symmetry

MDPI

Article

Correlation between Deep Neural Network Hidden Layer and Intrusion Detection Performance in IoT Intrusion Detection System

Hyojoon Han [1], Hyukho Kim [2] and Yangwoo Kim [1,*]

[1] Department of Information and Communication Engineering, Dongguk University, Seoul 04620, Korea
[2] Data and Information Technology Center, Samsung Electronics, Hwaseong-si 18448, Korea
* Correspondence: ywkim@dongguk.edu; Tel.: +82-2-2260-3821

Abstract: As the Internet of Things (IoT) continues to grow, a vast amount of data is generated. The IoT environment is quite sensitive to security challenges because personal information may be leaked or sensor data may be manipulated, which could cause accidents. Because traditional intrusion detection system (IDS) studies are often designed to work well on datasets, it is unknown whether they would work well in a changing network environment. In addition, IDSs for protecting IoT environments have been studied, but their performance was verified using datasets unrelated to the IoT, so it is not known whether the performance would be effective in an IoT environment. In this study, we propose an intrusion detection hyperparameter control system (ID-HyConSys) that automates the IDS using proximal policy optimization (PPO) to solve these problems and reliably protect the IoT environment. ID-HyConSys consists of an intrusion detection module consisting of a deep neural network (DNN) feature extractor that extracts efficient features from a changing network environment, a *k*-means cluster that clusters the extracted data, and a PPO agent that automates the IDS through learning and control. Through experimentation, it was confirmed that the hidden layer configuration, the number of feature extractions by the DNN feature extractor, and the number of clusters in the *k*-means cluster significantly affected the intrusion detection performance. The PPO directly controls these hyperparameters and determines the optimized value itself. The performance of ID-HyConSys was evaluated using the CICIDS2017 and MQTTset datasets. An F1-score of 0.9707 on CICIDS2017 and an F1-score of 0.9973 on the MQTTset were obtained. Finally, we merged the two datasets and obtained an F1-score of 0.9901. The superiority of the ID-HyConSys proposed in this study was confirmed because ID-HyConSys showed high performance on each dataset and, at the same time, very high performance on complex merged datasets. ID-HyConSys is expected to protect the IoT environment more quickly and safely by automatically learning network changes and adjusting the intrusion detection module accordingly.

Keywords: network intrusion detection system (NIDS); Internet of Things (IoT); deep learning (DL); deep neural network (DNN); *k*-means; proximal policy optimization (PPO); CICIDS2017; MQTTset

check for updates

Citation: Han, H.; Kim, H.; Kim, Y. Correlation between Deep Neural Network Hidden Layer and Intrusion Detection Performance in IoT Intrusion Detection System. *Symmetry* **2022**, *14*, 2077. https://doi.org/10.3390/sym14102077

Academic Editor: Peng-Yeng Yin

Received: 5 September 2022
Accepted: 4 October 2022
Published: 6 October 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 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 (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

As network-based large-capacity services, such as cloud services and over-the-top (OTT) services become diversified, a large amount of network data is being generated. In addition, network traffic is being generated by heterogeneous devices across a variety of industrial fields, such as smart farms, smart factories, and smart homes. Network attack technology is also developing day by day. The performance requirements of a network intrusion detection system (IDS) [1] are becoming more complex due to the heterogeneous equipment used, protocols such as TCP/UDP, IoT protocols such as constrained application protocol (CoAP), message queuing telemetry transport (MQTT), and the diversity of attacks.

IDSs are divided into misuse-based IDSs that detect an intrusion by directly matching a specific value of malicious data in a packet, and abnormal behavior-based IDSs that detect anomalies by analyzing traffic patterns [2].

Recent research on IDSs based on abnormal behaviors has been conducted to detect outliers by learning intrusion detection datasets in combination with machine learning (ML) techniques. In the early research on IDSs that applied ML, intrusion detection methods used algorithms, such as *k*-nearest neighbor (*k*-NN), support vector machine (SVM), self-organizing map (SOM), and decision trees (DT) [3–7]. These methods showed efficient performance on early, low-complexity intrusion detection datasets but gradually began to show limitations as cyber-attacks evolved and their complexity increased. Since then, research on IDSs has used a hybrid or ensemble-type classifier that combines two or more algorithms, such as convolutional long short-term memory and SparkML, or genetic algorithms and DT to detect intrusion data with high complexity [8–10].

Most of the above studies are based on intrusion detection datasets and sometimes show dataset-dependent results. To improve intrusion detection performance, efficient features are extracted and used for each attack [11], or as in [12,13], fixed features offer inferior performance on a specific attack. In addition, because these studies are focused on a specific intrusion detection dataset, they show inferior performance when using other datasets [14].

Because the general-purpose IDS environment receives a variety of data, and the network environment changes rapidly, very high false-positive and false-negative rates may occur when learning is carried out on a single dataset. Moreover, in the case of the IoT environment, it may not be appropriate to evaluate a system using an existing intrusion detection dataset because heterogeneous devices, various sensors, and a protocol different from that of the existing network environment are used.

To solve the above problem, we presented the intrusion detection hyperparameter control system (ID-HyConSys), which is an IDS that can respond flexibly to changing network environments by learning and adjusting the hyperparameters of the deep neural network (DNN) feature extractor and the *k*-means cluster through the proximal policy optimization (PPO) reinforcement learning agent in our previous study [15]. The DNN feature extractor extracts a factor O, called new features, from the input data. In the case of the existing DNN, the hidden layer model is strengthened by using the error-backpropagation function, depending on whether the output feature value is correct or not. However, in the case of the DNN feature extractor, it is difficult to improve the model through functions such as error-backpropagation, because newly generated data are used in learning. Thus, it cannot immediately be known whether the extracted features are correct or not. Therefore, iterative learning is performed through PPO to find an optimized model, and an intrusion is detected using a *k*-means cluster. The DNN feature extractor is used similarly to the auto-encoder and generative adversarial network (GAN) to generate data.

In deep learning, constructing the hidden layer is quite complex and challenging. Moreover, it is not yet known how the hidden layer can be optimized. Therefore, a model that shows good performance can be found through iteration. Furthermore, finding the hidden layer within a short period is challenging because the hidden layer is composed of various factors such as the number of layers, the number of nodes, and the activation function. Therefore, to respond to the changing network environment in an IDS, it is necessary to consider any changes in the hidden layer.

