precision, recall, and F1-score, and speed was measured based on the number of packets needed to finally determine if an intrusion was made for each session.

Various datasets also should be used to analyse performance independent of the dataset and network intrusion type. The selected datasets are listed in Table III.

A. *DETECTION SPEED*

The detection speed was compared with session-based classification algorithms, HAST-IDS, and the proposed method. The Gradient boosting, the AdaBoost decision tree, the TSE-IDS, the DNN, the Tree-CNN, and the 1D-CNN under session-based classification methods detect intrusions after the session terminates. Thus, we used the average session length, regardless of the classification algorithm, as the detection speed of the session-based classifiers, and therefore, a lower length means a higher speed.

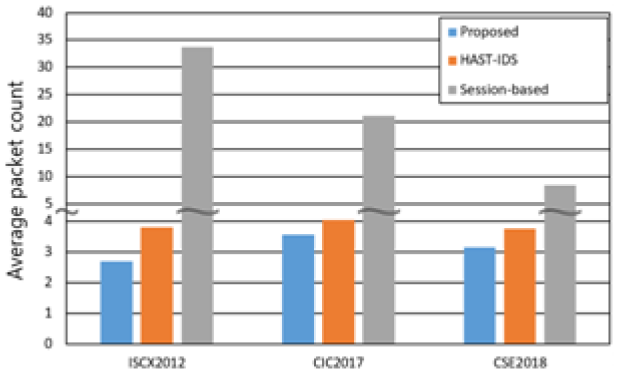


FIGURE 9. Average number of packets required for detection by the algorithms with each dataset

Figure 9 shows the result from comparing the average detection rate (that is, the average value of packet counts upon detection) for the entire test dataset created from each full dataset. As shown in the figure, the proposed method had the fastest intrusion detection speed compared to the existing session-based classification algorithm and HAST-IDS. In particular, it was 12 times faster with ISCX2012 compared with session-based algorithms. In addition, it detected intrusions even faster than HAST-IDS. Since HAST-IDS only sees a fixed number of packets (*N*), it has$_{\text{Name}}$ ISCX2012 [35]CIC2017 [36]CSE2018 [37]an almost constant detection speed regardless of the type of dataset. On the other hand, the proposed method detects anClass size 5 11 10intrusion using up to *N* packets, detecting it more quickly.Training datasetValidation dataset TestingdatasetSession 477K 607K 982K Packet 15.742K 9,415K 7,685KSession 159K 202K 327K Packet 5,479K 3,351K 2,750KSession 159K 202K 327K Packet 5,356K 3,338K 2,750K

Above all, considering that the proposed method has similar or higher intrusion detection accuracy than HAST-IDS, it was confirmed that the method of determining which packet is adequate for the intrusion detection using a GAN is quite effective in improving detection accuracy and speed simultaneously.

In order to accurately evaluate the performance of the proposed method, various existing IDS algorithms were selected and compared. The selected algorithms are listed in Table II.

Now, to compare detection speed in more detail, let us compare the average number of packets for each class in the dataset. Figure 10 shows the results for five classes from the ISCX2012 dataset. The detection speed of the proposed method was the fastest, regardless of the class. In particular, although the session lengths for each class were significantly different, we can see that the proposed method only required approximately the same number of packets, on average, for all classes.

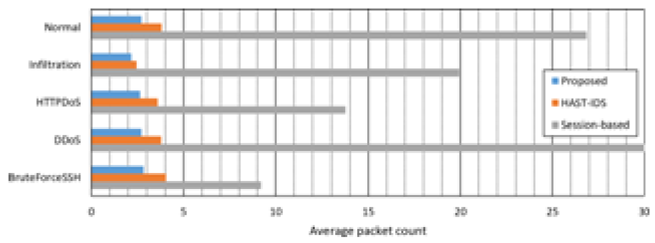


FIGURE 10. Average number of packets required for detection of each class by the algorithms with the ISCX2012 dataset

Figure 11 shows the network intrusion detection rates for 11 classes from the CIC2017 dataset. This also shows characteristics similar to the ISCX2012 dataset. For the PortScan class, all three methods showed the same result because (for most sessions) PortScan consisted of only two packets. On the other hand, for other classes with various session lengths, only the proposed method showed almost constant detection speed. When the average session length was greater than 7, the intrusion could be detected with the almost first four packets, regardless of session length.

