



Article

Packet Preprocessing in CNN-Based Network Intrusion Detection System

Wooyeon Jo ¹, Sungjin Kim ¹, Changhoon Lee ² and Taeshik Shon ^{1,*}

¹ Department of Computer Engineering, Ajou University, Suwon 16499, Korea; dndusdndus12@ajou.ac.kr (W.J.); ksjskyblue@ajou.ac.kr (S.K.)

² Department of Computer Engineering, SNUT, Seoul 80523, Korea; chlee@seoultech.ac.kr

* Correspondence: tsshon@ajou.ac.kr; Tel.: +82-31-219-3321

Received: 5 June 2020; Accepted: 14 July 2020; Published: 16 July 2020



Abstract: The proliferation of various connected platforms, including Internet of things, industrial control systems (ICSs), connected cars, and in-vehicle networks, has resulted in the simultaneous use of multiple protocols and devices. Chaotic situations caused by the usage of different protocols and various types of devices, such as heterogeneous networks, implemented differently by vendors renders the adoption of a flexible security solution difficult, such as recent deep learning-based intrusion detection system (IDS) studies. These studies optimized the deep learning model for their environment to improve performance, but the basic principle of the deep learning model used was not changed, so this can be called a next-generation IDS with a model that has little or no requirements. Some studies proposed IDS based on unsupervised learning technology that does not require labeled data. However, not using available assets, such as network packet data, is a waste of resources. If the security solution considers the role and importance of the devices constituting the network and the security area of the protocol standard by experts, the assets can be well used, but it will no longer be flexible. Most deep learning model-based IDS studies used recurrent neural network (RNN), which is a supervised learning model, because the characteristics of the RNN model, especially when the long-short term memory (LSTM) is incorporated, are better configured to reflect the flow of the packet data stream over time, and thus perform better than other supervised learning models such as convolutional neural network (CNN). However, if the input data induce the CNN's kernel to sufficiently reflect the network characteristics through proper preprocessing, it could perform better than other deep learning models in the network IDS. Hence, we propose the first preprocessing method, called “direct”, for network IDS that can use the characteristics of the kernel by using the minimum protocol information, field size, and offset. In addition to direct, we propose two more preprocessing techniques called “weighted” and “compressed”. Each requires additional network information; therefore, direct conversion was compared with related studies. Including direct, the proposed preprocessing methods are based on field-to-pixel philosophy, which can reflect the advantages of CNN by extracting the convolutional features of each pixel. Direct is the most intuitive method of applying field-to-pixel conversion to reflect an image's convolutional characteristics in the CNN. Weighted and compressed are conversion methods used to evaluate the direct method. Consequently, the IDS constructed using a CNN with the proposed direct preprocessing method demonstrated meaningful performance in the NSL-KDD dataset.

Keywords: IoT; deep learning; packet preprocessing; intrusion detection system; industrial control system; vehicle; artificial neural networks; data preprocessing

1. Introduction

Since the advent of the Internet of things (IoT), various platforms have been proposed. For example, Industrial Control System (ICSs) are gradually evolving into intelligent and autonomous systems. In the ICS field, the concept of Industrial IoT (IIoT) was proposed based on IoT. The IIoT communicates with various sensors and manages critical infrastructure and has a larger network scale than existing systems. For Industry 4.0, Germany proposed a smart environment to accelerate the development of ICSs; currently, concepts such as smart cities and smart factories are becoming more specific and realistic. Wireless Sensor Networks (WSNs) are sometimes treated as part of the IoT as a mesh network. Regardless of whether or not WSNs belong to IoT, they are treated as part of the IoT because various sensor devices are combined within WSNs to construct a network. The ecosystem in which networks are built with IoT devices around Artificial Intelligence (AI) speakers also has similar characteristics. AI speaker ecosystems are also being studied for security technologies such as digital forensics [1]. The features of many recently introduced devices, such as the strong connectivity, require flexible preprocessing techniques. Other studies demonstrated that a correct preprocessing method is required to apply intrusion detection systems (IDSs) to communication data in a WSN [2]. Software-defined networks (SDNs) continue to be developed; as a connecting platform, a network constructed using applications generates a variety of communication data due to the communication between applications. A network attack detection study was conducted in a data-driven SDN [3]. Automotive networks are transitioning from mixing Ethernet and legacy protocols. Therefore, autonomous vehicles should use learning-based applications such as convolutional neural networks (CNNs) through processing units such as field programmable gate array (FPGA) to apply novel detection technologies [4]. FPGAs, which are at least 10 times more energy efficient than graphics processing units (GPUs), are widely used for object recognition in hard conditions such as advanced driver assistance system [5].

In the past, only critical infrastructure such as the ICS required heterogeneous communication. However, since the advent of the IoT, networks requiring massive connectivity, such as IIoT and Smart Factory, have increased. Now, it is generally accepted that various types of devices communicate with each other in common network environments as well as in ICS environments. This communication scheme is expected to increase in the near future. The increasing number and type of connected devices are advantageous to automated systems, which has allowed massive connected heterogeneous networks to be created. The heterogeneity and connectivity, characteristics of IoT and ICS networks composed of various devices, provide convenience and allow different protocols to be used simultaneously. The constraints due to the use of different protocols have posed challenges for developing, applying, and updating security solutions. A typical security solution that aims at only a single protocol will not be able to provide macroscopic security. If multiple applications are applied, heterogeneous redundancy is difficult to maintain, thus resulting in vulnerabilities. The heterogeneity of ICS networks is an essential feature that should be considered when building a flexible security solution [6].

The ICS was initially operated in a closed network and required only basic level security. However, security requirements are increasing with security threats. An example of a security requirement enforcement is that in 2018, the North American Electric Reliability Corporation fined Duke Energy USD \$10 million for a cybersecurity failure related to critical infrastructure protection [7]. Another example is a hacking group called Xenotime, formerly known as TRISIS malware, that targeted ICSs [8], which again attempted to attack the US grid in June 2019 [9]. Security threats are not the only risks facing ICSs. If a staff's premature operation can cause danger, an alert should be sent as an anomaly. Cases of manual shutdown of nuclear reactors have been reported [10]. In these incidents, the situation was identified and a manual shutdown was performed due to abnormal temperature. However, the system should have found the command that caused the error to be an anomaly.

There are various other IoT platforms such as consumer IoT and commercial IoT, and all IoT platforms are based on wireless communication. However, compared to the rapidly developing wireless communication technology level from 4G to 5G and 6G, the security of IoT devices has not kept pace. The main differences between traditional IT and IoT are the increased complexity and connectivity.

