symmetry

MDPI

Article

An Efficient Hyperparameter Control Method for a Network Intrusion Detection System Based on Proximal Policy Optimization

Hyojoon Han, Hyukho Kim and Yangwoo Kim *

Department of Information and Communication Engineering, Dongguk University, Seoul 04620, Korea; han6343@dongguk.edu (H.H.); hyukho714@dongguk.edu (H.K.)
* Correspondence: ywkim@dongguk.edu

Abstract: The complexity of network intrusion detection systems (IDSs) is increasing due to the continuous increases in network traffic, various attacks and the ever-changing network environment. In addition, network traffic is asymmetric with few attack data, but the attack data are so complex that it is difficult to detect one. Many studies on improving intrusion detection performance using feature engineering have been conducted. These studies work well in the dataset environment; however, it is challenging to cope with a changing network environment. This paper proposes an intrusion detection hyperparameter control system (IDHCS) that controls and trains a deep neural network (DNN) feature extractor and *k*-means clustering module as a reinforcement learning model based on proximal policy optimization (PPO). An IDHCS controls the DNN feature extractor to extract the most valuable features in the network environment, and identifies intrusion through *k*-means clustering. Through iterative learning using the PPO-based reinforcement learning model, the system is optimized to improve performance automatically according to the network environment, where the IDHCS is used. Experiments were conducted to evaluate the system performance using the CICIDS2017 and UNSW-NB15 datasets. In CICIDS2017, an F1-score of 0.96552 was achieved and UNSW-NB15 achieved an F1-score of 0.94268. An experiment was conducted by merging the two datasets to build a more extensive and complex test environment. By merging datasets, the attack types in the experiment became more diverse and their patterns became more complex. An F1-score of 0.93567 was achieved in the merged dataset, indicating 97% to 99% performance compared with CICIDS2017 and UNSW-NB15. The results reveal that the proposed IDHCS improved the performance of the IDS by automating learning new types of attacks by managing intrusion detection features regardless of the network environment changes through continuous learning.

Keywords: intrusion detection system (IDS); reinforcement learning (RL); proximal policy optimization (PPO); deep learning (DL); deep neural network (DNN); *k*-means; CICIDS2017; UNSW-NB15; network security; feature extraction

Citation: Han, H.; Kim, H.; Kim, Y. An Efficient Hyperparameter Control Method for a Network Intrusion Detection System Based on Proximal Policy Optimization. *Symmetry* **2022**, *14*, 161. https://doi.org/10.3390/sym14010161

Academic Editors: Chin-Ling Chen and José Carlos R. Alcantud

Received: 24 November 2021
Accepted: 31 December 2021
Published: 14 January 2022

1. Introduction

As the use of large-scale high-performance systems, such as cloud systems, increases, the number of network packets also rapidly increases. Network traffic is asymmetric, as normal data are more numerous than the attack data. In addition, due to the high complexity of attack data, it is difficult to distinguish them from normal data, making it challenging to detect attack data. As a result, the network intrusion detection system (IDS) [1], which analyzes network attacks, has become more complex.

A network IDS belongs to one of two types based on the detection technique [2]. The misuse-based IDS targets a specific pattern, and if the pattern is included in the network traffic, it is regarded as an attack. This technique has the advantage of reliably detecting specific attacks. The existing system environment has the advantage of having a high true-positive detection rate for known attacks. However, this technique has the disadvantage

that a new type of attack cannot be detected, and the detection speed is significantly lower in a big data environment than in the existing environment.

In contrast, an abnormal behavior-based IDS detects intrusions by analyzing traffic patterns. An abnormal behavior-based IDS analyzes the normal network traffic and considers operations that deviate from normal behaviors to be intrusions. Although this method can detect new types of attacks, it has the disadvantage of having a high false-positive detection rate, as it is difficult to set a threshold for normal data.

Recently, many studies in which artificial intelligence (AI) technologies are combined have been conducted to further develop the various fields [3,4], as well as IDS. In early studies, intrusion detection research used machine learning models, such as the decision tree, support vector machine (SVM) and artificial neural network (ANN) [5–7]. As interest in and research on deep learning has increased, studies that use deep learning techniques, such as deep reinforcement learning, the recurrent neural network and the deep neural network (DNN), have been conducted [8–10]. Although these studies have displayed high performance on low-complexity datasets, such as KDDCUP-99 and NSL-KDD, they have had difficulty detecting recent high-complexity attacks, such as those in the UNSW-NB15 dataset [11].

Various hybrid studies have been conducted to detect attacks with high complexity. In [12], spark machine learning and convolutional long short-term memory hybrid algorithms were used to analyze the ISCX-UNB dataset. A hybrid machine learning technique using the *k*-means algorithm and SVM was proposed in [13].

Reinforcement learning is a trial-and-error-based learning system, and the number of studies that apply reinforcement learning to IDSs is increasing. An evaluation using a deep *Q* network-based algorithm was conducted on the NSL-KDD and UNSW-NB15 datasets in [14]. In [15], an IDS was built with adversarial/multiagent reinforcement learning using a deep *Q*-learning algorithm. Although these hybrid IDSs exhibited good performance on complex datasets, such as ISCX-UNB and UNSW-NB15, the performance varied widely in some cases, depending on the dataset.

In the study of the IDS to which reinforcement learning is applied, the learning subject was intrusion detection data. Reinforcement learning is often used in games, autonomous driving and smart factories as a structure to learn how to gain benefits through system control [16–19]. This study introduces a learning method that takes advantage of the system operation structure by applying reinforcement learning to the control part of the IDS—a different direction from previous studies.

Because intrusion detection data have many features, they are greatly influenced by learning. Speed degradation may occur when many features exist, and performance may be degraded due to overfitting [20]. Thus, it is important to reduce the number of features appropriately through feature engineering. Feature selection and feature extraction are widely used in feature engineering. Feature selection simplifies features by removing features that have no value or that overlap from the full-feature set. Feature extraction combines the existing full-feature set to create new features. Many studies have improved the intrusion detection speed or increased the detection rate through feature engineering [21–23]. Prasad et al. [24] demonstrated that the effective feature selection set differs for different attack patterns through different feature selection, even within the same dataset. However, it is costly and challenging to perform feature extraction according to the various attack patterns in a real environment.

Selecting valid features for feature engineering is a challenging task. An expert who can understand and analyze the characteristics of packets should analyze the network environment and select features relevant to the attack. If the network environment changes even a little, it may be necessary to select other features. These tasks can be very time-consuming. Moreover, the expert may respond too late to new attacks because the environment changes quickly.

In addition, the dataset used in IDS research is quite important to the direction of the research. Datasets used for the network packet analysis of commercial products are

difficult to use for research due to privacy issues. Therefore, published datasets are used in many studies as a benchmark to evaluate the IDS. Various types of datasets are used for intrusion detection, each with its own characteristics. As the attack scenarios and types are also different, the results of a learning algorithm are quite diverse, depending on the dataset. In addition, because these studies are trained according to a specific dataset, there are cases in which the results are inferior for other types of datasets [25]. Therefore, many attacks may not be detected if an algorithm with high performance only in a specific dataset is used without considering the network environment when introducing the IDS. Therefore, a method that improves the IDS performance regardless of the dataset is required when introducing the IDS in a real environment.