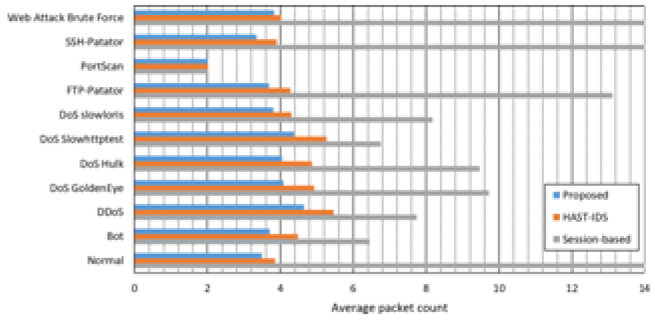


FIGURE 11. Average number of packets required to detect each class by the algorithms with the CIC2017 dataset

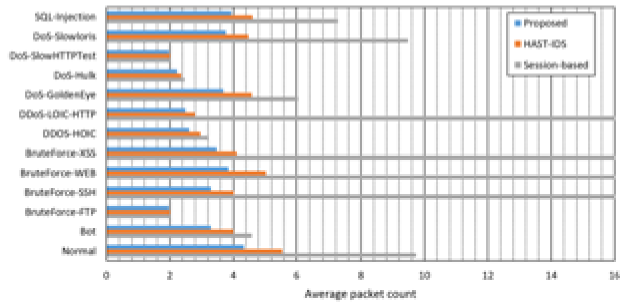


FIGURE 12. Average number of packets required for detection by each class according to detection algorithm in CSE2018 dataset

Figure 12 shows the results for the CSE2018 dataset. Most sessions belonging to DoS-SlowHTTPTest and BruteForce- FTP consisted of two packets, so even using the proposed method, the detection speed cannot be improved. However, for other classes, such as DDoS-LOIC-HTTP and BruteForce-SSH, even if the average session length exceeded 16, intrusion could be determined with only the first three packets. In addition, it was confirmed that performance by the proposed method was higher than HAST-IDS or the existing session-based method for any class in all the datasets.

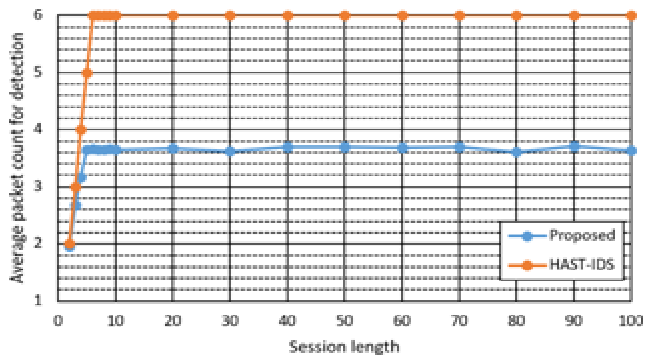


FIGURE 13. Average number of packets required for detection based on average session length with the ISCX2012 dataset

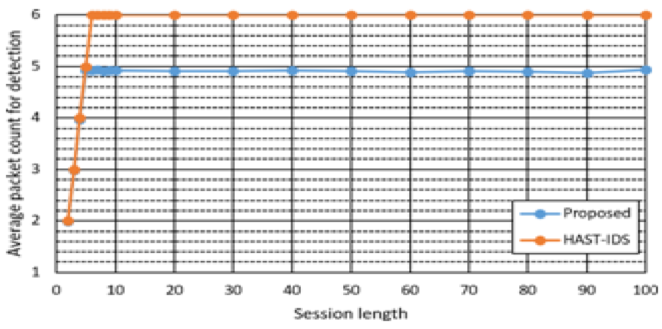


FIGURE 14. Average number of packets required for detection based on average session length with the CIC2017 dataset