For example, massive machine-type communications (mMTC) means that the heterogeneity of the current IoT network will become increasingly complex. In the communication method, the orthogonal frequency division multiplexing (OFDM) used in 4G and the non-orthogonal multiple access (NOMA) proposed to satisfy the performance requirements of 5G have increased connectivity and will continue to do so in the future. Ideally, each IoT device should have optimized security solutions, but the vendor's inadequate technical expertise and insufficient performance of each device is not enough to provide security. To overcome the above limitations, a suitable method to apply security to IoT is the use of a router or cloud that is the center of distributed devices. Commercial IoT security products are sold with security solutions installed on router devices, and studies showed that the devices with the highest quality links will be appropriate for security when IoT is applied [11]. Routers and clusters are examples of these; since they communicate with many IoT and even IIoT devices [12], they require flexible security technologies.

We define a flexible security solution as a universally applicable security solution based on minimum requirements. The minimum requirements mean basic information that can be easily provided or obtained, such as a communication protocol. Examples of basic information are those that do not require a security expertise, such as size and offset information of protocol fields defined by the standard. If the communication-protocol-related items defined by the communication standard are not properly followed, communication between products will not work and the communication will operate poorly. Therefore, minimum requirements are the most accessible and basic information with which all devices in the network must comply. However, security standards that are defined and regularly updated by many researchers and organizations, such as international standards organizations, NERC-CIP, and CERT, are usually not properly maintained [13,14]. As security-related matters are difficult to confirm until an accident occurs, this is more fatal in a massive-connected network environment such as the IoT and ICS. In other words, an attacker can adversely affect the network even if only one vulnerable attack point is secured [15]. We defined the flexible IDS conditions for connected/heterogeneous networks as follows: if a solution needs a security expertise to apply a security to network, it is not flexible. However, if the characteristics of the network are not properly reflected, the performance of the security solution deteriorates. As a countermeasure, many security technologies have been proposed, from simple rule/signature-based whitelist/blacklist to anomaly detection (AD) or IDSs. However, applying the proposed novel security technologies to ICS is challenging [16]. Unlike general network environment, the IoT, ICSs, and even in-vehicle networks (IVNs) must have security techniques that comply with rules such as hard-real time, reliability, and availability.

To overcome these challenges faced by IDS, we propose preprocessing methods for CNN-based IDSs using only minimum communication protocol information, and size and offset of the fields. The remainder of this paper is structured as follows: Related studies are analyzed in Section 2. Studies using NNs for network anomaly detection have constructed several models that are appropriate in terms of performance such as CNN, recurrent neural network (RNN), long short-term memory (LSTM), and autoencoder. Section 3 outlines the process of selecting a deep learning model that can reflect the characteristics of the IoT environment and the preprocessing processes suitable for the model. In Section 4, the performance of different methods is compared through various studies to demonstrate the efficacy of the proposed methodology. The experiment used NSL-KDD [17], which is a network packet dataset containing an attack. Finally, discussions and conclusions are provided in Section 5.

2. Materials and Methods

Recently, anomaly detection technology was developed due to the introduction of deep learning and neural network (NN) concepts. These developments have been actively studied in ICS networks [18,19]. More research is being conducted on the IoT-related networks, but cases using supervised learning models such as CNN are much less effective than hybrid (including semi-supervised) models or unsupervised models because the supervised models require difficult resources such as accurate labels

and anomalous data. However, NNs studied with sufficient data resources to provide, such as labels and balance, generally have the highest accuracy when supervised [20]. CNN has rarely been used in IDS among supervised models in a few cases. Some cases used CNN for feature extraction or autoencoder encoding rather than as the main classifier because CNN is excellent for outlier detection and spatiotemporal feature extraction from data [21]. Some studies built IDSs mainly using CNN, but the resulting performance was worse than in other studies [22]. To effectively optimize a CNN for IDS, the input data must be designed to sufficiently reflect the importance of the kernel. Since CNNs work effectively on convolutional features like images, applying network packets as the data stream does not have much significance.

IDS-targeting networks, such as IoT, IVN, and ICS, are also called network intrusion detection systems (NIDSs) [23]. Proven datasets that contain attacks such as NSL-KDD and research to detect them have contributed to NIDS research [24]. Initially, the KDD Cup 99 dataset was used in IDS technology proposals [25]; however, the performance results are less reliable than those of the improved NSL-KDD version. Currently, IDSs based on various machine-learning techniques compare performance using the NSL-KDD dataset. When the NSL-KDD dataset was proposed, multiple detection methods were used for its testing. The results indicated that support vector machine (SVM), with the lowest accuracy, was the most robust to the dataset change from KDD Test to Test + as a result of the biased dispersion of the dataset [26]. The results of this study indicated that the experimental results should be analyzed in depth. A deep learning approach for IDSs was studied recently, but the applied model is still biased. A method using an autoencoder has been widely used in IDSs from simple applications to an application reflecting the characteristics of the network [18,19]. In RNN, LSTM is often used, which solves the vanishing problem and improves the LSTM performance for network traffic [27,28]. However, although CNNs are widely used for deep learning, they are rarely used in IDSs [29]. The performance has been improved by combining different models to utilize each of the deep-running models [30]. However, using multiple deep learning models is difficult in the IoT where real-time is more important than in any other network. The biased trend of selecting a deep-running model indirectly indicates the appropriate model for the network. However, the deep learning model used in IDSs, which targets typical networks, cannot be equally applied to next-generation networks such as IIoT and IVN. Because they have unique characteristics that distinguish them from general networks, the appropriate model must be selected.

Some IDS research has detected based on binaries. Malware has a specific execution structure because it is an executable binary file. Additionally, the header, body, and footer of malware can be distinguished by its specific structural characteristics. Malware classification is associated with IDSs because using specific structures and patterns in binary executables is similar to using specific fields and values in packets. However, comparing a particular structure to a field is highly general. Therefore, studies using CNN, which is universally applicable, should be performed. A CNN-based malware classifier uses images to visualize and distinguish malware [31]. The specific structure of malware can be used to increase the detection rate [32]. However, relying on the variable areas of malware can seriously damage a system. If the same attacking malware has a different structure through the development of different methods, a CNN-based IDS will classify them as different malware and even classify them as normal. However, ICS and IVN protocols can expect high use because the size of the fields is mostly fixed.

3. Methodology

Protocols used in actual IoT networks are implemented according to standards; however, the communication software differs from one another depending on the implementation methods. The variety of available communication software makes it difficult to apply predefined security methods. Building a secure application considering heterogeneity and variety requires a learning-based application. Learning-based security applications should be based on appropriate learning models. In this study, we implemented learning-based security applications that can consider heterogeneity and variety.