In this study, we propose an intrusion detection hyperparameter control system (IDHCS) based on reinforcement learning to solve the above problems. The IDHCS consists of a DNN-based feature extractor and *k*-means-based clustering. In the IDHCS, feature extraction is performed using a deep learning system based on the DNN to respond to rapidly changing features in each network environment. The DNN quickly analyzes characteristics and extracts valuable data for intrusion detection. Next, clustering is performed with the *k*-means algorithm to distinguish the attack data from the normal data. In addition, to build an IDS that can quickly respond to a changing network environment, a method to control intrusion detection hyperparameters based on reinforcement learning is developed. The IDS studies that use reinforcement learning have generally learned using intrusion detection datasets. However, the proposed reinforcement learning system controls and learns the algorithm that operates the IDS.

In machine learning modeling, the value set by the user is called a hyperparameter. Reinforcement learning controls and learns the feature extraction and clustering hyperparameters and immediately analyzes the dataset features. Even when the DNN feature extractor and *k*-means clusters are not trained using reinforcement learning, good results can be obtained when training targets on a specific dataset and detecting intrusions. However, good performance cannot be expected on datasets with even slightly different attack patterns. Therefore, the performance of the feature extractor and *k*-means cluster must be continuously adjusted through the hyperparameter control system proposed in this study. Using this approach, the IDS automatically adapts according to the changes in the network environment, and it can respond to attacks without user involvement. Performance improvement is possible because the IDHCS finds and learns the optimal value.

The structure of this manuscript is as follows. Section 2 examines IDS studies using machine learning. Next, Section 3 examines the structure and dataset of the proposed system. Section 4 evaluates the performance of the proposed system and Section 5 concludes the study.

2. Related Work

The latest network attacks take various forms, and have very high complexity as their patterns are very similar to those of normal network traffic. The system performance must be verified using a dataset that includes the latest attack types to respond to these network attacks. In addition, CICIDS2017 and UNSW-NB15 are representative intrusion detection datasets that include the latest attack types. Various studies have analyzed the above datasets using AI algorithms.

In [26], the CICIDS2017 dataset was classified using the decision tree, naïve Bayes, random forest and SVM methods. The experiment was conducted by extracting ten high-value features. Naïve Bayes exhibited the best performance with an F1-score of 0.92481. In [26], the information gain for each feature was measured by evaluating feature-based entropy for feature selection. However, not all ten high-value features perform best in all environments. In contrast, the proposed system uses a DNN to extract new features from the full-feature set according to the changing environment, so there is no need to select specific features according to the attack type. It also extracts the optimal number of features that can detect intrusions using reinforcement learning.

In [27], a system optimized for distributed denial of service (DDoS) detection was designed by combining an autoencoder (AE) and a DNN. The AE performed hyperparameter optimization using sparsity, unit standardization, orthogonality and a grid search, and the DNN performed intelligent learning rate determination and optimization using hyperband tuning. The F1-scores of 0.9835 on CICIDS2017 and 0.9857 on NSL-KDD were obtained from training the DNN based on 25 main features obtained by the AE. In contrast, in the proposed system, the number of features extracted from the DNN may vary, considering that the importance of the features depends on the network environment.

In [28], an F1-score of 0.8183 was obtained on CICIDS2017 using AdaBoost, principal component analysis (PCA) and the synthetic minority oversampling technique (SMOTE). In addition, SMOTE was used to solve the imbalance in the training data, and the PCA and ensemble feature selection techniques were used to select the main features.

In [29], feature selection was performed using recursive feature elimination and random forest. The selected features classified the UNSW-NB15 dataset using the decision tree, naïve Bayes and SVM. Naïve Bayes achieved the best F1-score of 0.824. In [27], the binary classification technique classified even detailed attacks using the polynomial classifier. In the proposed system, only normal and attack data are classified through clustering.

In [30], 45 features of the UNSW-NB15 dataset were extracted into four features using a combined random forest and decision tree algorithm. An F1-score of 0.92018 was obtained as a result of classifying the extracted features using an ANN. In this paper, as in [30], we propose reducing the dimensionality of the data to improve detection performance. However, the number of extracted features is not fixed to cope with the active network environment. In addition, new features are generated through deep learning rather than by using the importance of the features as determined by the classifier.

In the feature engineering applied in the above studies, many papers use a specific number of features determined to achieve good efficiency in a specific dataset. As in [25], studies focused on a specific dataset may exhibit good performance on that dataset but may have poor results when using other datasets. However, due to the nature of IDS research, the types of datasets used are not very diverse, so it can only be considered a limitation. In this paper, we examine how to automate the IDS using reinforcement learning to respond to changes in the network environment and overcome dataset limitations by merging datasets.

3. Intrusion Detection Hyperparameter Control System

In this section, we investigate the overall structure of the IDHCS. We examine the modules composing the IDS and main algorithms that operate each module. In addition, we assess the characteristics of the dataset used to evaluate the IDS.

3.1. Hyperparameter Control System

We review an IDS consisting of a DNN feature extractor, a k-means clustering intrusion detection module and an intrusion detection hyperparameter control module using proximal policy optimization (PPO). The configuration of the IDS is presented in Figure 1. The preprocessing stage refines packet data and applies feature extraction, and the intrusion detection module identifies intrusion by clustering data. The IDS also consists of a reinforcement learning agent that controls the preprocessor and clustering module to improve system performance and automation. The reinforcement learning agents improve performance by updating the policies in the preprocessors and clustering module. The preprocessor, clustering module and reinforcement learning agent run repeatedly until a suitable performance is achieved.

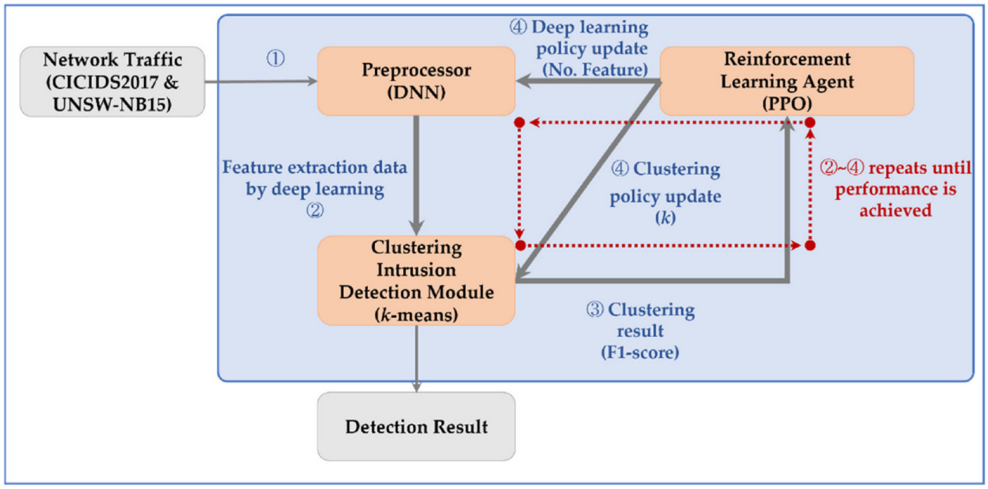

Figure 1. Process for intrusion detection hyperparameter control system.