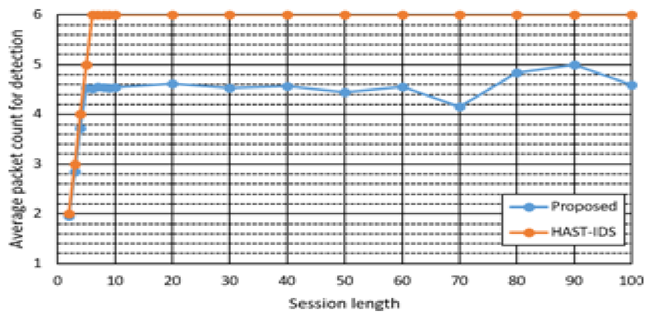
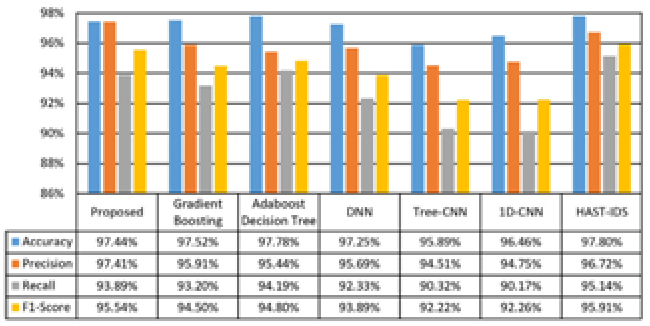


FIGURE 15. Average number of packets required for detection based on average session length with the CSE2018 dataset

The results from measuring the detection speed of the proposed method and HAST-IDS based on session length are shown in Figures 13 to 15. When the session length was less than *N*, the number of packets required for detection increased according to the session length, but when the session length was greater than *N*, HAST-IDS always used *N* packets. Interestingly, the proposed method showed that when the session length was greater than *N*, a constant number of packets less than *N* was always required, regardless of the session length. This proves that the proposed method has high scalability for session lengths.

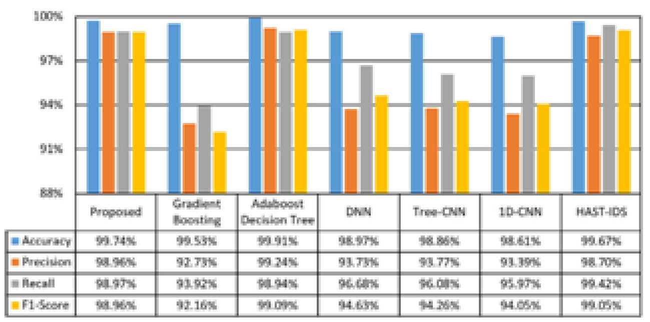
B. ***DETECTION RATE***

To measure the detection accuracy of each algorithm, we compared performance for accuracy, precision, recall, and F1-score metrics. Figures 16 to 18 show the performance with the three datasets. For each dataset, the proposed algorithm shows very high F1-scores regardless of the dataset type but session-based algorithms and HAST-IDS show fluctuating results according to the dataset type. For example, Adaboost decision tree, one of the best session- based classifiers showing the highest F1-score, achieves 0.74% lower but 0.13% higher F1-scores with ISCX2012 and CIC2017 compared to the proposed algorithm. HAST- IDS shows 0.37% higher but 8.9% lower F1-scores with ISCX2012 and CSE2018, respectively.



	Proposed	Gradient Boosting	Adaboost Decision Tree	DNN	Tree-CNN	1D-CNN	HAST-IDS
■ Accuracy	97.44%	97.52%	97.78%	97.25%	95.89%	96.46%	97.80%
■ Precision	97.41%	95.91%	95.44%	95.69%	94.51%	94.75%	96.72%
■ Recall	93.89%	93.20%	94.19%	92.33%	90.32%	90.17%	95.14%
■ F1-Score	95.54%	94.50%	94.80%	93.89%	92.22%	92.26%	95.91%