In this study, the CNN was selected for the following reasons: autoencoders, RNNs, and many other add-on models, such as LSTM, can learn large amounts of data over a long period of time. However, they cannot handle multiple protocols simultaneously. Various functions exist within a single protocol but are rarely used in most cases. These characteristics of IoTs can be misleading when constructing IDSs based on deep learning, and errors such as misunderstanding rarely used behaviors as abnormal behaviors can easily occur. Even a CNN model does not have the specific strength to handle a normal behavior that rarely occurs; such behavior is expected to be solved by combining the CNN with the proposed preprocessing method. The CNN, i.e., a NN that can be learned by converting an image or a binary enables re-scaling a specific feature in the preprocessing. This means that a CNN's input is regarded as binary with a specific pattern, rather than an image, when performing artificial preprocessing. In the IoT environment, various types of devices are used and, accordingly, the protocols used in a single station are subdivided into several levels. Not only do the position and size of the field vary depending on the protocol, variable length within the same protocol is also common. Therefore, the preprocessing method proposed herein is an independent method that depends less on the protocol.

The preprocessing techniques are divided into three parts, but their basic frameworks are the same. The proposed preprocessing method minimizes the field dependency. Consequently, we derived a scheme that can distinguish each field but is independent of the subdivided meaning of the field. However, even if each field is preprocessed without distinction, the resulting model should be able to detect abnormal behaviors. The preprocessing scheme converts a packet data to an image, and the pixels of the preprocessed image are designed so that they can fully affect the NN of each field. Each field converted to an image occupies one pixel, and the pixels are converted to a convolutional layer through a pooling layer. At this time, each pixel of the convolutional layer represents the intersection of the fields. As the field is represented by one pixel, the convolutional layer denotes a combination of various fields.

The NSL-KDD dataset used in this study has 41 fields as packet data in a csv file, and they are preprocessed according to predefined rules such as the csv file. There are two types of fields, and they are converted into two-dimensional images as follows:

1. Continuous: A field having a real value such as a "src/dst bytes". This field type measures the range of possible values. The maximum and minimum values of the range are normalized into the range 0–255 because a grayscale two-dimensional image is used, which may be different if there are multiple values in one pixel, such as RGB.
2. Symbolic: A field having a specific string value such as a "service" or "protocol type". This also applies to "src/dst IP", which is a target for building access lists used in security solutions such as whitelist. This field type measures the total number of possible symbols and assigns pixel values in the range 0–255. For example, if the target field can have three kinds of symbolic values, the field values will be normalized as 0, 128, and 255.

This leads to symbolic variables having the maximum gap from each other. Each of the fields clearly separated through the gap automatically derives a feature map similar to building an access list through the convolutional kernel.

Figure 1 shows the method to extract 41 fields from a single packet data and convert them back into 28×28 images. On the left, 41 fields are pretreated with 28×28 -sized images, and its initial 5×5 area is expanded. The enlarged 5×5 image area shows the field that will be preprocessed for each pixel. An example of a 5×5 kernel with highlighted areas is shown on the right of the figure. These highlighted areas are also displayed in the enlarged area, representing a combination of these fields [F1, F3, F4, F5, F6, F17, F30, F31, F32, F33, F34] that will affect learning. This conversion method is "direct" and is converted into the rule of one field to one pixel. If the CNN has a 5×5 kernel as shown in Figure 1, the following convolved feature will be reflected in the following fields, where the number after F (field) is the order of each field. Each pixel (field) will have one numeric value of 0–255 when converted to a grayscale image.

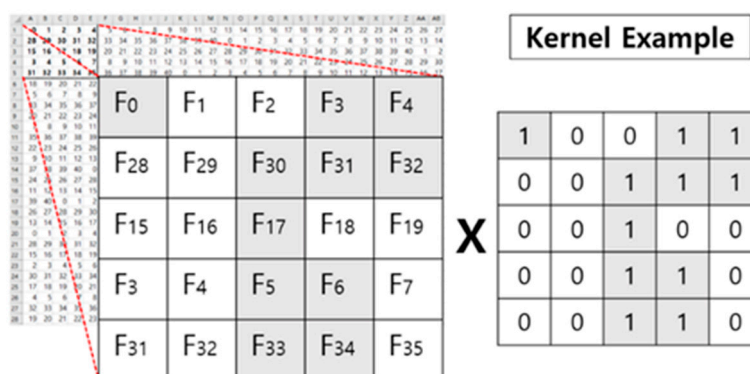


Figure 1. Proposed preprocessing methodology.

Subsequently, the pool size and stride allow the CNN to automatically consider almost any combination of fields. The field can be set to various values such as general IP address, port number, protocol type, and byte. The method shown above is a direct conversion, and two other preprocessing methods are proposed. Through the proposed three preprocessing methods, various results can be obtained even if only one CNN model is used. The independence of the preprocessing method and the deep running enables splitting of the tasks. Splitting tasks is more effective in distributed network such as ICSs and IVNs. Attaching security devices to specific routers or gateways for IDSs may be subject to restrictions when there are one or more stations. The devices afford flexibility to the IDS because they can change their performance by updating only the preprocessing method, except for the deep learning model. This method can be considered as the one that can be excluded by fewer operations through model modifications, such as modifying the kernel map.

The protocol field in the pixel preprocessing method can be divided into three parts. The typical parts of the preprocessing method are described in the direct conversion method, and the subsequent methods focus on the differences.

3.1. Direct Conversion

Direct conversion is the most basic preprocessing method for the protocol field to preprocess pixels, as shown in Figure 2a. After separating the packet data into fields, each field is converted into an 8-bit vector. Subsequently, the input data are sequentially inserted into a pixel of an appropriate size. This normalization by category generally yields interlaced images because the number of fields is smaller than the number of pixels contained in an image. Each pixel in the image represents one field. The method of converting a field to a pixel depends on the value of each field. It also requires prior knowledge such as the range of values possible in each field of the protocol.

3.2. Weighted Conversion

Weighted conversion is a weighting method that forces the kernel to select a few types of fields redundantly when the convolutional layer is created. Weighting can be applied from a simplified vertical or horizontal method, as shown in Figure 2b, or by applying only a specific field redundantly. The final shape shown in Figure 2b is an example of dividing an image into two regions of 28×14 and vertically enlarging the pixels. When weighing to increase the number of pixels, the field convolved by the 5×5 kernel is limited to a maximum of 10 fields. The example above is for extreme pixel increments; in normal cases, it would be better to increment them slightly depending on the results.