3.1.1. Feature Extraction Using Deep Neural Network (Preprocessor)

In the preprocessing stage, data refinement is performed to cluster incoming packet data. The intrusion detection dataset consists of many features. When intrusion detection is performed using all features, performance may deteriorate due to overfitting problems. As in [24], it can be challenging to cover all attack patterns through feature selection because the feature groups that are easy to detect for each attack type are different. Therefore, existing features are combined and used in the proposed system through feature extraction. Many studies have proved that using feature extraction improves intrusion detection performance compared with using all features [31–33]. However, the number of features to extract inevitably varies from situation to situation. Therefore, through the reinforcement learning control algorithm, the above feature extraction conditions are controlled and features appropriate to the situation are extracted. As illustrated in Figure 2, In the preprocessing stage, feature extraction is performed using the DNN. The reinforcement learning algorithm adjusts N output stages according to the learning result and extracts the optimal features for intrusion detection. The number of features input into the preprocessor depends on the dataset used. For example, CICIDS2017 uses 78 features and UNSW-NB15 uses 48 features. The input data go through the hidden layer of the DNN and come out as the N output value. The N value is determined by reinforcement learning, and is used to determine the intrusion in the clustering algorithm through the extracted features.

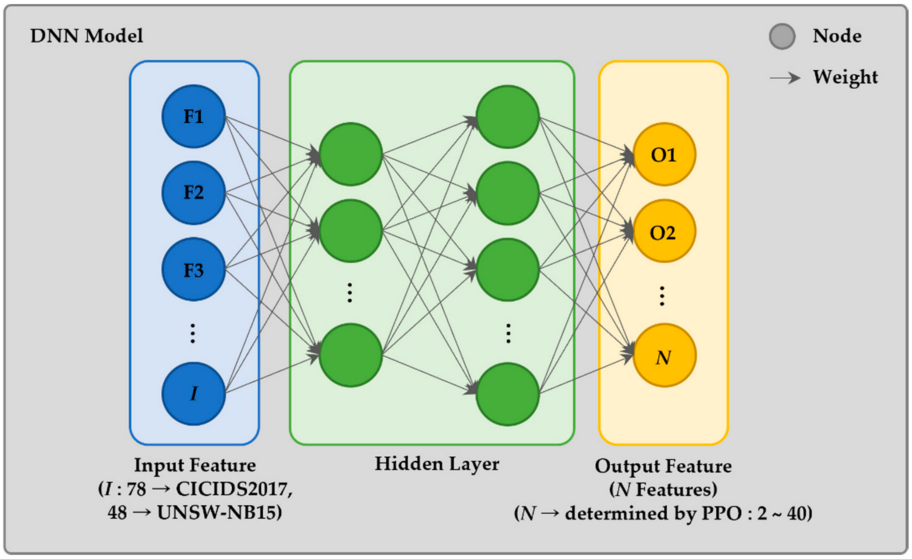

Figure 2. Deep neural network (DNN)-based feature extractor.

3.1.2. *k*-Means Cluster Module

In the clustering intrusion detection module, the attack and normal data are clustered using features extracted from the DNN. The *k*-means [34] method is used as the clustering algorithm and the characteristics of features are analyzed and classified into *k* clusters. Additionally, *k*-means is divided into *k* clusters according to the similarity of the given data. Through continuous learning, new data can be quickly classified if a well-established center point exists for attack and normal data. This paper aims to respond quickly to the rapidly changing network environment. Other supervised learning-based systems detect intrusion based on the learned network pattern. It can be used in a single system where the network environment does not change quickly, such as an IDS dedicated to a web or search server.

In this study, however, similar data are clustered using the unsupervised k-means clustering algorithm to cope with the rapidly changing network environment. The most important thing in *k*-means is to set the number of clusters. Data in the IDS can be classified into two types, attack and normal data, but upon close examination, several patterns exist in attack and normal data. Normal activities, such as database access and log searches, have different characteristics and attack patterns, such as a port scan and denial of service (DoS), which also have different characteristics. If there is attack data between different normal patterns, the attack cannot be detected if it is clustered into two groups.

Therefore, it is necessary to find the optimal number of clusters to classify the attack and normal patterns in detail according to the network environment. As the number of clusters increases, the probability of detecting an intrusion may increase as data can be classified more precisely, but this is not necessarily the case. In addition, it is challenging to determine the optimal number of clusters as the detection time can increase significantly as the number of clusters increases. In this study, the number *k* of the clustering algorithm is controlled and learned through the PPO algorithm to determine the optimal value.

3.1.3. Intrusion Detection Hyperparameter Condition Controller

The intrusion detection control algorithm proposed in this study is a method that enables the automation and improved performance of the IDS by controlling feature extraction and clustering methods using PPO. The reinforcement learning algorithm is based on trial and error, and is divided into model-based and model-free methods depending on the existence of an environmental model [35]. Reinforcement learning in model-based methods is a learning method that enables efficient behavior, knowing how the environment will change according to the behavior. The model-free method is used when it is difficult to build a model for the environment, and it is a model that learns with higher rewards by passively acquiring the next state and reward.

In this study, we introduce a model-free method so that the reinforcement learning environment can learn how to receive a high reward through manipulation the IDS. In addition, the model-free-based reinforcement learning algorithm can be classified into a policy optimization method and a Q-learning method. The policy optimization method seems more suitable for stably improving the performance of the IDS.

Algorithms related to model-free based policy optimization include actor-critic [36] and asynchronous advantage actor-critic (A3C) [37], focusing on parallel training in the actor-critic method and advantage actor-critic (A2C) [37] to solve agent update problems due to asynchronous A3C. In addition, such algorithms as soft actor-critic [38], which encourages exploration by incorporating the measures of policy entropy into rewards, are widely used. Moreover, deterministic policy gradient (DPG) [39]-type algorithms model the reinforcement learning policy as a deterministic decision. Deep DPG (DDPG) [40] combines deep Q network and DPG, and has been improved to learn high-dimensional continuous action space policies. In addition, such algorithms as distributed distribution DDPG [41] partially improve DDPG and execute it in a distributed way. The multiagent DDPG [42] extends DDPG to an environment where multiple agents complete tasks with only local information.