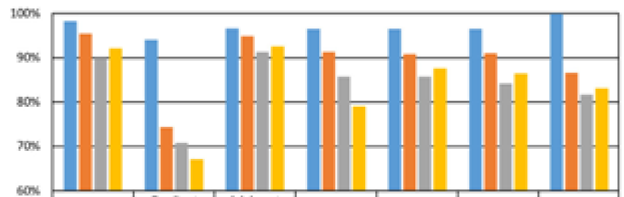
FIGURE 16. Detection performance by the algorithms with the ISCX2012 dataset



	Proposed	Gradient Boosting	Adaboost Decision Tree	DNN	Tree-CNN	1D-CNN	HAST-IDS
■ Accuracy	99.74%	99.53%	99.91%	98.97%	98.86%	98.61%	99.67%
■ Precision	98.96%	92.73%	99.24%	93.73%	93.77%	93.39%	98.70%
■ Recall	98.97%	93.92%	98.94%	96.68%	96.08%	95.97%	99.42%
■ F1-Score	98.96%	92.16%	99.09%	94.63%	94.26%	94.05%	99.05%

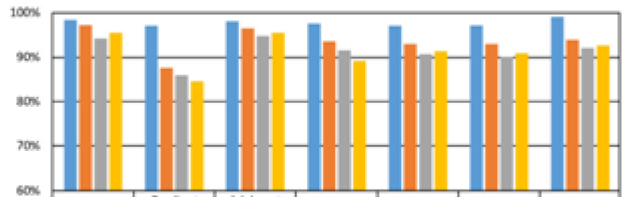
FIGURE 17. Detection performance by the algorithms with the CIC2017 dataset

To compare overall performance, let us show average detection rate for each dataset in Figure 19. As confirmed by this figure, the proposed algorithm showed the highest accuracy, precision, and F1-score. Only exception is recall and it achieved the second highest value after Adaboost decision tree. From Figures 16 to 19, it proves that the proposed algorithm has an ability to guarantee that high classification accuracy regardless of dataset type. It is a strong and crucial merit that NIDS algorithm should support. As mentioned above, compared to the session- based method or HAST-IDS, the proposed method can detect a network intrusion as soon as possible, and the intrusion detection rate showed higher results compared to the other two methods, which is a big advantage. This means it can reliably provide a high detection rate and fast detection speeds in various environments.



	Proposed	Gradient Boosting	Adaboost Decision Tree	DNN	Tree-CNN	1D-CNN	HAST-IDS
🟦 Accuracy	98.26%	94.09%	96.62%	96.57%	96.58%	96.53%	99.92%
🟧 Precision	95.43%	74.37%	94.93%	91.30%	90.86%	90.98%	86.61%
⬜ Recall	89.88%	70.75%	91.31%	85.77%	85.73%	84.16%	81.68%
🟨 F1-Score	92.16%	67.08%	92.57%	79.04%	87.61%	86.40%	83.06%

FIGURE 18. Detection performance by the algorithms with the CSE2018 dataset



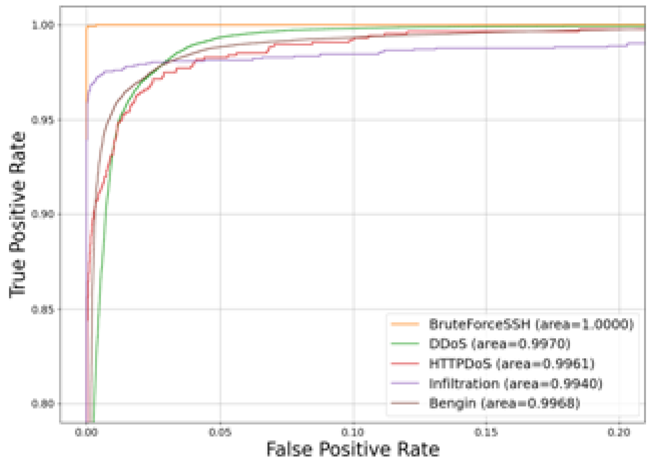
	Proposed	Gradient Boosting	Adaboost Decision Tree	DNN	Tree-CNN	1D-CNN	HAST-IDS
🟦 Accuracy	98.48%	97.05%	98.10%	97.60%	97.11%	97.20%	99.13%
🟧 Precision	97.27%	87.67%	96.54%	93.57%	93.05%	93.04%	94.01%
⬜ Recall	94.25%	85.96%	94.81%	91.59%	90.71%	90.10%	92.08%
🟨 F1-Score	95.55%	84.58%	95.49%	89.18%	91.36%	90.90%	92.68%