The CNN API provided by most deep learning platforms, including TensorFlow, automatically provides the user with randomly generated kernels. Even if the kernel can be manually entered, overfitting may occur due to the lack of diversity through randomness. To overcome the development constraints of the deep running development, weighted conversion is proposed as the preprocessing method.

In the case of NSL-KDD with 41 features, duplicate fields can be listed because they are not divided into 28 lines, which is the number of pixels in one line. To prevent overfitting by a specific field, padding may be applied.

3.3. Compressed Conversion

The two preprocessing methods introduced thus far convert a single packet into one image. However, the compressed conversion method converts multiple packets into one image, as shown in Figure 2c. The number of packets can be divided using one of two methods: naïve and adjective. The naïve method binds multiple packets based on protocol transactions. At this point, the device must be able to recognize the protocol. The adjective method binds multiple packets based on a predefined unit. The adjective method is based on time and numbers. The numbers translate packets according to a certain number defined by the user. In the case of time, incoming packets are converted during a certain time period. Compressed conversion methods are harder to develop when using time units, but they can easily detect traffic-related attacks such as distributed denial of service (DDoS). In networks such as ICS and IVN, commands such as read are continuously and repeatedly transmitted. When most of the traffic share is filled with packets with the same time interval to execute the same command, the proposed method will be effective for detection when anomalous packets occur. This method is recommended for use as a warning, as the method based on repetitive patterns could be sensitive to functions that are rarely used.

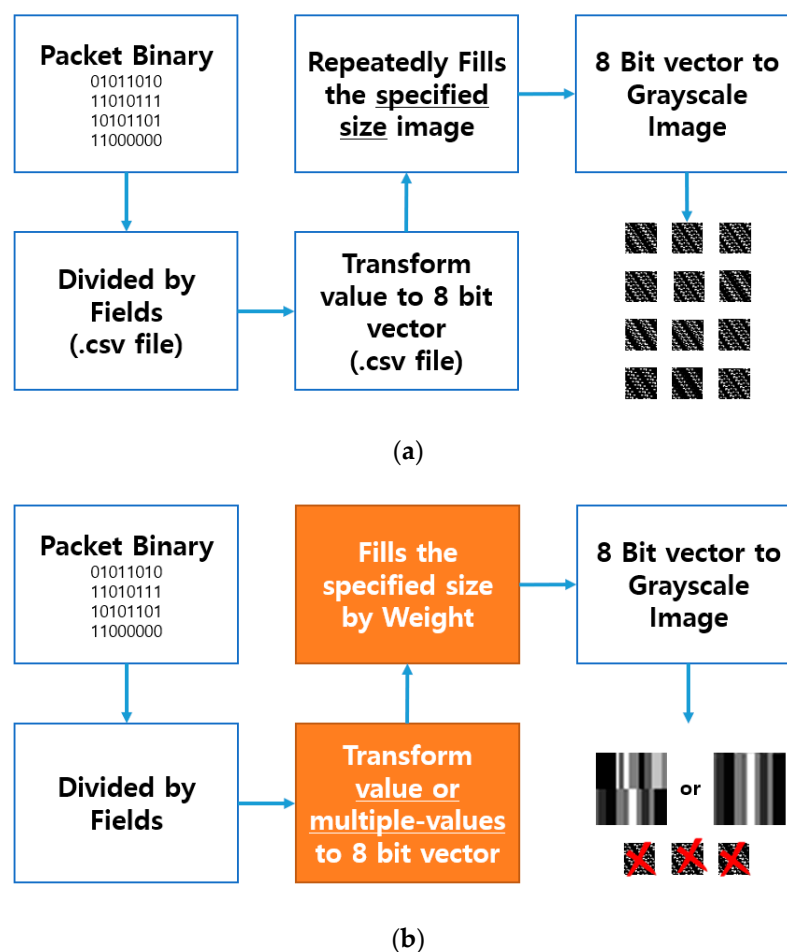


Figure 2. Cont.

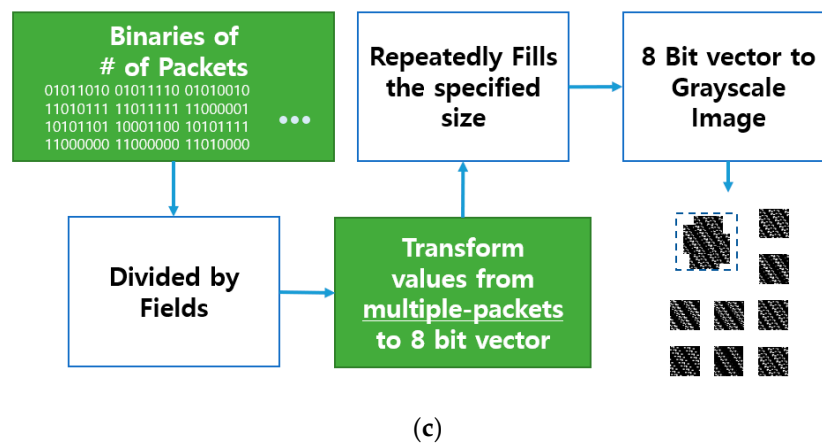


Figure 2. Proposed conversion methods: (a) direct conversion method, (b) weighted conversion method, (c) compressed conversion method.

4. Performance Measurements

In this section, the measured performance of the proposed techniques based on the NSL-KDD dataset is presented. The NSL-KDD dataset is an improved version of the KDD Cup 99 dataset [31]; it has been investigated by many IDS researchers to prove its performance. The NSL-KDD training dataset contains more than 100,000 packets, and these packets are processed to extract 41 features and then stored in csv format. The test dataset contains over 20,000 packets and 17 additional attacks that are not included in the training data. These NSL-KDD features allowed the precise measurement of the performance of the proposed techniques and the performance of various IDS studies. In addition, various use cases enabled the performance comparison between the techniques proposed in previous studies. However, problems could arise as other researchers did not use the NSL-KDD dataset in the same way and did not describe it clearly and concisely [3,27,32,33]. In the paper that analyzed the NSL-KDD dataset in detail, the number of “ps” belonging to “user to root (U2R)” was expressed as 5 even though 15 existed [33]. These confusing dataset uses prevent comparison with each study and measurement of the exact comparative advantage. The extremely small set of attacks in NSL-KDD are difficult to detect. Therefore, in this study, the performance was measured by adjusting the dataset to be the same as the dataset used in previous studies.