The trust region policy optimization (TRPO) [43] algorithm was created to reliably train DNN policies in a controlled state. It is a method of learning complex policies through an objective function called a surrogate. However, TRPO is complex and difficult to implement. To solve this problem, Schulman [43] developed a simpler and more generalized PPO algorithm [44] than TRPO. This PPO algorithm is similar to TRPO, but is relatively simple and easy to implement. In this study, The PPO algorithm was introduced for the stable operation and performance of the IDS. The PPO algorithm, which performs well in many environments [45,46], was expected to perform well in the operation of the IDS.

Algorithm 1 increases the performance of the IDS by manipulating the DNN feature extractor and *k*-means cluster module using the tuned PPO. The policy optimizes the number of features extracted by the DNN feature extractor and the k value of the k-means cluster module. The objective function for policy optimization is obtained in Equation (1) [44]:

$$L_t^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t\left[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](S_t)\right] \quad (1)$$

Algorithm 1: Update IDHCS policy Using PPO

Input: ***D*** //Intrusion Detection Dataset (CICIDS2017 or UNSW-NB15 or both)
Begin
Initialize: ***I*** //The number of features to extract.
Initialize: ***K*** //The number to cluster.
for iteration = 1,2, . . . do
***F*(F$_0$, F$_1$, . . . , F$_{n-1}$) = DNNFeatureExtractor(*D*, *I*)** //A extracted features by DNN.
***S* = KmeansCluster(*F*, *K*)** //F1 score obtained by *k*-means Cluster
for actor = 1,2, . . . ,M **do**
Run policy $\pi_{\theta old}$ in environment for T timesteps
Compute advantage estimates $\hat{A}_1, \ldots, \hat{A}_T$
end for
Optimize surrogate L wrt, θ, with K epochs and minibatch size $M \leq NT$
$\theta_{old} \leftarrow \theta$
***I*, *K* = updatepolicy(θ)**
end for
end;

3.2. Datasets

We used several intrusion detection dataset types verified and published for evaluating IDSs. Datasets can be classified according to whether complete packets, real data, zero-day attacks and modern attacks are included [47]. The KDDCUP-99 and NSL-KDD datasets produced by defense advanced research projects agency have been used in many studies in IDS research on abnormal behavior [48]. The above datasets have had a tremendous influence on the development of IDSs, but many parts are unsuitable for use now due to outdated attacks, lack of attack data complexity and lack of attack variety two decades after they were published. In addition, the University of New Brunswick (UNB) published datasets such as CICIDS2017, which include the latest attacks. Old datasets are unsuitable for use due to their limitations in traffic diversity and volume, lack of anonymized packet information and payload and restrictions on a diversity of attacks [49]. Therefore, we built an IDS that can detect various attacks by reducing dependency on the dataset using two reliable datasets that include the latest attacks.

The datasets in this study are CICIDS2017 and UNSW-NB15. The experiment was conducted using each dataset independently. In addition, the two datasets were merged and tested to confirm whether the reinforcement learning algorithm is trained to determine an intrusion according to a change in the network environment. The characteristics of each dataset are discussed as follows.

3.2.1. CICIDS2017 Dataset

The UNB published 11 intrusion detection datasets between 1998 to 2016, which helped many IDS studies. However, it was recognized that these datasets were insufficient in analyzing modern network patterns due to their lack of data volume and attack diversity. Therefore, according to modern network trends, UNB published CICIDS2017, which includes seven attack types: brute force, DoS, Heartbleed, web attack, infiltration, botnet and DDoS. In addition, CICIDS2017 was produced by capturing packets based on the contents of the attacks during working hours from 9:00 a.m. to 5:00 p.m. for five days from Monday, 3 July to Friday, 7 July 2017. Moreover, CICIDS2017 established ten criteria (listed in Table 1) to evaluate the reliability of the dataset.

Table 1. Criteria of CICIDS2017 [50].

Criteria	Explanation
Complete network configuration	The complete network topology includes modems, firewalls, switches, routers and various operating systems such as Windows, Ubuntu and Mac OS X.
Complete traffic	Deploy a user profiling agent and 12 different systems into real attacks from the victim and attack networks.
Labelled dataset	Details of attack timing are published in the dataset document
Complete interaction	CICIDS2017 covers both internal local area networks through two different networks and Internet communication
Complete capture	CICIDS2017 log all traffic to the storage server using a taping system and mirror port.
Available protocols	All available common protocols such as HTTP, HTTPS, FTP, SSH and email protocols, are provided.
Attack diversity	Includes the most common attacks, such as web-based, brute force, DoS, DDoS, infiltration, Heart-bleed, bot and port scan, based on a 2016 McAfee report.
Heterogeneity	During the attack execution, network traffic is captured from the main switch and memory dump and system calls are captured from all targeted systems.
Feature set	CICIDS2017 uses CICFlowMeter to extract more than 80 network flow features from generated network traffic and pass the network flow dataset as a CSV file.
Meta data	Fully described dataset with time, attack, flow and labels in the published article.

Additionally, CICIDS2017 consists of eight files and attacks were carried out on each day of the week. The composition of each file is presented in Table 2. In this experiment, all files were integrated and the experiment was conducted as one file.

Table 2. CICIDS2017 components.

Filename	No. Benign	No. Malicious
Monday-WorkingHours	529,918	0
Tuesday-WorkingHours	432,074	13,835
Wednesday-WorkingHours	440,031	252,672
Thursday-WorkingHours-Morning-WebAttacks	168,186	2180
Thursday-WorkingHours-Afternoon-Infiltration	288,566	36
Friday-WorkingHours-Morning	189,067	1966
Friday-WorkingHours-Afternoon-PortScan	127,537	158,930
Friday-WorkingHours-Afternoon-DDos	97,718	128,027
Total	2,273,097	557,646

Table 3 lists the number of data observations in CICIDS2017 by type. Of 2,830,743 data observations, about 80% are normal data and about 20% are attack data. In addition, the seven types of attacks consist of 14 different types of detail attacks. This dataset is suitable for judging the performance of the IDS.

Table 3. CICIDS2017 data by type.

Data	No	Data	No
Benign	2,273,097	DoS Slowhttptest	5499
DoS Hulk	231,073	Bot	1966
PortScan	158,930	Web Attack Brute Force	1507
DDoS	128,027	Web Attack XSS	652
DoS GoldenEye	10,293	Infiltration	36
FTP_Patator	7938	Web Attack SQL Injection	21
SSH_Patator	5897	Heartbleed	11
DoS slowloris	5796		

3.2.2. UNSW-NB15 Dataset

Unlike CICIDS 2017, UNSW-NB15 [51] is not generated from a real environment but comprises actual modern normal activities and synthetic attack behaviors created by IXIA PerfectStorm [52]. It includes nine attacks, including DoS, worms, backdoors and fuzzers. The UNSW-NB15 dataset consists of four files, and each component is provided in Table 4.

Table 4. UNSW-NB15 components.