In this study, we propose ID-HyConSys, which improves the previously announced ID-HyConSys. First, the improved ID-HyConSys considers the adjustment in the hidden layer model of the DNN feature extractor. To verify that the hidden layer has an impact on intrusion detection performance, we examine the correlation between the change in the hidden layer model and the intrusion detection performance. The experiment verified that the change in the hidden layer model has an enormous influence on intrusion detection performance. In the existing PPO, only the number of feature extractions by the DNN feature extractor and the number of *k*-means clusters were adjusted. However, the new

PPO also adjusts the hidden layer model of the DNN feature extractor. The DNN feature extractor in this study builds a robust IDS by creating a new feature that can easily detect an intrusion, just like the GAN generates new data. In addition, the CICIDS2017 [16] and MQTTset [17] datasets are used to evaluate the performance of the IDS. Although many existing IoT studies have published an IDS suitable for IoT security by modifying the systematic approach, such as establishing a lightweight attack detection strategy to detect DoS attacks that the IoT is vulnerable to, studies have often used existing intrusion detection datasets for performance evaluation [18–21]. Because existing IDS studies are dataset-dependent, it may be inappropriate to evaluate performance using only existing datasets that are unrelated to the IoT. This study evaluates a system performance on MQTTset, which is composed of CoAP and MQTT protocols and a variety of sensor data. Therefore, it is possible to build an IDS that can operate not only in a wired network environment, but also in an IoT environment. Furthermore, by evaluating performance using a merged dataset from CICIDS2017 and MQTTset, we show that responding to various network environments is possible for a general-purpose IDS.

The structure of this paper is as follows. Section 2 examines IDS studies using ML and deep learning models. Next, Section 3 examines the structure of the proposed system, and the changes that occur in intrusion detection performance as the deep learning models are changed. Section 4 evaluates the performance of the proposed system. Finally, Section 5 presents conclusions drawn from the experiments.

2. Relational Works

Section 2 examines existing IDS studies and the activation function used to construct the deep learning hidden layer.

2.1. ML for IDS

Recent network attacks have high complexity; there are many types of attacks, and they present in a pattern that is very similar to normal data. In addition, although the IoT system network is similar to the network of a general system, it is different from the existing network environment in that it uses protocols such as CoAP and MQTT, and it generates a lot of sensor data. In order to respond to network attacks, it is necessary not only to include the latest attack types, but also to verify a system's performance using a dataset that is suitable for the IoT environment. CICIDS2017 is a representative intrusion detection dataset that includes the latest attack types. In many studies, CICIDS2017 has been used to evaluate a system's performance and verify its superiority. However, because CICIDS2017 was created in a general network environment, it does not include elements of the IoT environment. It is necessary to evaluate the performance of an IoT IDS on a dataset created in the IoT environment, such as MQTTset.

Some studies have analyzed these two datasets using various ML algorithms. Manimurugan et al. [22] proposed an IDS based on the deep belief network (DBN) algorithm to protect the Internet of Medical Things environment. After the IDS was trained by creating an optimal DBN structure using the roulette technique, the system's performance was evaluated using the CICIDS2017 dataset to obtain an F1-score of 0.97. Although the DBN algorithm was utilized well and an optimal model was presented, there was a problem with data suitability because only the CICIDS2017 dataset was used in the Internet of Medical Things environment.

Yang et al. [23] proposed a multi-layer hybrid IDS that combined signature-based and anomaly-based IDS for vehicle network security. The performance was verified using the controller area network (CAN)-intrusion-dataset, which is a dataset that is suitable for the vehicle network. The verification results were an F1-score of 0.963 on the CAN intrusion dataset and an F1-score of 0.8 on the CICIDS2017. The multi-layer hybrid IDS showed suitability for the vehicle network, as fast intrusion detection time is required due to the characteristics of the vehicle network. However, the multi-layer hybrid IDS showed poor performance in CICIDS2017, so it is unsuitable for systems other than vehicle networks.

Vaccari et al. [17] produced an intrusion detection dataset consisting of sensor data, such as lighting, temperature, humidity, carbon dioxide, and motion data, using CoAP and MQTT protocols that are mainly used in IoT environments for IoT security. Vaccari presented basic performance measurement results using algorithms such as the neural network, random forest, and naïve Bayes. The result for the random forest algorithm showed that the best performance achieved an F1-score of 0.914. This performance result became a benchmark for other researchers who sought to improve IoT IDS performance on MQTTset.

Rachmadi et al. [24] proposed an IDS based on adaptive boosting (AdaBoost) [25] to efficiently detect denial of service (DoS) attacks in the IoT environment. A performance evaluation was performed on MQTTset, and an F1-score of 0.9572 was obtained. Only significant features selected through feature selection were used to improve detection performance, and good performance was obtained by optimizing the AdaBoost model. However, because the method in [24] focuses only on DoS attacks, there is room for improvement because it still shows vulnerability to other attacks.

2.2. Activation Function

The activation function is a function that converts the weighted sum of input data from the deep learning network into a non-linear output signal. The IDS proposed in this study responds to the changing network environment by changing the DNN internal activation function that newly creates the characteristics of the intrusion detection data. The values fed into the activation function node are passed to the next layer through the non-linear function and are used to express the final output value. In a deep learning network where nodes are made of only linear data, there is no effect from stacking layers, so a non-linear function must be used. The activation functions used for the experiments in this study are as follows.

2.2.1. Sigmoid

The sigmoid function is also called the logistic function. It was first used to obtain non-linear values in a linear multi-layer perceptron (MLP), and it is the most-used activation function. The sigmoid function is as shown in Equation (1).

$$sigmoid(x) = \sigma = \frac{1}{1 + e^{-x}} \tag{1}$$

The domain is the whole real number domain, but the function returns a finite value in the finite space (a, b). The sigmoid function usually returns as a value between 0 and 1. As the absolute value of the domain increases, the derivative value of the sigmoid function converges to 0. Therefore, gradient vanishing may occur, in which weights are not updated and are lost.

2.2.2. TanH

The hyperbolic tangent function (TanH) is a hyperbolic function that can be obtained by transforming the sigmoid function, as shown in Equations (2)–(4).

$$tanh(x) = 2\sigma(2x) - 1 \tag{2}$$

$$tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{3}$$

$$tanh'(x) = 1 - tanh^{2}(x) \tag{4}$$

The TanH function moves the center point of the function to 0 to solve the problem of slowing down during the optimization of the sigmoid function. Still, the problem of gradient vanishing, in which the differential value is lost above a certain value of the differential function, remains.