4.1. NSL-KDD Dataset

In this study, the NSL-KDD datasets used for performance measurement and comparison included three datasets: Train+, Test+, and Test-21 [17]. The Train+ dataset includes 23 packet types, including “normal”, and 22 attacks are classified into four attack types: denial of service (DoS), probe, user-to-root (U2R), and remote-to-local (R2L) [24]. Test-21 is a subset of Test+, and Test+ contains 17 additional attacks that do not exist in Train+. The characteristics of these NSL-KDD train/test datasets allowed us to measure the sensitivity of a learning-based IDS in response to unknown attacks. However, both the Train+ and Test+ datasets have biased data distributions, making them difficult to learn and detect [26]. Although this biased distribution may be considered as reflecting the actual environment as much as possible, its dispersion is poor because researchers focus on the balance of the dataset to detect the attack optimally even when learning the actual IDS. Hence, most studies did not use the dataset naturally [25], and detailed information regarding the datasets without modification is shown in Table 1.

As shown in Table 1, the Test+ dataset contains significantly more R2L attacks than Train+. There are few U2R attacks in two of the datasets (Test-21 is a subset of Test+; therefore, they are treated as two datasets), making learning more difficult.

Table 1. NSL-KDD dataset.

Type	Train+	Test+	Test-21
Normal	67,343	9710	2152
R2L	995	2885	2885
Probe	11,656	2421	2402
DoS	45,927	7460	4344
U2R	52	67	67
Total	125,973	22,543	11,850

Table 2 shows the number of additional attacks that are included in Test+. The ratio shows the number of additional attacks occupying the dataset. U2R is an extreme example, accounting for less than 0.1% in Train+ and less than 1% in Test+. Of U2R attacks, 44.78% are additional attacks. A serious imbalance of these datasets can degrade their reliability. Test-21, extracted from a subset of Test+ primarily from difficult-to-detect data, has only approximately half the number of the total packets. However, the additional attacks have only 10 fewer probe types compared to Test+, while the remainder of the measures are the same. Thus, 39% of the total 11,850 instances of the dataset are additional attacks, i.e., 3740 instances.

Table 2. NSL-KDD Test+ dataset comparison.

Type	Train+	Test+	Test-21
R2L	2885	686	23.78%
Probe	2421	1315	54.32%
DoS	7460	1719	23.04%
U2R	67	30	44.78%
Total	12,833	3750	29.22%

4.2. Accuracy Metrics

In multiclass classification problems, the meaning is difficult to decipher if each indicator (true, positive, false, negative) is not clarified when expressing the performance index (i.e., precision and recall) [18]. Therefore, IDS research typically treats true positives as an attack and a true positive is considered a successfully detected attack [18,25]. Otherwise, it directly describes the performance by referring to intuitive indicators such as “detected attacks” [34].

Herein, we use the metrics defined in Table 3:

Table 3. NSL-KDD dataset.

	Actual	Positive (Attack Packet: 1–4)	Negative (Normal Packet: 0)
Predicted			
Positive		True Positive-Attack Correctly classified	False Positive-Attack Incorrectly classified
Negative		False Negative-Normal Incorrectly classified	True Negative-Normal Correctly classified

In Table 3, positive/negative is classified as an attack/normal packet. In the implementation, attack packets have labels of 1–4 (true) and normal packets have labels of 0 (false).

4.3. Performance Comparison

Similar to most deep learning studies, the CNN model was developed based on TensorFlow (v1.8.0) and we used the MNIST code provided by the TensorFlow tutorial. All experiments were performed using a GPU GTX 1080Ti 11 GB and CPU i7-8600K based on Windows 10. The developed IDS was tested with fixed values: 100 for batch size and 10% dropout rate. The only variable that

changed was “step.” However, the performance change with the increase and decrease in “step” was slight. When “step” had an extremely low value of 100, a performance degradation of ~4% occurred. This performance change flow was expected because multiple fields existed in an image, thus enabling a large number of fields to be learned with a small number of steps. The learning model differs from the case where the performance suddenly improved due to overfitting by factors such as dropout.

Studies using a CNN in IDSs have been reported [16,24]. However, using a CNN as a classifier to solve the classification problem of IDSs is rare [24]. An autoencoder was used as a preprocessing method for feature extraction [16]. Although the CNN is generally fast, building an IDS using a CNN as preprocessing and learning it again using other learning techniques can weaken the real-time feature of the system, thus rendering it less practical.

Although NSL-KDD has been used in various studies, the total number of packets is rarely exactly the same as the number of packets per subcategory. Hence, different datasets may be downloaded and used according to the updates or edits for research purposes. However, if the performance is measured with a modified dataset, acceptable reasons and detailed changes must be specified [3]; otherwise, comparison will be difficult [25]. Figure 3 shows the performance using NSL-KDD Train+ for learning and NSL-KDD Test+ for testing.

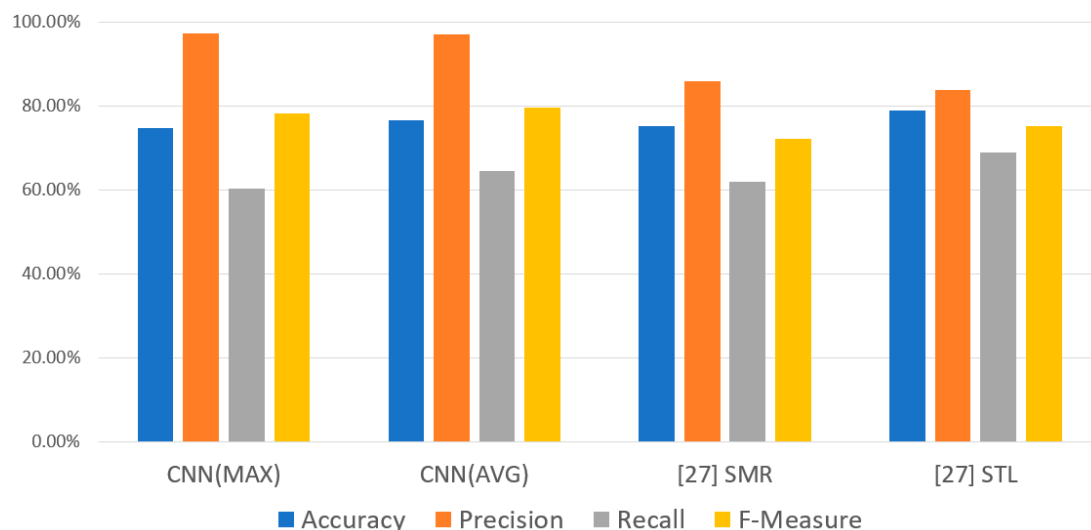


Figure 3. Performance comparison with non-modified NSL-KDD.