Filename	No. Benign	No. Malicious
UNSW-NB15_1	677,786	22,215
UNSW-NB15_2	647,252	52,749
UNSW-NB15_3	542,576	157,425
UNSW-NB15_4	351,150	88,894
Total	2,218,764	321,283

Table 5 presents the number of data observations of UNSW-NB15 by type. Among the 2,540,047 data observations, about 87% are normal data and about 13% are attack data.

Table 5. UNSW-NB15 data by type.

Data	No	Data	No
Benign	2,218,764	Reconnaissance	13,987
Generic	215,481	Analysis	2677
Exploits	44,525	Backdoor	2329
Fuzzers	24,246	Shellcode	1511
Denial of Service	16,353	worms	174

Among the attack types in each dataset, only DoS is duplicated. In the case of CICIDS2017, it is not known how similar the attack types are, because DoS attacks are divided into four kinds of attacks. However, the above two datasets have very different attack types, so if the two datasets are combined, they can be used to detect a wide range of attacks. Therefore, when two dataset types are used, it is very useful in evaluating the performance of the IDS in terms of the ability to detect new attack types.

4. Experimental Result

4.1. Experimental Overview

In this section, we measure the performance of the proposed system in terms of the F1-score. First, the measurement elements in the experiment are explained and the proposed system operation process and experimental results are described. The goal of the experiment was to demonstrate that qualitative and quantitative learning is possible, regardless of the type of dataset, using the IDHCS.

4.2. Experimental Environment

We conducted experiments in the following environments to evaluate the IDHCS. The proposed IDHCS was implemented in the Keras [53] environment with a 3.2 GHz CPU (AMD Ryzen 7 2700 8-Core) with a 32 GB RAM, RTX-2080Ti GPU, using Python 3.7.3 and a 64-bit Ubuntu 18.04.2 LTS operating system. All experiments were conducted in python code, and the machine learning algorithm was implanted using Keras.

4.3. Performance Metrics

The F1-score is used as a factor to evaluate the performance of the IDHCS. Accuracy and recall indicators can evaluate prediction rates, but the performance evaluation is unreliable if the data are asymmetric. The ratio between the normal and attack data in the dataset in this experiment is unbalanced: in CICIDS2017, it is 80:20 and in UNSW-NB15, it is 87:13. In the real environment, like datasets, all data have an unbalanced shape. Thus, the F1-score was used to evaluate the model performance more accurately in this study. The F1-score is obtained through Equations (2)–(4) based on Table 6:

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

$$\text{F1 Score} = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{4}$$

Table 6. Confusion matrix.

		Real Answer	
		True	**False**
Classification result	True	True positive	False positive
	False	False negative	True negative

4.4. Experimental Method

The experiment in this study was conducted as depicted in Figure 3. The reinforcement learning agent consists of the PPO. The reinforcement learning environment refers to IDS comprising the DNN-based feature extractor and *k*-means clustering module. The environment is the target that the agent requests for action and it receives the result. The agent sends several features extracted from the DNN feature extractor and the number of clusters to the clustering module as actions. In the environment, the F1-score is reported to the agent by performing feature extraction and clustering based on the actions received from the agent. The agent improves the performance of the IDS through repeated learning policy evaluations and updates.

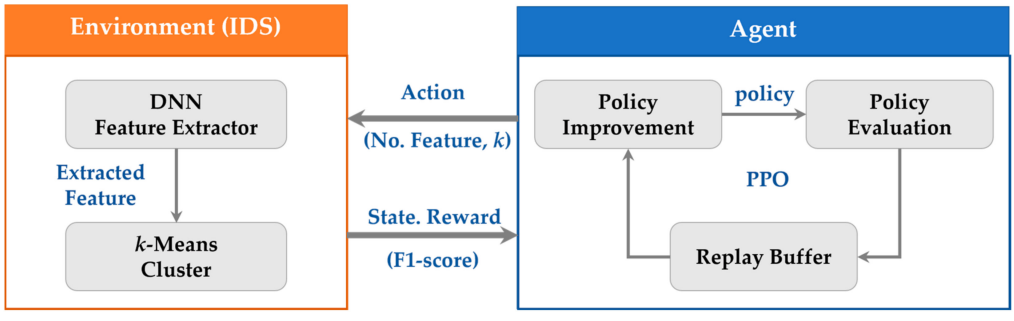

Figure 3. Experimental operation process.

The experiment was repeated in three different environments. In the first experiment, only the CICIDS2017 dataset was used. In the second experiment, only the UNSW-NB15 dataset was used. Finally, the CICIDS2017 and UNSW-NB15 datasets were merged and used in the third experiment. The dataset merger was performed by integrating features extracted from the DNN feature extractor into one file. When the agent passes the new number of features in each experiment, the features are extracted again and an integrated file is produced. Because the two datasets contain different attack types, the two datasets were combined to verify whether the proposed IDS used with the reinforcement learning control algorithm can cover a wide range of attacks. Combining the two datasets is difficult to represent all of the rapidly changing network environments. However, due to the diversity of attack and normal patterns in the two datasets, they appear to be sufficient to exhibit changes in the network environment. The PPO improves the performance by iterating policy improvement, policy evaluation and replay buffers until an appropriate performance is achieved.

4.5. Experimental Evaluation

Table 7 lists the results of experiments using CICIDS2017 and UNSW-NB15. The experiments using the IDHCS resulted in F1-scores of 0.96552 on CICIDS2017 and 0.94268 on UNSW-NB15. In the experiment that merged the datasets, an F1-score of 0.93567 was obtained. Each experiment resulted in better or similar performance compared to other studies [24–28]. Despite the wide range of attack types, mixing the two types of data provided excellent results. Both CICIDS2017 and UNSW-NB15 datasets contain the latest attacks, but they have different characteristics. Different dataset types may perform relatively poorly, as the IDS is trained to classify a specific dataset, as demonstrated in [23]. However, the proposed system exhibited high performance in terms of the F1-score, which suggests the possibility of overcoming the limitations of the IDS due to the dependency on the dataset.

Table 7. Comparative performance verification table.