2.2.3. ReLU

The rectified linear unit (ReLU) [26] is the most-used activation function. It is a function that solves the gradient vanishing problem of the sigmoid and TanH function. ReLU is obtained by Equation (5).

$$ReLU(x) = \max(0,\ x) \tag{5}$$

ReLU is a straight line having a slope of 1 when x is greater than 0, and a function value of 0 when x is less than 0. This causes the disadvantage that neurons can die at values less than 0. However, relative to sigmoid and TanH, its learning is fast, computational cost is low, and implementation is straightforward.

2.2.4. ELU

The exponential linear unit (ELU) [27] is an activation function that includes all the advantages of ReLU. However, unlike general ReLU, there is a cost to calculating the exponential function. ELU is obtained by Equation (6).

$$ELU_{\alpha}(x) = \begin{cases} \alpha(\exp(x) - 1), & if\ x < 0 \\ x & , if\ x \geq 0 \end{cases} \tag{6}$$

In ELU, a negative value occurs when the value of x is less than zero, so the activation function's average output is close to zero. This alleviates the gradient vanishing problem. Furthermore, even if the value of x is less than 0, the ELU does not output 0, so it does not create dead neurons. However, it is slower than ReLU because it has to perform exponential calculations.

2.2.5. SELU

Scaled ELU (SELU) is a variant of the ELU function. Klambauer et al. [28] argued that if only fully connected layers are stacked to create a neural network, and all hidden layers use SELU functions, the network is self-normalized. During training, the output of each layer tends to maintain a mean of 0 and a standard deviation of 1, which prevents gradient vanishing and exploding problems. This function often shows superior performance relative to other activation functions. SELU is obtained by Equation (7).

$$SELU(x) = \lambda \begin{cases} x & , if\ x > 0 \\ \alpha e^{x} - a & , if\ x \leq 0 \end{cases} \tag{7}$$

3. ID-HyConSys for IoT

In this section, we look at the system that improved the ID-HyConSys of [15], which we previously proposed to protect the IoT environment. In addition, the structure of ID-HyConSys and the change in intrusion detection performance according to the structure of the hidden layer will be examined.

3.1. ID-HyConSys for IoT

This study proposes an improved ID-HyConSys to protect the IoT environment. Figure 1 shows the structure of a system that uses ID-HyConSys in the IoT environment. A centralized IDS structure in ID-HyConSys is adapted to prevent any overload of the IoT server that is conducting the intrusion detection work. In addition, Apache Kafka is introduced to process data to overcome development difficulties stemming from the heterogeneity of equipment and the expansion of the IoT.

The operation structure of ID-HyConSys is shown in Figure 2. ID-HyConSys consists of a DNN feature extractor, a preprocessing stage that refines packet data and performs feature extraction; a *k*-means cluster that determines intrusion by clustering data; and a PPO reinforcement learning agent that controls the DNN feature extractor and *k*-means cluster to improve the system performance. In addition, the reinforcement learning agent controls

the hyperparameters of the DNN feature extractor and *k*-means cluster and receives an F1-score as a reward for updating policies and hyperparameters to improve intrusion detection performance.

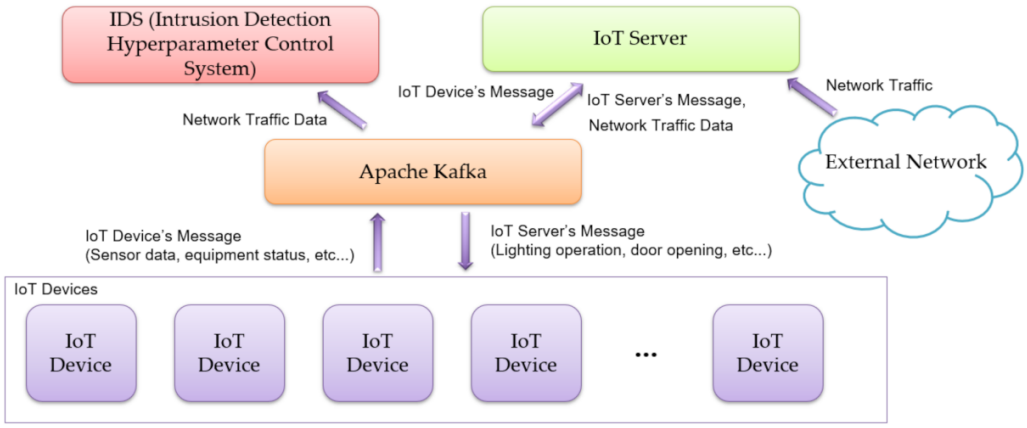


Figure 1. Intrusion detection system structure for IoT security.

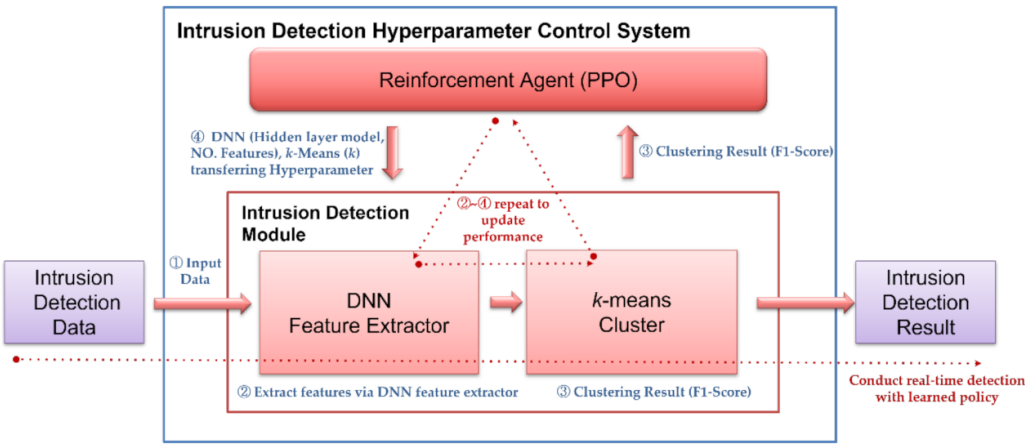


Figure 2. Intrusion detection hyperparameter control system operation process.