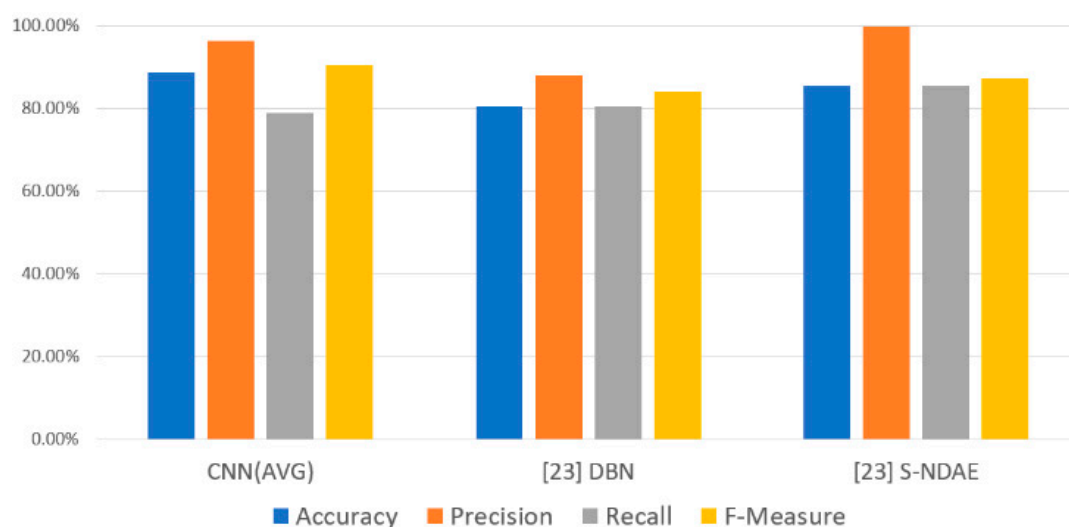
Tables 4 and 5 show the confusion matrix for the NSL-KDD Test+ dataset. U2r had the lowest amount of data because it had the fewest data point, and R2L had the worst result. In the modified dataset, R2L was 0% accurate, which is typical in deep-learning-based detections due to its structure that is highly similar to a normal structure [32]. Figures 3 and 4 compare the performance of the methods reported in relevant studies through bar graphs. The modified dataset is described below along with performance comparisons.

Table 4. Confusion matrix for non-modified NSL-KDD Test+.

Actual \ Predicted					
	Normal	R2L	Probe	DoS	U2R
Normal	9463	7	172	65	3
R2L	2689	70	120	2	4
Probe	737	1	1530	153	0
DoS	1064	0	178	6218	0
U2R	55	2	0	1	9

Table 5. Confusion matrix for modified NSL-KDD Test+.

Actual \ Predicted					
	Normal	R2L	Probe	DoS	U2R
Normal	9469	0	165	76	1
R2L	1219	0	12	0	0
Probe	255	0	845	6	0
DoS	212	0	25	5497	0
U2R	20	0	0	0	0

**Figure 4.** Performance comparison with modified NSL-KDD Test+.

The proposed techniques were expressed in the CNN (AVG/MAX) using average pooling (AVG) and max pooling (MAX), as their performances are highly affected by the kernel. Soft-max regression (SMR) and self-taught learning (STL) were measured [18], and Figure 3 shows that the proposed technique yielded slightly better precision and F-measure scores.

Figure 4 compares the performance using the modified NSL-KDD Test+ dataset. Because the modified dataset excludes 17 types of additional attacks, it was compared separately [19]. Stacked nonsymmetric deep auto-encoder (S-NDAE) performed better with 100% precision and 85.42% recall. However, the proposed method, with 88.82% accuracy and 90.67% F-measure, exhibited better performance. For the deep belief network (DBN), all performance indicators except recall were higher.

The performance of methods in existing studies using CNN as the main classifier is shown in Figure 5. The F-measure indicator was used for performance measurement and showed 99% performance with the KDD-Cup 99 dataset. However, when the NSL-KDD dataset was used, the performance was similar as Figure 5. The study graphically presented the results obtained but did not describe the exact performance result. Therefore, Figure 5 depicts an approximation of the degree indicated in the graph. The proposed method received an F-measure of 90.67%, which is a 15% performance improvement over the existing CNN study (approximately 79%) [22].

Although the time was not comparable to that of the GPU used in the related studies, the performance was similar: less than 30 s of learning time. In the largest number of steps, i.e., 80,000 entered attempts, 300 s were required from learning to classification. The step entered for performance comparison was 60,000, which consumed 225 s. Even with the GTX 1060 as a GPU, only 294 s were consumed.

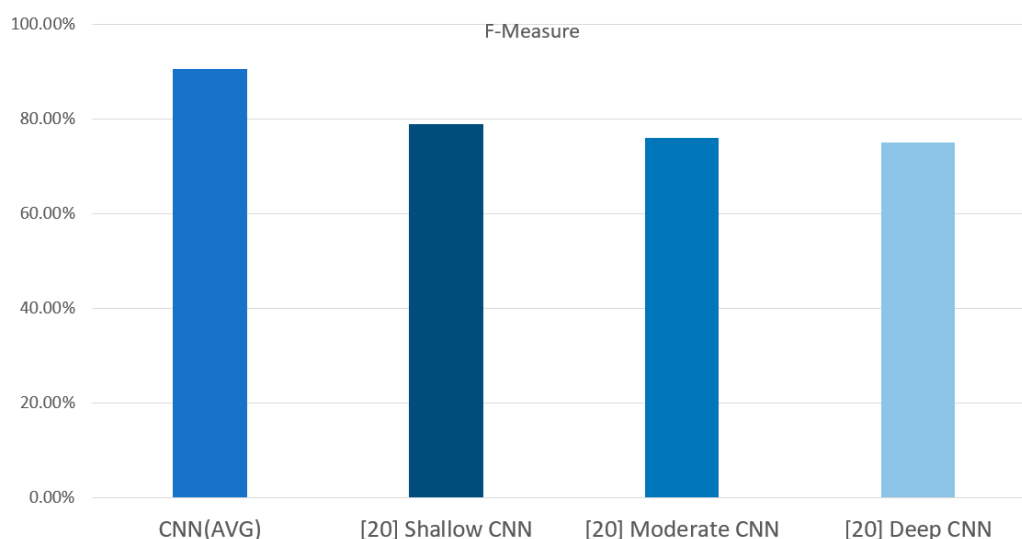


Figure 5. Performance comparison with CNN-based studies using F-Measure.