Reference	Algorithm	Dataset	F1 Score
Our Proposed	PPO + DNN + *k*-means	CICIDS 2017	0.96552
Our Proposed	PPO + DNN + *k*-means	UNSW-NB15	0.94268
Our Proposed	PPO + DNN + *k*-means	CICIDS2017 + UNSW-NB15	0.93567
[24]	C5.0	CICIDS 2017	0.92303
[24]	Naïve Bayes	CICIDS 2017	0.92481
[24]	Random forest	CICIDS 2017	0.88003
[24]	Support vector machine	CICIDS 2017	0.88219
[25]	AE + DNN	CICIDS 2017	0.98570
[26]	AdaBoost + PCA + SMOTE	CICIDS 2017	0.81830
[27]	Decision trees (C5.0)	UNSW-NB15	0.86
[27]	Naïve Bayes	UNSW-NB15	0.824
[27]	Support vector machine	UNSW-NB15	0.755
[28]	ANN	UNSW-NB15	0.92018

Figures 4–6 present the training results for 1000 epochs for each dataset on the IDHCS. Figure 4 displays the results using the CICIDS2017 dataset, revealing a curve in which the F1-score rises from a minimum of 0.86831 to a maximum of 0.96552. Figure 5 depicts the results using the UNSW-NB15 dataset, exhibiting a curve in which the F1-score rises from a minimum of 0.58132 to a maximum of 0.94268. Figure 6 illustrates the results of using the merged dataset and demonstrates that the F1-score rises from a minimum of 0.56436 to a maximum of 0.93567. The dotted red line in each figure is the log trend line. The trend line reveals a gentle upward curve. The F1-score has a range of fluctuations due to the exploration process of reinforcement learning. As time goes by, however, the F1-score continuously increases and the range of fluctuations also decreases, indicating that it is

stabilizing. Due to the nature of PPO, stability is pursued and the policy is updated, so it is expected that a very stable F1-score is obtained when more learning is conducted.

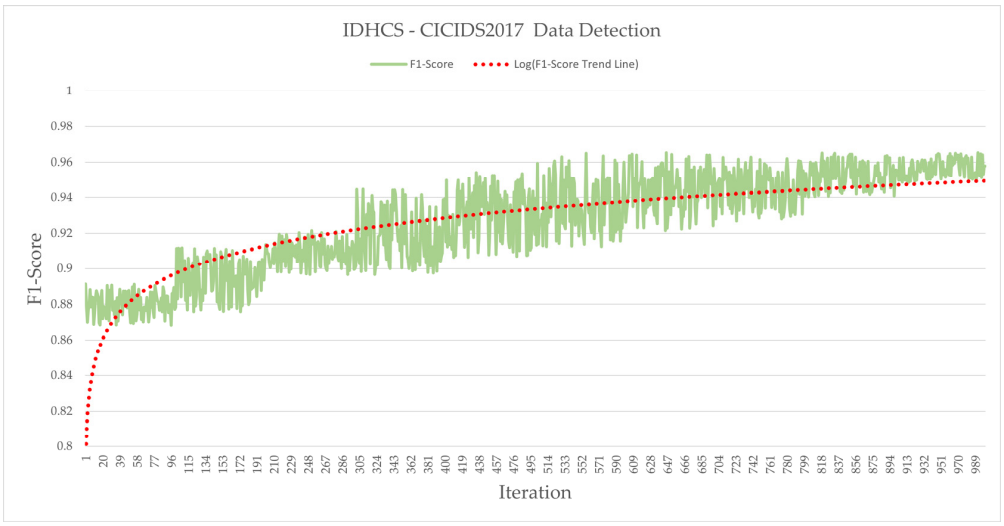

Figure 4. IDHCS—CICIDS2017 data detection.

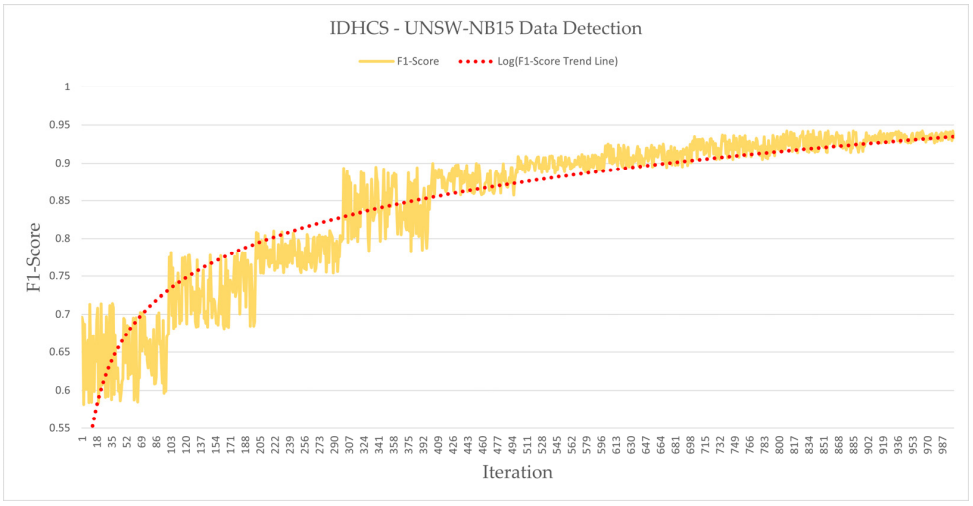

Figure 5. IDHCS—UNSW-NB15 data detection.

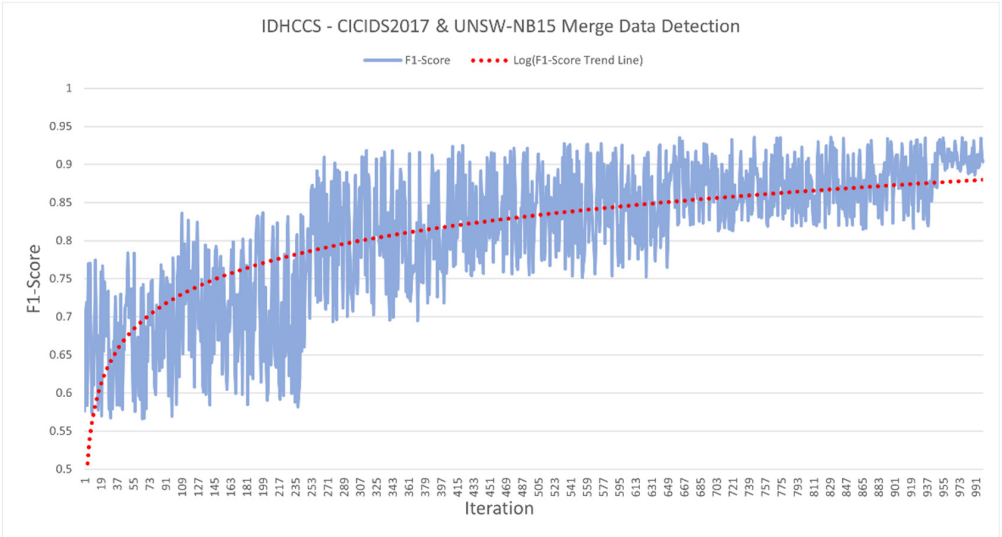

Figure 6. IDHCS—CICIDS2017 and UNSW-NB15 merged data detection.

5. Conclusions

In this study, we proposed an IDHCS that automatically updates the intrusion detection process and improves performance by controlling the hyperparameters of the DNN-based feature extractor and the *k*-means cluster module using the PPO algorithm. The CICIDS2017 and UNSW-NB15 datasets were used individually and merged to conduct the experiment and verify the performance. An F1-score of 0.96552 for CICIDS2017, an F1-score of 0.94268 for UNSW-NB15 and an F1-score of 0.93567 for the merged experiment were obtained, with high F1-scores in all experiments.