FIGURE 19. Average detection performance by the algorithms

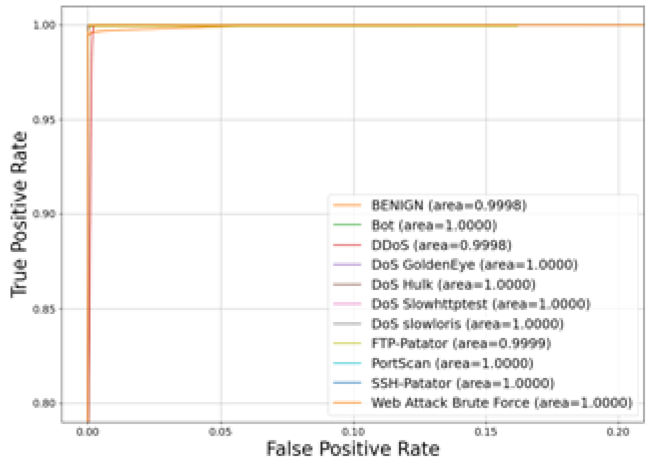
Figure 20 shows the ROC for each class with each dataset. This also confirmed a high detection rate for almost all classes without significant deviations.

V. CONCLUSION

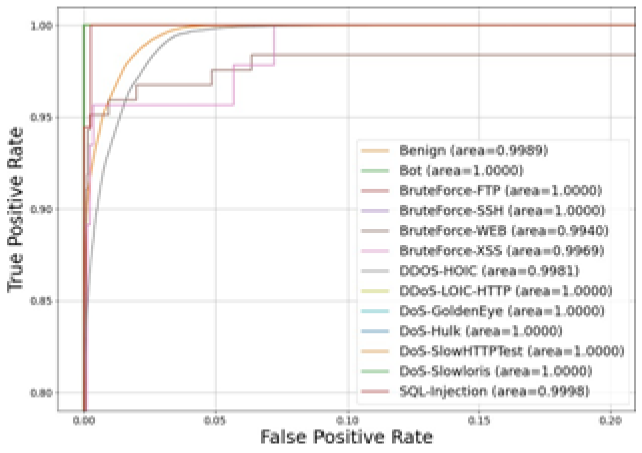
Unlike the existing methods, the proposed NIDS has the advantage of being able to stop malicious users before they cause damage to the network, because it can determine whether an intrusion is occurring before the session terminates. In addition, using the packet feature to reduce unnecessary time and memory to statistically analyze and calculate packet data to generate the session feature is a great advantage. This is important considering that network intrusions are becoming more diverse while, at the same time, the number and quantity of sessions are increasing significantly. Basically, machine learning models require a lot of memory and computational power, so powerful and expensive hardware is absolutely necessary. Therefore, it is important in practical terms for the requirements in hardware to be lower than with the existing methods. An ML-based IDS should ultimately evolve into a real-time IPS that can detect and defend network intrusions without delay. Since the hardware requirements of an IPS are much higher than those of an IDS, the proposed method suggests a useful direction for real-time IPS development in terms of a lightweight design.



a. ISCX2012



b. CIC2017



c. CSE2018

FIGURE 20. ROC for each class in each dataset

Although the detection speed of the proposed approach is fast, accuracy needs to be further improved, compared to the conventional approaches. Since the proposed structure does not depend on a specific classifier, any higher accuracy ML model can be used, and it can improve detection performance easily without sacrificing the advantages of the proposed method. Through this, it is expected that more secure and faster network services can be provided to users.

REFERENCES

[1] C. Seelammal, K.V. Devi (2016) Computational intelligence in intrusion detection system for snort log using hadoop, in: 2016 International Conference on Control, Instrumentation, Communication and Computational Technologies, ICCICCT, 2016, pp. 642–647, http://dx.doi.org/10.1109/ICCICCT.2016.7988029.