5. Discussion

The tested NSL-KDD dataset was provided separately from the training and testing datasets. The Test+ dataset includes additional attacks that are not included in training data, and Test- (also called Test-21) removes all duplicate attacks and includes only new attack types. This characteristic of NSL-KDD restricts the performance of overfitted learning-based IDS studies. When developing, we experimented to see if performance degradation occurred due to overtraining by setting the drop-ratio. Even then, the change in performance was less than 3%.

In most cases, when defining symbolic fields in protocol standards, similar types are often assigned sequentially. For example, in the function code of distributed network protocol 3 (DNP3), one of the ICS protocols, similar functions such as Read(1)/Write(2) or Solicited(129)/Unsolicited(130) have sequentially defined values. The characteristics of symbolic field values that are defined as having sequential values related to each other can be found in any field in an intuitive definition manner. Since the proposed pre-processing techniques maximize the characteristics of the field by maximizing the gaps to one another, the classifier can be better distinguished. This has more advantages compared to models that are unsupervised or use the simple data stream as the input. This is because the actual value of the symbolic field changes little, but the change in meaning is much larger than the change in value. In addition to the distinction of symbolic variables, the distinction between packets is also important. Each packet has its own function or purpose, and fields such as the function code mentioned above represent it. Therefore, when the process is repeated, such as ICS, patterns of the traffic are repeated, and frequently used services have a robust and periodic pattern. For example, in the case of DNP3, the function code may be weighted to help classification results, frequent functions may be “read”, and infrequent functions may be “unsolicited”, These frequent and infrequent elements are expected to yield effective performance using the third proposed preprocessing method, i.e., compressed conversion as a unit of time because, in most cases, frequent services such as read/write are periodically transmitted and received.

The CNN’s kernel handles multiple pixels together, and multiple fields are combined during this process. The randomly generated kernels can be induced to reflect hidden correlation information, including symbolic fields whose characteristics are maximized. For example, in ICSs or IVNs, packets such as read/write periodically transmitted by devices located in the leaf produce large amounts of traffic. At this time, the IP used to build the access list can be combined with periodic values or function codes and reflected to the CNN model to obtain an effect similar to automatic generation of various types of access lists.

The CNN is a supervised learning that is severely affected by train data set. For this reason, most NIDSs do not use CNN, which cannot perform well due to the lack of datasets. CNN is a proven DL model, and in practical terms, it can help network administrators apply security in the most efficient way to use their own network resources possible. Therefore, to properly apply the proposed technique, it is necessary to learn not only about normal network data, but also to include randomly generated attack data such as system shutdown or manipulation of critical data.

6. Conclusions

We proposed three preprocessing techniques for a CNN. One of the three proposed techniques, called direct conversion, was tested, and the other two techniques were not as they have more requirements than those applied in the tested technique. All three techniques are based on the same principle. They were designed to reflect the convolved characteristics of a CNN's input into the network packet data rather than image data traditionally received as the input.

We verified through the performance comparison that the proposed preprocessing method and the CNN model are sufficiently practical. We did not compare the performance with the deep-learning-based IDS study mainly using CNN including the shallow, moderate, and deep mentioned above. The study did not report the exact performance measures such as accuracy ratio, preventing accurate comparison. The only performance measure they provided, the result figure, showed that all proposed techniques did not exceed 80% accuracy using a graph. Therefore, the proposed preprocessing technique, which has over 88% accuracy, improved performance by more than 10% in the same environment (Train+/Test+) with a less-configured CNN [22]. Therefore, if an appropriate preprocessing method that can simultaneously activate the characteristics of the network and the CNN will perform better than other deep learning models.

For future works, the two untested preprocessing methods can be validated in a restricted environment like the IVN, and the proposed preprocessing methods can be used in combination. The proposed preprocessing methods focus on the meaning of one pixel. Unlike related studies that converted packet data into binary, adding one field to one pixel through preprocessing is highly advantageous for convolved learning. Therefore, a new method of preprocessing should be studied accordingly.

Author Contributions: T.S. and C.L. conceived and designed the experiments; W.J. performed the experiments and analyzed the data; W.J. and S.K. contributed building IDS prototype tool; W.J. wrote the paper; T.S. conducted the direction of the study as a supervisor. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by Energy Cloud R&D Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT (2019M3F2A1073386). This work was supported by the ICT R&D program of MSIT/IITP (No. 2018-0-00336, Advanced Manufacturing Process Anomaly Detection to prevent the Smart Factory Operation Failure by Cyber Attacks).

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

References

1. Jo, W.; Shin, Y.; Kim, H.; Yoo, D.; Kim, D.; Kang, C.; Jin, J.; Oh, J.; Na, B.; Shon, T. Digital Forensic Practices and Methodologies for AI Speaker Ecosystems. *Digit. Investig.* **2019**, *29*, S80–S93. [\[CrossRef\]](#)
2. Selvakumar, K.; Karuppiah, M.; Sai Ramesh, L.; Hafizul Islam, S.K.; Hassan, M.M.; Fortino, G.; Raymond Choo, K.-K. Intelligent Temporal Classification and Fuzzy Rough Set-Based Feature Selection Algorithm for Intrusion Detection System in WSNs. *Inf. Sci.* **2019**, *497*, 77–90. [\[CrossRef\]](#)
3. Wang, P.; Yang, L.T.; Kuang, L.; Nie, X.; Ren, Z.; Li, J. Data-Driven Software Defined Network Attack Detection: State-of-the-Art and Perspectives. *Inf. Sci.* **2019**, *513*, 65–83. [\[CrossRef\]](#)
4. Li, T.-H.S.; Chang, S.-J.; Chen, Y.-X. Implementation of Human-like Driving Skills by Autonomous Fuzzy Behavior Control on an FPGA-Based Car-like Mobile Robot. *IEEE Trans. Ind. Electron.* **2003**, *50*, 867–880. [\[CrossRef\]](#)