Feature engineering is an important factor in intrusion detection data preprocessing. Classifying an intrusion is easy if the datasets are carefully analyzed and their characteristics are well understood. In a real environment, however, it is difficult to quickly analyze all data and extract valuable features of the attack as data characteristics change quickly.

The IDHCS proposed in this paper demonstrates the automation and performance improvement of the IDS by controlling the conditions of a hybrid IDS, such as feature extraction and clustering with the PPO algorithm. The IDHCS exhibited high performance for each dataset and the merged dataset. When a testbed is used, various AI algorithms can be applied and explored to demonstrate a high level of performance. In a real environment, however, more complex situations are encountered. The proposed IDHCS demonstrates the ability to solve these complex problems by iteratively learning through reinforcement learning.

We plan to study a system that considers the automation and stability pursued in this study and fast-processing capability in future work. Although PPO indicates stable policy update ability, there is room for improvement in learning speed. In addition, a more diverse dataset could be considered to reflect a more realistic network environment.

Author Contributions: Conceptualization, H.H., H.K. and Y.K.; methodology, H.H.; software, H.H.; validation, H.H., H.K. and Y.K.; formal analysis, H.H.; investigation, H.H.; resources, Y.K.; data curation, H.H.; writing—original draft preparation, H.H.; writing—review and editing, H.H., H.K. and Y.K.; visualization, H.H. and H.K.; supervision, Y.K.; project administration, Y.K.; All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Dongguk University Research Fund of 2021.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The dataset used in the paper are publicly available to everyone and can be access at: https://www.unb.ca/cic/datasets/ids-2017.html (accessed on 24 November 2021), https://research.unsw.edu.au/projects/unsw-nb15-dataset (accessed on 24 November 2021), for CICIDS2017, and UNSW-NB15 dataset, respectively.

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

References

1. Mukherjee, B.; Heberlein, L.T.; Levitt, K.N. Network Intrusion Detection. *IEEE Netw.* **1994**, *8*, 26–41. [CrossRef]
2. Catania, C.A.; Garino, C.G. Automatic network intrusion detection: Current techniques and open issues. *Comput. Electr. Eng.* **2012**, *38*, 1062–1072. [CrossRef]
3. Shahsavar, M.M.; Akrami, M.; Gheibi, M.; Kavianpour, B.; Fathollahi-Fard, A.M.; Behzadian, K. Constructing a smart framework for supplying the biogas energy in green buildings using an integration of response surface methodology, artificial intelligence and petri net modeling. *Energy Convers. Manag.* **2021**, *248*, 114794. [CrossRef]
4. Ghadami, N.; Gheibi, M.; Kian, Z.; Faramarz, M.G.; Naghedi, R.; Eftekhari, M.; Fathollahi-Fard, A.M.; Dulebenets, M.A.; Tian, G. Implemenutation of solar energy in smart cities using an integration of artificial neural network, photovoltaic system and classical Delphi methods. *Sustain. Cities Soc.* **2021**, *74*, 103149. [CrossRef]
5. Chen, W.H.; Hsu, S.H.; Shen, H.P. Application of SVM and ANN for intrusion detection. *Comput. Oper. Res.* **2005**, *32*, 2617–2634. [CrossRef]
6. Mulay, S.A.; Devale, P.R.; Garje, G.V. Intrusion Detection System using Support Vector Machine and Decision Tree. *Int. J. Comput. Appl.* **2010**, *3*, 40–43. [CrossRef]

7. Mohammed, M.N.; Sulaiman, N. Intrusion Detection System Based on SVM for WLAN. *Procedia Technol.* **2012**, *1*, 313–317. [CrossRef]
8. Niyaz, Q.; Sun, W.; Javaid, A.Y.; Alam, M. A Deep Learning Approach for Network Intrusion Detection System. *EAI Endorsed Trans. Secur. Saf.* **2016**, *3*, e2.
9. Yin, C.; Zhu, Y.; Fei, J.; He, X. A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks. *IEEE Access* **2017**, *5*, 21954–21961. [CrossRef]
10. Tang, C.; Luktarhan, N.; Zhao, Y. SAAE-DNN: Deep Learning Method on Intrusion Detection. *Symmetry* **2020**, *12*, 1965. [CrossRef]
11. Siddique, K.; Akhtar, Z.; Khan, F.A.; Kim, Y. KDD Cup 99 Data Sets: A Perspective on the Role of Data Sets in Network Intrusion Detection Research. *IEEE Comput.* **2019**, *52*, 41–51. [CrossRef]
12. Khan, M.A.; Karim, M.R.; Kim, Y. A Scalable and Hybrid Intrusion Detection System Based on the Convolutional-LSTM Network. *Symmetry* **2019**, *11*, 583. [CrossRef]
13. Tahir, H.M.; Hasan, W.; Said, A.M.; Zakaria, N.H.; Katuk, N.; Kabir, N.F.; Omar, M.H.; Ghazail, O.; Yahya, N.I. Hybrid Machine Learning Technique for Intrusion Detection System. In Proceedings of the 5th International Conference on Computing and Informatics' ICOCI 2015, Istanbul, Turkey, 11–13 August 2015; pp. 1–5.
14. Hsu, Y.F.; Matsuoka, M. A Deep Reinforcement Learning Approach for Anomaly Network Intrusion Detection System. In Proceedings of the 2020 IEEE 9th International Conference on Cloud Networking (CloudNet), Piscataway, NJ, USA, 9–11 November 2020; pp. 1–6.
15. Suwannalai, E.; Polprasert, C. Network Intrusion Detection System Using Adversarial Reinforcement Learning with Deep Q-Network. In Proceedings of the 2020 18th International Conference on ICT and Knowledge Engineering (ICT&KE), Bangkok, Thailand, 18–20 November 2020; pp. 1–7.
16. Li, D.; Zhao, D.; Zhang, Q.; Chen, Y. Reinforcement Learning and Deep Learning Based Lateral Control for Autonomous Driving [application notes]. *IEEE Comput. Intell. Mag.* **2019**, *14*, 89–98. [CrossRef]
17. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M. Playing Atrai with Deep Reinforcement Learning. *arXiv* **2013**, arXiv:1312.5602.
18. Liang, Y.; Machado, M.C.; Talvitie, E.; Bowling, M. State of the Art Control of Atari Games Using Shallow Reinforcement Learning. *arXiv* **2016**, arXiv:1512.01563.
19. Park, K.T.; Son, Y.H.; Ko, S.W.; Noh, S.D. Digital Twin and Reinforcement Learning-Based Resilient Production Control for Micro Smart Factory. *Appl. Sci.* **2021**, *11*, 2977. [CrossRef]
20. Dietterich, T. Overfitting and Undercomputing in Machine Learning. *ACM Comput. Surv.* **1995**, *27*, 326–327. [CrossRef]
21. Alabdulwahab, S.; Moon, B. Feature Selection Methods Simultaneously Improve the Detection Accuracy and Model Building Time of Machine Learning Classifiers. *Symmetry* **2020**, *12*, 1424. [CrossRef]
22. Aghdam, M.H.; Kabiri, P. Feature Selection for Intrusion Detection System Using Ant Colony Optimization. *Int. J. Netw. Secur.* **2016**, *18*, 420–432.
23. Huang, H.; An, S. A Lightweight Intrusion Detection System Based on Feature Selection. In Proceedings of the 2012 Second International Conference on Electric Information and Control Engineering, Washington, DC, USA, 18–20 May 2012; Volume 1, pp. 339–342.
24. Prasad, M.; Tripathi, S.; Dahal, K. An efficient feature selection based Bayesian and Rough set approach for intrusion detection. *Appl. Soft Comput. J.* **2020**, *87*, 105980. [CrossRef]
25. Rababah, B.; Srivastava, S. Hybrid Model for Intrusion Detection Systems. *arXiv* **2020**, arXiv:2003.08585.
26. Abdulrahman, A.A.; Ibrahem, M.K. Evaluation of DDoS Attacks Detection in a CICIDS2017 Dataset Based on Classification Algorithms. *Iraqi J. Inf. Commun. Technol.* **2018**, *1*, 1–7.
27. Bhardwaj, A.; Mangat, V.; Vig, R. Hyperband Tuned Deep Neural Network With Well Posed Stacked Sparse AutoEncoder for Detection of DDoS Attacks in Cloud. *IEEE Access* **2020**, *9*, 181916–181929. [CrossRef]
28. Yulianto, A.; Sukarno, P.; Suwastika, N.A. Improving AdaBoost-based Intrusion Detection System (IDS) Performance on CIC IDS 2017 Dataset. *J. Phys. Conf. Ser.* **2019**, *1192*, 12–18. [CrossRef]
29. Meftah, S.; Rachidi, T.; Assem, N. Network Based Intrusion Detection Using the UNSW-NB15 Dataset. *Int. J. Comput. Digit. Syst.* **2019**, *8*, 478–487.
30. Kanimozhi, V.; Jacob, P. UNSW-NB15 Dataset Feature Selection and Network Intrusion Detection using Deep Learning. *Int. J. Recent Technol. Eng.* **2019**, *7*, 443–446.
31. Yan, B.; Han, G. Effective Feature Extraction via Stacked Sparse Autoencoder to Improve Intrusion Detection System. *IEEE Access* **2019**, *6*, 41238–41248. [CrossRef]
32. Ishaque, M.; Hudec, L. Feature Extraction Using Deep Learning for Intrusion Detection System. In Proceedings of the 2019 2nd International Conference on Computer Applications & Information Security (ICCAIS), Riyadh, Saudi Arabia, 1–3 May 2019; pp. 1–5.
33. Kasongo, S.M.; Sun, Y. A deep learning method with wrapper based feature extraction for wireless intrusion detection system. *Comput. Secur.* **2020**, *92*, 101752. [CrossRef]
34. Krishna, K.; Murty, M.N. Genetic K-Means Algorithm. *IEEE Trans. Syst. Man Cybern. Part B Cybern.* **1999**, *29*, 433–439. [CrossRef]
35. Kind of Reinforcement Algorithms. Available online: https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html (accessed on 18 December 2021).