ID-HyConSys is composed of a learning mode and a detection mode. In learning mode, performance continuously increases based on the detection result through the reinforcement learning agent. The detection mode determines the hyperparameters of the DNN feature extractor and *k*-means cluster based on the results obtained in the learning mode, and it detects intrusions from the input data. ID-HyConSys uses the learning mode and the detection mode at the same time, and in general, after learning enough to achieve stable performance in the learning mode, it uses the detection mode. In addition, to cope with the changing network, the learning mode continuously learns data, and as the accumulated data increase, the overall performance of the IDS improves.

Because the detection mode processes data using the hyperparameters determined in the learning mode, intrusions can be detected in real-time. The data input to the IDS is moved in real time to the DNN feature extractor and *k*-means cluster to derive the results. Because this resulting value is transmitted to both the network manager and the reinforcement learning agent, continuous learning is possible in the learning mode. Furthermore, the network manager also checks the intrusion detection result in real time so that the system can be operated stably.

Figure 3 shows the detailed structure of the intrusion detection module. The detailed operational process of each module is as follows.

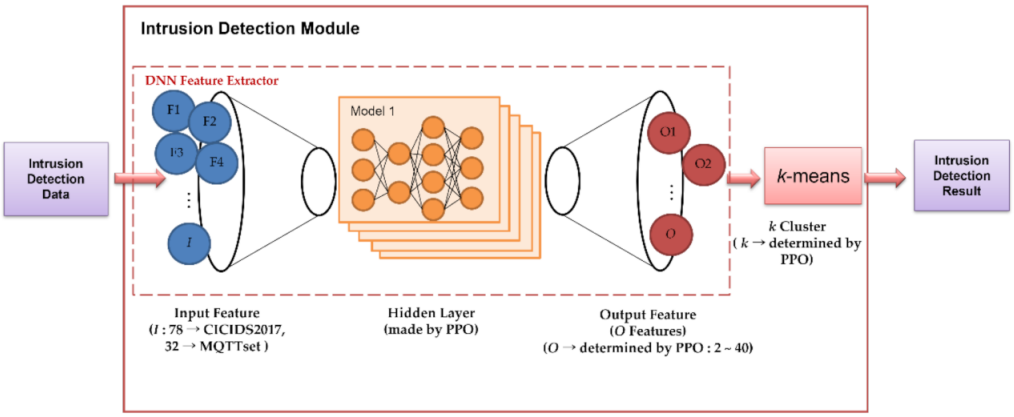


Figure 3. Intrusion detection module details.

3.1.1. DNN Feature Extractor

The DNN feature extractor works to refine the data to cluster the incoming packets. The intrusion detection dataset consists of a large number of features. When intrusion detection is performed using all the features, performance may deteriorate due to overfitting problems. Furthermore, as in [11], it may be challenging to cover all attack patterns through feature selection because each attack has different easy-to-detect feature groups. Therefore, in this study, existing features are combined and used through feature extraction. It has been proven through many studies that intrusion detection using feature extraction result in better performance than using all features [29–31]. However, the number of features that need to be extracted differs from situation to situation. Therefore, the reinforcement learning control algorithm controls the above feature extraction hyperparameter, and the features appropriate to the case are extracted.

The DNN feature extractor extracts features through the DNN. The reinforcement learning algorithm adjusts the *O* output and hidden layers according to the learning results, and it extracts the optimal features for intrusion detection. The number of features that are input to the DNN feature extractor varies, depending on the dataset. The dataset used in this study, CICIDS2017, has 78 features, and MQTTset has 32 features. The input data go through the DNN model and come out as the *O* output. The *O* value is determined by reinforcement learning. A clustering algorithm uses the extracted features to determine the intrusion.

Unlike the DNN algorithm used in previous IDS studies, the DNN feature extractor that is used and proposed in this study does not obtain intrusion detection results. Extracting the intrusion detection result is the role of the *k*-means cluster. The DNN feature extractor generates the data to be clustered by the *k*-means clustering algorithm. The reinforcement learning algorithm adjusts the hidden layer model and the number of output features of the DNN feature extractor. It generates data in the form that is most suitable for the network environment from the input network data.

Existing IDS studies show poor performance on data having other characteristics, such as DoS attacks and malware, because they design models for detection on a specific dataset or within specific attack range. The authors of [12] built an IDS using random forest and evaluated performance using the Kyoto 2006+ dataset. Most of the data were detected with high performance. Still, the F1-score for the normal data, which was the highest percentage, was only 0.72. The F1-score for the shellcode attack was 0.16, which indicates unsatisfactory performance. Even if an intrusion is detected with a high probability for specific attacks, if some attacks are rarely detected, this is a considerable vulnerability, so it is necessary to develop a method that responds to a variety of attacks. In addition, in [13], an IDS was constructed through a hybrid learning model that detects triangular-shaped adjacent nodes by partially modifying the *k*-NN algorithm. The result of detection on the KDDCUP-99 dataset showed that normal data achieved good performance with an accuracy greater than

97%, and DoS and probe attacks with an accuracy of over 90%. However, the return-to-libc (R2L) attack showed an accuracy of 80%, and the user-to-root (U2R) attack achieved an accuracy of 60%, confirming that the system is focused on a specific attack.

The problem with these systems is that they are very unlikely to detect new types of attacks. We propose a DNN feature extractor to extract the most effective features when the network environment changes to increase the detection probability for new attack types. The DNN feature extractor finds an optimal model by changing the number of feature extractions and the configuration of the hidden layer using the reinforcement learning agent.

3.1.2. *k*-Means Cluster

In the *k*-means clustering intrusion detection module, attack data and normal data are clustered through the features extracted from the DNN. *k*-means is an algorithm that analyzes the characteristics of features and classifies them into *k* clusters according to the similarity of the data. Assuming that each central point between the attack and normal data is well established through continuous learning, it is possible to quickly determine an attack and normality when new data come in.