5. Guo, K.; Zeng, S.; Yu, J.; Wang, Y.; Yang, H. A Survey of FPGA-Based Neural Network. 2017. Accelerator. *arXiv* **2017**, arXiv:1712.08934.
6. National Institute of Standards and Technology (NIST). *Security and Privacy Controls for Federal Information Systems and Organizations*; SP 800-53r4 SC-29; National Institute of Standards and Technology (NIST): Gaithersburg, MD, USA, 2015.
7. Slepian, M. NERC Fines Duke Energy \$10 Million for Cybersecurity Failings. 2019. Available online: <https://www.itgovernanceusa.com/blog/nerc-fines-duke-energy-10-million-for-cybersecurity-failings> (accessed on 16 April 2020).
8. Dragos, TRISIS Malware—Analysis of Safety System Targeted Malware. Version 1. 2017. Available online: <https://dragos.com/wp-content/uploads/TRISIS-01.pdf> (accessed on 16 April 2020).
9. Lyngaas, S. The Group behind Trisis Has Expanded Its Targeting to the U.S. Electric Sector. 2019. Available online: <https://www.cyberscoop.com/trisis-xenotime-us-electric-sector/> (accessed on 16 April 2020).
10. Nuclear Agency Expands Probe into Manual Shutdown of Hanbit 1 Reactor. May 2019. Available online: <https://en.yna.co.kr/view/AEN20190520004600320> (accessed on 16 April 2020).
11. Yasser, A.-E.; Ekram, H. The D-OMA Method for Massive Multiple Access in 6G: Performance, Security, and Challenges. *IEEE Veh. Technol. Mag.* **2019**, *14*, 92–99.
12. Mahmood, K.; Naqvi, H.; Ahmad, H.F.; Shon, T.; Chaudhry, S.A. A Lightweight Message Authentication Scheme for Smart Grid Communications in Power Sector. *Comput. Electr. Eng.* **2016**, *52*, 114–124.
13. Chaudhry, S.A.; Naqvi, H.; Farash, M.S.; Ramzan, M.S.; Shon, T. An Improved and Robust Biometrics-Based Three Factor Authentication Scheme for Multiserver Environments. *J. Supercomput.* **2018**, *74*, 3504–3520. [[CrossRef](#)]
14. Yoo, H.; Taeshik, S. Challenges and Research Directions for Heterogeneous Cyber–Physical System Based on IEC 61850: Vulnerabilities, Security Requirements, and Security Architecture. *Future Gener. Comput. Syst.* **2016**, *61*, 128–136. [[CrossRef](#)]
15. Chaudhry, S.A.; Shon, T.; Alsharif, M.H.; Al-Turjman, F. Correcting Design Flaws: An Improved and Cloud Assisted Key Agreement Scheme in Cyber Physical Systems. *Comput. Commun.* **2020**, *153*, 527–537. [[CrossRef](#)]
16. Kwon, S.; Jaehan, J.; Taeshik, S. Toward Security Enhanced Provisioning in Industrial IoT Systems. *Sensors* **2018**, *18*, 4372. [[CrossRef](#)]
17. University of New Brunswick. NSL-KDD Dataset. Available online: <https://www.unb.ca/cic/datasets/nsl.html> (accessed on 16 April 2020).
18. Nathan, S.; Ngoc, T.N.; Phai, V.D.; Shi, Q. A Deep Learning Approach to Network Intrusion Detection. *IEEE Trans. Emerg Top. Comput. Intell.* **2018**, *2*, 41–50.
19. Javaid, A.Y.; Niyaz, Q.; Sun, W.; Alam, M. A deep Learning Approach for Network Intrusion Detection System. In Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS), New York, NY, USA, 3–5 December 2015.
20. Chalapathy, R.; Sanjay, C. Deep Learning for Anomaly Detection: A Survey. *arXiv* **2019**, arXiv:1901.03407, 2019.
21. Kim, S.; Jo, W.; Shon, T. APAD: Autoencoder-based Payload Anomaly Detection for industrial IoE. *Appl. Soft Comput.* **2020**, *88*, 106017. [[CrossRef](#)]
22. Kwon, D.; Natarajan, K.; Suh, S.C.; Kim, H.; Kim, J. An Empirical Study on Network Anomaly Detection using Convolutional Neural Networks. In Proceedings of the 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS), Vienna, Austria, 2–6 July 2018.
23. Scarfone, K.; Mell, P. Guide to Intrusion Detection and Prevention Systems (IDPS) Recommendations of the National Institute of Standards and Technology. *NIST Spec. Publ.* **2007**, *800*, 127.
24. Dhanabal, L.; Shanharajah, S.P. A Study on NSL-KDD Dataset for Intrusion Detection System Based on Classification Algorithms. *Int. J. Adv. Res. Comput. Commun. Eng.* **2015**, *4*, 446–452.
25. Wang, W.; Liu, J.; Pitsilis, G.; Zhang, X. Abstracting Massive Data for Lightweight Intrusion Detection in Computer Networks. *Inf. Sci.* **2018**, *433*, 417–430. [[CrossRef](#)]
26. Tavallaee, M.; Bagheri, E.; Lu, W.; Ghorbani, A.A. A Detailed Analysis of the KDD CUP 99 Data Set. In Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications 2009, Ottawa, ON, Canada, 8–10 December 2009. [[CrossRef](#)]

27. Kim, J.; Kim, J.; Thu, H.L.T.; Kim, H. Long Short Term Memory Recurrent Neural Network Classifier for Intrusion Detection. In Proceedings of the 2016 International Conference on Platform Technology and Service (PlatCon), Jeju, Korea, 15–17 February 2016.
28. Kwon, S.; Yoo, H.; Shon, T. IEEE 1815.1-Based Power System Security with Bidirectional RNN-Based Network Anomalous Attack Detection for Cyber-Physical System. *IEEE Access* **2020**, *8*, 77572–77586. [[CrossRef](#)]
29. Mohammadpour, L.; Ling, T.C.; Liew, C.S.; Chong, C.Y. A Convolutional Neural Network for Network Intrusion Detection System. In Proceedings of the 15th APAN Research Workshop 2018, Auckland, New Zealand, 6 August 2018; pp. 50–55.
30. Vinayakumar, R.; Soman, K.P.; Poornachandran, P. Applying Convolutional Neural Network for Network Intrusion Detection. In Proceedings of the 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Udupi, India, 13–16 September 2017; pp. 1222–1228.
31. Nataraj, L.; Jacob, G.; Karthikeyan, S.; Manjunath, B.S. Malware Images: Visualization and Automatic Classification. In Proceedings of the 8th International Symposium on Visualization for Cyber Security, Santa Barbara, CA, USA, 20 July 2011; pp. 1–7.
32. Le, Q.; Boydell, O.; Mac Namee, B.; Scanlon, M. Deep learning at the shallow end: Malware classification for non-domain experts. *Digit. Investig.* **2018**, *26*, S118–S126. [[CrossRef](#)]
33. Protić, D.D. Review of KDD Cup '99, NSL-KDD and Kyoto 2006+ datasets. *Vojn. Glas.* **2018**, *66*, 580–596. [[CrossRef](#)]
34. Liu, Y.; Liu, S.; Zhao, X. Intrusion Detection Algorithm Based on Convolutional Neural Network. *DEStech Trans. Eng. Technol. Res.* **2018**. [[CrossRef](#)]



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).