[2] L. Bilge, T. Dumitras, (2012) Before we knew it: an empirical study of zero-day attacks in the real world, in: T. Yu, G. Danezis, V.D. Gligor (Eds.), The ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16- 18, 2012, ACM, 2012, pp. 833–844, http: //dx.doi.org/10.1145/ 2382196.2382284.

[3] M. Al-Qatf, Y. Lasheng, M. Al-Habib, K. Al-Sabahi (2018) Deep learning approach combining sparse autoencoder with SVM for network intrusion detection, IEEE Access 6, pp. 52843–52856, http://dx.doi.org/10.1109/ACCESS.2018. 2869577.

[4] I. Ahmad, M. Basheri, M.J. Iqbal, A. Rahim (2018) Performance comparison of support vector machine, random forest, and extreme learning machine for intrusion detection, IEEE Access 6, pp. 33789– 33795, http://dx.doi.org/ 10.1109/ACCESS.2018.2841987.

[5] M. Belouch, S.E. hadaj (2017) Comparison of ensemble learning methods applied to network intrusion detection, Proceedings of the Second International Conference on Internet of Things, Data and Cloud Computing, in: ICC, vol. 17, Association for Computing Machinery, New York, NY, USA, pp. 1–4, http://dx.doi.org/10.1145/ 3018896.3065830.

[6] Xilinx (2020) Accolade Technology, IPS/IDS offload, https://www.xilinx.com/products/acceleration-solutions/1-1bkvll1. html. Accessed 20 July 2022.

[7] NVIDIA (2022) NVIDIA Bluefield Data Processing Units, https://www.nvidia.com/en-us/networking/products/data-processing- unit, Accessed 20 July 2022.

[8] Yifan Sun (2019) Summarizing CPU and GPU Design Trends with Product Data, https://deepai.org/publication/summarizing-cpu-and- gpu-design-trends-with-product-data, Accessed 20 July 2022.

[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio (2014) Generative Adversarial Nets. Proceedings of the International Conference on Neural Information Processing Systems (NIPS 2014). pp. 2672–2680.

[10] Sepp Hochreiter, Jürgen Schmidhuber (1997) Long short-term memory. Neural Computation. 9 (8): pp. 1735–1780. doi:10.1162/neco.1997.9.8.1735. PMID 9377276. S2CID 1915014.

[11] N. Almusallam, Z. Tari, J. Chan, A. Fahad, A. Alabdulatif, M. Al- Naeem (2021) Towards an Unsupervised Feature Selection Method for Effective Dynamic Features, IEEE Access, vol. 9, pp. 77149-77163, 2021, doi: 10.1109/ACCESS.2021.3082755.

[12] A. Nugroho, A. Z. Fanani, G. F. Shidik (2021) Evaluation of Feature Selection Using Wrapper For Numeric Dataset With Random Forest Algorithm, 2021 International Seminar on Application for Technology of Information and Communication (iSemantic), pp. 179-183, doi: 10.1109/iSemantic52711.2021.9573249.

[13] X. Peng, W. Huang and Z. Shi (2019) Adversarial Attack Against DoS Intrusion Detection: An Improved Boundary-Based Method, 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI), pp. 1288–1295, doi:10.1109/ICTAI.2019.00179.

[14] Z. Wang (2018), Deep Learning-Based Intrusion Detection with Adversaries, IEEE Access, vol. 6, pp. 38367–38384, doi:10.1109/ACCESS.2018.2854599.

[15] M.A. Ferrag, L. Maglaras, S. Moschoyiannis, H. Janicke (2020) Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study, J. Inf. Secur. Appl. 50, 102419, http://dx.doi.org/10.1016/j.jisa.2019.102419.

[16] R.V. Mendonça, A.A.M. Teodoro, R.L. Rosa, M. Saadi, D.C. Melgarejo, P.H.J. Nardelli, D.Z. Rodríguez (2021)

Intrusion detection system based on fast hierarchical deep convolutional neural network, IEEE Access 9, pp. 61024–61034, http://dx.doi.org/10.1109/ACCESS.2021.3074664.