This study aims to quickly respond to the rapidly changing network environment. In other supervised learning-based systems, intrusions are detected based on learned network patterns. Such a system may be suitable for a single system where the network environment does not change rapidly, such as an IDS dedicated to a web or search server. However, in this study, similar data are clustered using the unsupervised learning *k*-means clustering algorithm to cope with the rapidly changing network environment. The most important task in *k*-means is to set the number of clusters. IDS data can be classified into attack and normal data. However, various patterns exist in both attack and normal data. Normal activities, such as database access and log search, and attack activities, such as port scan and DoS, have different characteristics. Suppose there is an attack pattern between different normal patterns: in this case, an attack cannot be detected when there are two clusters. Therefore, it is necessary to find the optimal number of clusters and to finely classify the attack and normal patterns according to the network environment.

As the number of clusters increases, detecting an intrusion may increase because the data can be more accurately classified, but not in all situations. In addition, it is difficult to determine the optimal number of clusters because the detection time can increase significantly as the number of clusters increases.

3.1.3. Intrusion Detection Hyperparameter Control

The intrusion detection hyperparameter control module proposed in this study is a method that enables automation and performance improvement for the IDS by controlling the feature extraction and clustering method using PPO [32]. The PPO agent improves performance by iteratively performing data sampling through interactions with the environment and optimizing the objective function surrogate using stochastic gradient ascent. Additionally, unlike the existing method of performing performance updates for every data sampling, a method of updating based on the minibatch size is introduced. Algorithm 1 is a PPO method that improves the performance of ID-HyConSys in this study.

3.2. Deep Neural Network Hidden Layer and Intrusion Detection

The ID-HyConSys proposed in this study detects the intrusion by clustering the features extracted through the DNN feature extractor. We examine the change in intrusion detection performance when the number of extracted features and clustering change while the deep learning hidden layer model is fixed or when the hidden layer model changes while the number of features and clustering are fixed.

Algorithm 1. Update ID-HyConSys policy using PPO.

Input: ***D*//**Intrusion Detection Dataset (CICIDS2017 or MQTTset or both)
Begin
Initialize: ***O*//**The number of features to extract.
Initialize: ***H*//**The DNN's hidden layer model
Initialize: ***K*//**The number to cluster.
for iteration = 1,2, ... do//Run policy in IDS environment for 2000 timesteps
  $\boldsymbol{F}(\mathbf{F}_0, \mathbf{F}_1, \ldots, \mathbf{F}_{\mathbf{o}-1})$ **= DNNFeatureExtractor(*D*, *O*, *H*)//**A extracted features by DNN.
  S **= KmeansCluster(*F*, *K*)//**F1 score obtained by k -means Cluster
  O*, *H*, *K **= UpdatePolicy(*S*)//** $\theta_{old} \leftarrow \theta$
end for
end;

3.2.1. Correlation between Feature Extraction and Intrusion Detection

We examine the effect of the number of extracted features on intrusion detection performance when the deep learning model is fixed.

Table 1 shows the results of measuring performance by creating a random model for the ID-HyConSys proposed in this study. All three models use the same number of hidden layers and functions, the number of clusters is the same, and only the number of features to be extracted changes. When 19 features were extracted and clustered by the DNN feature extractor, the F1-score was 0.9205. When 10 features were extracted, the F1-score was 0.8990, and when 36 features were extracted, the F1-score was 0.8949. It can be confirmed that there is a performance change according to the number of extracted features.

Table 1. Number of feature extraction and intrusion detection performance.

CICIDS2017	Model 1	Model 2	Model 3
No. Input Features	78	78	78
No. Extracted Features	19	10	36
No. Hidden Layers	7	7	7
Activation Function	ReLU	ReLU	ReLU
k-means Clusters	32	32	32
F1-score	0.9205	0.8990	0.8949

Performance degradation generally occurs when the number of features is excessively large or small. Finding an optimal value is important because a significant performance change can be confirmed, even if only the number of features is adjusted.

3.2.2. Correlation between Number of Cluster and Intrusion Detection

In the state where the deep learning model and the number of extracted features is fixed, we examine the effect of the number of clusters on intrusion detection performance.

Table 2 shows the performance results when the number of clusters in the same DNN model is changed. As the number of clusters increases, the boundaries between the data become tighter, and more fine-grained classification is possible, so performance improves. In the case of Model 1 and Model 2, the number of clusters increases from 8 to 128, and the performance improves by approximately 5%. However, the increase in the number of clusters does not unconditionally improve performance. In the case of Model 3 and Model 4, as in the previous case, the number of clusters increases from 8 to 128, but the difference in performance gain is insignificant.

Therefore, it is important to find the optimal number of clusters through continuous learning while controlling the number of clusters in the clustering algorithm using the PPO algorithm.

Table 2. Number of clustering and intrusion detection performance.

CICIDS2017	Model 1	Model 2	Model 3	Model 4
No. Input Features	78	78	78	78
No. Extracted Features	21	21	10	10
No. Hidden Layers	10	10	10	10
Activation Function	ReLU	ReLU	ReLU	ReLU
k-means Clusters	8	128	8	128
F1-score	0.8906	0.9404	0.8906	0.8987

3.2.3. Correlation between Hidden Layer Configuration and Intrusion Detection

In the state where the number of extracted features and the number of clusters were fixed, the effect of changes in the hidden layer configuration on intrusion detection performance was examined.

Table 3 shows the result of measuring the performance of the ID-HyConSys proposed in this study by changing only the configuration of the hidden layer under the same hyperparameters for the number of extracted features and k-means clusters. Even if the same number of identical features are extracted, it can be seen that the intrusion detection performance also changes when the number of hidden layers changes or the activation function changes.

Table 3. Hidden layer configuration and intrusion detection performance.

CICIDS2017	Model 1	Model 2	Model 3
No. Input Features	78	78	78
No. Extracted Features	10	10	10
No. Hidden Layers	3	3	6
Activation Function	TanH	SELU + ELU	ReLU
k-means Clusters	32	32	32
F1-score	0.9205	0.8990	0.8989

Because there is no optimized method for constructing the hidden layer, a model is generally created through many iteration experiments. In the case of executing intrusion detection in a constrained setting, such as a web server-only IDS or a database-only IDS, it may be beneficial to find and operate one optimized model. However, when the available resources are limited, such as in the IoT environment or small- and medium-sized enterprises, processing various kinds of data with one IDS is necessary. Furthermore, it is necessary to process data quickly to detect the complexity of various data and new cyber-attack patterns. The network speed is too fast for existing IDSs to process and analyze data individually. The types of attacks are becoming more diverse. To respond quickly in such a situation, ID-HyConSys automatically adjusts the hidden layer through reinforcement learning and responds to intrusion.