36. Konda, V.R.; Tsitsiklis, J.N. Actor-Critic Algorithms. In Proceedings of the Advances in Neural Information Processing Systems, Cambridge, MA, USA, 1 January 2000; pp. 1008–1014.
37. Mnih, V.; Badia, A.P.; Mirza, M.; Graves, A.; Lillicrap, T.P.; Harley, T.; Silver, T.; Kavukcuoglu, K. Asynchronous Methods for Deep Reinforcement Learning. *arXiv* **2016**, arXiv:1602.01783.
38. Haarnoja, T.; Zhou, A.; Abbeel, P.; Levine, S. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. *arXiv* **2018**, arXiv:1801.01290.
39. Silver, D.; Lever, G.; Heess, N.; Degris, T.; Wierstra, D.; Riedmiller, M. Deterministic Policy Gradient Algorithms. In Proceedings of the 31st International Conference on Machine Learning, Beijing, China, 21–26 June 2014; Volume 32, pp. 387–395.
40. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2019**, arXiv:1509.02971.
41. Barth-Maron, G.; Hoffman, M.W.; Budden, D.; Dabney, W.; Horgan, D.; TB, D.; Muldal, A.; Heess, N.; Lillicrap, T. Distributed Distributional Deterministic Policy Gradients. *arXiv* **2018**, arXiv:1804.08617.
42. Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, P.; Mordatch, I. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. *arXiv* **2020**, arXiv:1706.02275.
43. Schulman, J.; Levine, S.; Moritz, P.; Jordan, M.; Abbeel, P. Trust Region Policy Optimization. In Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 6–11 July 2015; pp. 1889–1897.
44. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal Policy Optimization Algorithms. *arXiv* **2017**, arXiv:1707.06347.
45. Bohn, E.; Coates, E.M.; Moe, S.; Johansen, T.A. Deep Reinforcement Learning Attitude Control of Fixed-Wing UAVs Using Proximal Policy Optimization. In Proceedings of the 2019 International Conference on Unmanned Aircraft Systems, Atlanta, GA, USA, 11–14 June 2019; pp. 11–14.
46. Vanvuchelen, N.; Gijsbrechts, J.; Boute, R. Use of Proximal Policy Optimization for the Joint Replenishment Problem. *Comput. Ind.* **2020**, *119*, 103239. [CrossRef]
47. Khraisat, A.; Gondal, I.; Vamplew, P.; Kamruzzaman, J. Survey of intrusion detection systems: Techniques, datasets and challenges. *Cybersecurity* **2019**, *2*, 20. [CrossRef]
48. Maseer, Z.K.; Yusof, R.; Bahaman, N.; Mostafa, S.A.; Foozy, C.F.M. Benchmarking of Machine Learning for Anomaly Based Intrusion Detection System in the CICIDS2017 Dataset. *IEEE Access* **2021**, *9*, 22351–22370. [CrossRef]
49. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP), Madeira, Portugal, 22–24 January 2018; pp. 1–8.
50. CICIDS2017, Intrusion Detection Evaluation Dataset. Available online: https://www.unb.ca/cic/datasets/ids-2017.html (accessed on 18 December 2021).
51. Moustafa, N.; Slay, J. UNSW-NB15: A Comprehensive Data Set for Network Intrusion Detection Systems (UNSW-NB15 Network Data Set). In Proceedings of the 2015 Military Communications and Information Systems Conference (MILCIS), Canberra, Australia, 10–12 November 2015; pp. 1–6.
52. PerfectStorm Product Page. Available online: https://www.keysight.com/us/en/products/network-test/network-test-hardware/perfectstorm-one.html (accessed on 18 December 2021).
53. Keras the Python Deep Learning API Home Page. Available online: https://keras.io (accessed on 18 December 2021).