[17] C. Liu, Z. Gu, J. Wang (2021) A hybrid intrusion detection system based on scalable K-Means+ random forest and deep learning, IEEE Access 9, pp. 75729–75740, http://dx.doi.org/10.1109/ACCESS.2021.3082147.

[18] P.-F. Marteau (2021) Random partitioning forest for point-wise and collective anomaly detection—Application to network intrusion detection, IEEE Trans. Inf. Forensics Secur. 16, pp. 2157–2172, http://dx.doi.org/10.1109/TIFS.2021.3050605.

[19] J. Kevric, S. Jukic, A. Subasi (2017) An effective combining classifier approach using tree algorithms for network intrusion detection, Neural Comput. Appl. 28, http://dx.doi.org/10.1007/s00521-016-2418-1.

[20] H. Jia, J. Liu, M. Zhang, X. He, W. Sun (2021) Network intrusion detection based on IE-DBN model, Comput. Commun. 178, pp. 131– 140, http://dx.doi.org/10.1016/j.comcom.2021.07.016.

[21] K. Narayana Rao, K. Venkata Rao, P.R. P.V.G.D (2021) A hybrid intrusion detection system based on sparse autoencoder and deep neural network, Comput. Commun. 180, pp. 77–88, http://dx.doi.org/10.1016/j.comcom.2021.08.026.

[22] N. Kunhare, R. Tiwari (2018) Study of the attributes using four class labels on KDD99 and NSL-KDD datasets with machine learning techniques, 2018 8th International Conference on Communication Systems and Network Technologies, CSNT, 2018, pp. 127–131, http://dx.doi.org/10.1109/CSNT.2018.8820244.

[23] J. Song, H. Takakura, Y. Okabe, M. Eto, D. Inoue, K. Nakao (2011) Statistical analysis of honeypot data and building of kyoto 2006+ dataset for NIDS evaluation, in: Proceedings of the First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security, in: BADGERS, vol.11, Association for Computing Machinery, New York, NY, USA, pp. 29–36, http://dx.doi.org/10.1145/1978672.1978676.

[24] W. Wang, Y. Sheng, J. Wang, X. Zeng, X. Ye, Y. Huang, M. Zhu (2018) HAST-IDS:learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection, IEEE Access 6, pp. 1792–1806, http://dx.doi.org/10.1109/ACCESS.2017. 2780250.

[25] [V.S.M. Srinivasavarma, S.R. Pydi, S.N. Mahammad (2022) Hardware-based multi-match packet classification in NIDS: an overview and novel extensions for improving the energy efficiency of TCAM-based classifiers. J Supercomput 78, pp. 13086–13121. https://doi.org/10.1007/s11227-022-04377-8

[26] Pearson, K. (1901) On Lines and Planes of Closest Fit to Systems of Points in Space, Philosophical Magazine. 2 (11): 559–572. doi:10.1080/14786440109462720.

[27] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, Aaron Courville (2017) Improved training of wasserstein GANs. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17). Curran Associates Inc., Red Hook, NY, USA, pp. 5769–5779.

[28] Andrew L. Maas, Awni Y. Hannun, Andrew Y. Ng (2014) Rectifier Nonlinearities Improve Neural Network Acoustic Models

[29] Adriel Cheng (2019) PAC-GAN: Packet Generation of Network Traffic using Generative Adversarial Networks, pp. 0728-0734, https:/doi.org/10.1109/IEMCON.2019.8936224.

[30] Jerome H. Friedman (2001) Greedy Function Approximation: A Gradient Boosting Machine, The Annals of Statistics, vol. 29, no. 5, 2001, pp. 1189–232. JSTOR.

[31] Trevor Hastie, Saharon Rosset, Ji Zhu, Hui Zou (2009) Multi-class AdaBoost, Statistics and Its Interface. 2 (3): pp. 349–360. https://doi.org/10.4310/sii.2009.v2.n3.a8.

[32] Yoshua Bengio (2009) Learning Deep Architectures for AI, Foundations and Trends in Machine Learning. 2 (1): pp. 1–127. https://doi.org/10.1561/2200000006