4. Experimental Result

4.1. Experimental Overview

In this section, we first describe the experimental data and experimental measures used in the experiment to evaluate the performance of the system proposed in this study. The system's performance is analyzed by examining the system's operational process and experimental results.

4.2. Experimental Dataset

The intrusion detection dataset that is used to evaluate the performance of the IDS proposed in this study is reviewed. Several types of publicly available intrusion detection datasets are used as elements for evaluating IDSs. Datasets can be classified according to whether they contain complete packets, real data, zero-day attacks, or modern attacks [33].

The KDDCUP-99 and NSL-KDD datasets produced by DARPA are widely used to study abnormal behavior IDSs [34]. The above datasets have had an enormous impact on IDS research. However, after more than 20 years of publication, these datasets are not suitable for system evaluation due to their outdated attack types, lack of attack data complexity, and lack of attack data diversity [35]. The University of New Brunswick (UNB) has also published datasets, such as CICIDS2017, which contain the latest attacks on the grounds that older datasets are not suitable for use due to their diversity in traffic and limitations on volume, the lack of anonymized packet information and payloads, and the limitations in the variety of attacks [16].

Although many protocols are used for data communication in the IoT environment, market research shows that the MQTT protocol is the most frequently used to communicate data [36]. Many IoT IDS studies have shown various systematic approaches to IoT security. Still, performance measurement showed that there were problems with reliability when old datasets that did not relate to the IoT were used [18–21]. Therefore, to reliably measure the performance of IoT IDSs, it is necessary to include the MQTT dataset in the experiment.

In this study, we conduct experiments using two types of datasets, CICIDS2017 and MQTTset. CICIDS2017 is a dataset created with a reliable benchmark and has been used in numerous existing IDS studies. MQTTset is the latest dataset created with MQTT that is intended for IoT IDSs. Experiments were performed using each dataset independently. In addition, the datasets were merged and evaluated to confirm that the reinforcement learning algorithm was trained well enough to judge intrusion according to changes in the network environment. The characteristics of each dataset are as follows.

4.2.1. CICIDS2017

UNB released CICIDS 2017 to compensate for the weaknesses of the previously announced intrusion detection dataset, the lack of data volume and attack diversity, and the inability to include the latest attacks. CICIDS2017 contains seven attack types: brute force, DoS, heartbleed, web attack, infiltration, botnet, and distributed DoS (DDoS), according to the latest network trends. CICIDS2017 was based on 10 criteria for data reliability [37].

Table 4 lists the number of data samples for CICIDS2017 by type. Of the 2,830,743 data samples, about 80% were normal, and about 20% were attack types. In addition, because the seven attack types consist of 14 specific attack types, the dataset is suitable for judging the performance of IDSs using various attack types.

Table 4. CICIDS2017 data by type.

Data	No. Sample	Data	No. Sample
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		

4.2.2. MQTTset

The MQTT protocol is one of the most frequently used protocols in the IoT. Although many IoT IDS studies have been conducted, due to the absence of an IoT-specific dataset, the intrusion detection datasets used in general network IDS research, such as KDDCUP-99, NSL-KDD, and CICIDS2017, have generally been used. However, various IoT-based intrusion detection datasets have recently been released due to the increase in the number of IoT networks. In this study, system performance is measured using the MQTTset dataset, which is the most-used IoT dataset.

MQTTset was built using IoT-Flock [38], which is a network traffic generation tool that can emulate IoT devices and networks based on MQTT and CoAP protocols. It uses sensors for temperature, light intensity, humidity, motion, smoke, door opening, fan status, etc., to communicate data according to 10 scenarios, and it contains normal and attack data.

Table 5 lists the number of data observations in MQTTset by type. The SlowITe attack is a new type of DoS attack that was created using the MQTT vulnerability in IoT [39].

Table 5. MQTTset data by type.

Data	No. Sample	Data	No. Sample
Legitimate	11,915,716	Malformed	10,924
DoS	130,233	SlowITe	9202
Brute Force	14,501	Flood	613

The attack types in the two datasets have very different forms. Because the kinds of attacks between datasets are quite different, the above datasets can be combined to detect a wide range of attacks. In addition, unlike the CICIDS2017 dataset, which is mainly composed of TCP or UDP protocols, the MQTTset dataset consists of MQTT and CoAP protocols, so a greater variety of protocols can be learned and used to detect a greater diversity of attacks. Therefore, using a merged dataset does not guarantee that new types of attacks will be detected. However, high performance on complex datasets may lead to the detection of new types of attacks. As such, a merged dataset is very useful for evaluating the performance of an IDS.

4.3. Performance Metrics

The F1-score is used to evaluate the performance of the proposed system, ID-HyConSys. The prediction rate can also be evaluated using accuracy and recall indicators. However, if the data ratio is asymmetric, this reduces the reliability of the performance evaluation. The ratio of the normal and attack data in the dataset used in this experiment is quite unbalanced, with 80:20 in CICIDS2017 and 98:2 in MQTTset. Table 6 shows the data ratios for CICIDS2017, MQTTset, and the merged dataset.

Table 6. Data ratio by dataset.

No. Sample	CICIDS2017	MQTTset	Merged Dataset
Total	2,830,743	12,081,189	14,911,932
Normal	2,273,097	11,915,716	14,188,813
Attack	557,646	165,473	723,119
Ratio	80:20	98:2	95:5

Like datasets, even in real environments, normal data are asymmetric. Therefore, the F1-score is used in this study to evaluate model performance accurately. The F1-score is obtained from the data in Table 7 and Equations (8)–(10).

Table 7. Confusion matrix.

		Real Answer	
		True	**False**
Classification Result	True	True Positive	False Positive
	False	False Negative	True Negative

$$Recall = \frac{TP}{TP+FN} \quad (8)$$

$$Precision = \frac{TP}{TP+FP} \quad (9)$$

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (10)$$

4.4. Experimental Environment

To evaluate the ID-HyConSys proposed in this study, an experiment was performed in the following environment. The proposed ID-HyConSys was implemented on the Ubuntu 18.04 operating system and written in Python 3.7.3 in an environment with 32 GB RAM, an RTX 2080 Ti GPU, and a 3.2 GHz CPU (AMD Ryzen 7 2700 8-Core). All experiments were written in Python code, and the ML algorithm was implemented using the Keras API [40].

4.5. Experimental Method

The experiment was performed as shown in Figure 4, where the reinforcement learning agent consists of a PPO, and the environment is the object that the reinforcement learning agent learns. In this experiment, the environment refers to the IDS composed of a DNN feature extractor and k-means clustering module. The agent updates the hidden layer model, the number of features to be extracted from the DNN feature extractor, and the number of k-means clusters to be used as policies in the environment. The environment performs intrusion detection based on a policy and sends the resulting value, the F1-score, to the agent. The agent improves the performance of the IDS by iteratively evaluating and improving the policy based on the F1-score.

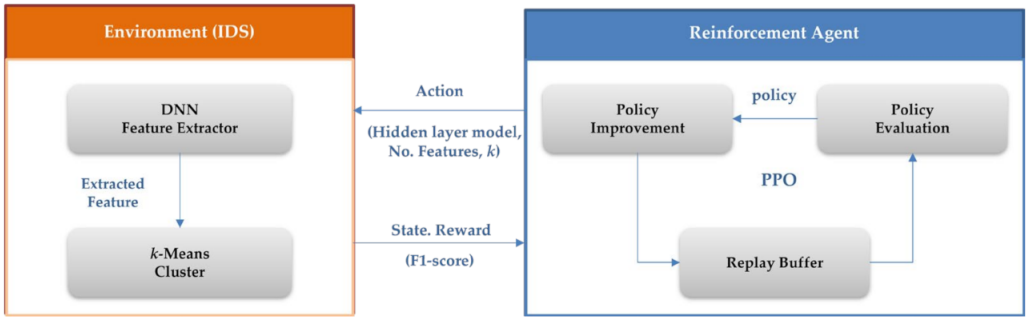


Figure 4. Experimental operation process.

Experiments were performed in three different environments. In the first experiment, only the CICIDS2017 dataset was used. The second experiment used only the MQTTset dataset. Finally, the CICIDS2017 and MQTTset datasets were used as a merged set. The dataset merge was performed by integrating the features extracted from the DNN feature extractor into one file. When the agent sent a new number of features in each experiment, the features were re-extracted, and an integrated file was created. Because the two datasets contain different attack types, they were combined to verify that the proposed IDS with the reinforcement learning control algorithm could cover a wide range of attacks. It is challenging to represent all the rapidly changing network environments by combining two datasets. However, due to the diversity of attacks and normal data patterns in both datasets, the merged dataset cloud better show changes in the network environment than the previous datasets. PPO improves performance by iterating policy refinement, policy evaluation, and replay buffers until the appropriate performance level, which is set by the administrator, is reached.

In this experiment, the PPO reinforcement learning agent sets hyperparameters according to the hyperparameter ranges listed in Table 8.

Table 8. Ranges of hyperparameters.

Model	Range
Output Features	2~40
Hidden Layers	2~10
Activation Function	Sigmoid, TanH, ReLU, ELU, SELU
K-means Clusters	2, 4, 8, 16, 32, 64, 128, 256, 512

4.6. Experimental Evaluation

Table 9 lists the experimental results of measuring the performance of ID-HyConSys using the CICIDS2017 and MQTTset datasets. The result of the experiment using ID-HyConSys was an F1-score of 0.9707 on CICIDS2017 and an F1-score of 0.9973 on MQTTset. An F1-score of 0.9901 was obtained in the experiment in which the datasets were merged. The experiment with merged datasets provided excellent results, despite the wide range of attack types and increased complexity due to the increased volume and increased number of protocols. Each experiment showed performance that was better than or similar to other studies [17,22–24]. A comparison of the experimental results of the CICIDS2017 dataset showed similar performance to the IDS using DBN in [22]. ID-HyConSys showed superior performance to other datasets according to the results of other studies.

Table 9. Comparative performance verification table.

Reference	Algorithm	Dataset	F1 Score
Our Proposed	PPO + DNN + k -means	CICIDS 2017	0.9707
Our Proposed	PPO + DNN + k -means	MQTTset	0.9973
Our Proposed	PPO + DNN + k -means	CICIDS2017 + MQTTset	0.9901
[22]	DBNIDS	CICIDS 2017	0.97
[23]	MTD-IDS	CICIDS 2017	0.800
[17]	Neural Network	MQTTset	0.9023
[17]	Random Forest	MQTTset	0.9140
[17]	Naïve Bayes	MQTTset	0.6872
[17]	Decision Tree	MQTTset	0.9140
[17]	Gradient Boost	MQTTset	0.8727
[17]	Multilayer Perceptron	MQTTset	0.9018
[24]	AdaBoost	MQTTset	0.9572

Additionally, in the old version published in a previous paper [15], before the revision to the existing ID-HyConSys, the F1-score on CICIDS2017 was 0.96552. However, the performance was improved to an F1-score of 0.9707 by adjusting the hidden layer. Because ID-HyConSys is designed to respond flexibly to changes in the network environment, it is expected to be useful in a real environment.

Figures 5–7 show the 2000-repetition experiment training results for each dataset of ID-HyConSys. Figure 5 shows the result of measuring performance using the CICIDS2017 dataset and shows the curve for the F1-score rising from a minimum of 0.8799 to a maximum of 0.9707. Figure 6 shows the performance measurement results using the MQTTset dataset and shows the curve of the F1-score rising from a minimum of 0.9736 to a maximum of 0.9973. Finally, Figure 7 shows the merged dataset results, showing the F1-score rising from a minimum of 0.8747 to a maximum of 0.9901. The red dotted line in each figure is a logarithmic trend line. The trend line shows a gentle upward curve in all graphs. The F1-score shows a range of fluctuations due to the exploratory reinforcement learning process. However, as time passes, the F1-score increases, and the fluctuation range decreases, indicating that it stabilizes. Due to the characteristics of PPO, stability is pursued, and the policy is updated, so it is expected that a very stable F1-score would be obtained if a lot of learning is done.

Tables 10 and 11 show the results of analyzing the models having the best and worst performance on each dataset. Just like the number of extracted features affected intrusion detection, the change in the hidden layers of the model, and the change in the number of clusters affected intrusion detection.

In the case of CICIDS2017, the case where four features were extracted from a hidden-layer model with 10 layers and SELU and ELU activation functions and were then clustered into 512 clusters showed the best performance. In addition, the case where four features were extracted from the hidden layer model using four layers and the ReLU activation function and were then clustered into eight clusters showed the worst performance.

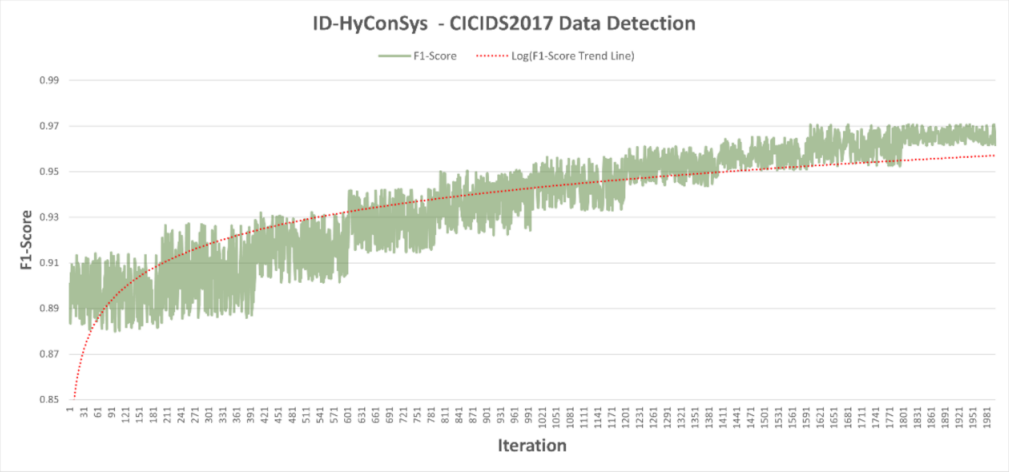


Figure 5. ID-HyConSys—CICIDS2017 data detection.

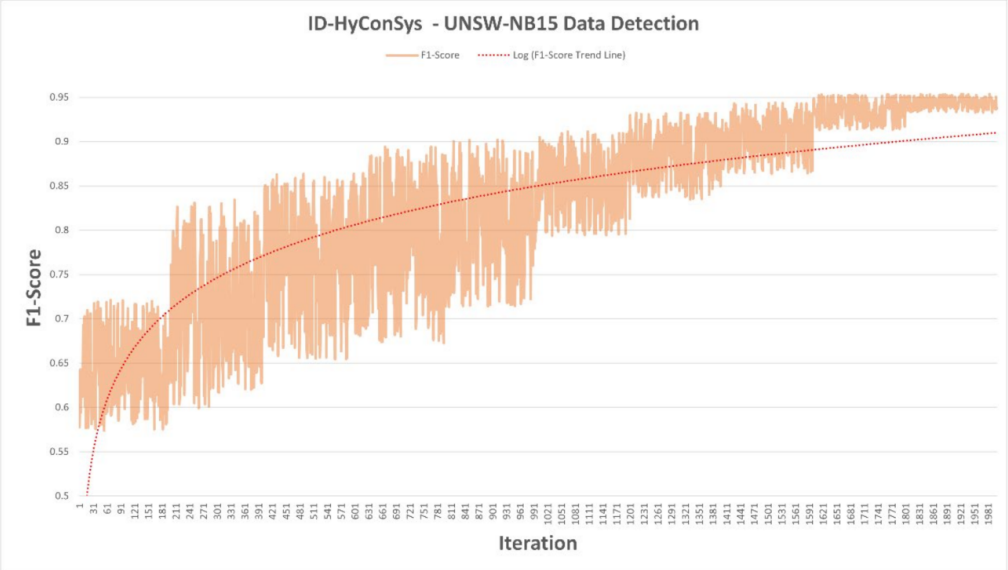


Figure 6. ID-HyConSys—UNSW-NB15 data detection.

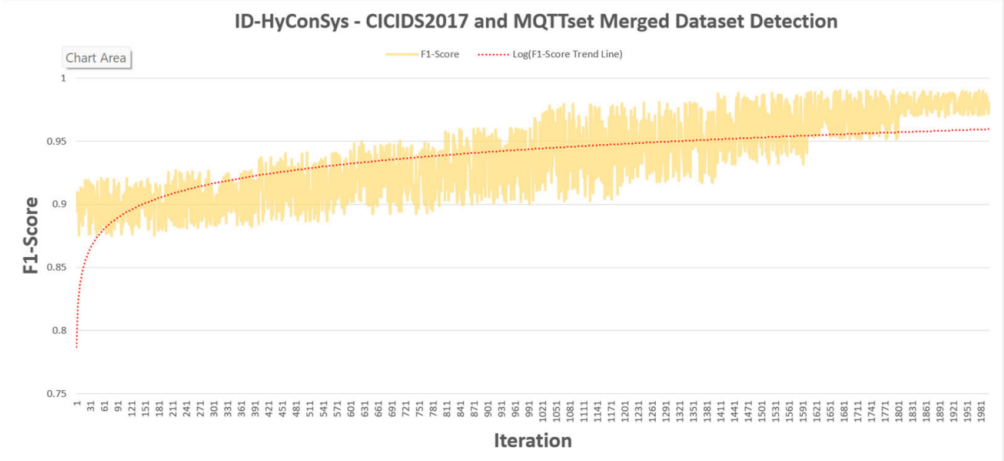


Figure 7. ID-HyConSys—CICIDS2017 and MQTTset merge data detection.

Table 10. Maximum performance model by dataset.

Model	CICIDS2017	MQTTset	Merged Dataset
No. Input Features	78	48	74, 48
No. Extracted Features	4	12	14
No. Hidden Layers	10	7	4
Activation Function	SELU+ELU	ReLU	TanH
k-means Clusters	512	64	128
F1-score	0.9707	0.9973	0.9901

Table 11. Minimum performance model by dataset.

Model	CICIDS2017	MQTTset	Merged Dataset
No. Input Features	78	48	74, 48
No. Extracted Features	22	4	17
No. Hidden Layers	4	4	6
Activation Function	ReLU	TanH	ReLU
k-means Clusters	8	2	8
F1-score	0.8799	0.9736	0.8747

In the case of the MQTTset, the case where 12 features were extracted from the seven layers and the hidden layer model using the ReLU activation function and were then clustered into 64 clusters showed the best performance. In addition, the case where four features were extracted from the hidden layer model using four layers and the TanH activation function and were then clustered into two clusters showed the worst performance.

In the case of the merged dataset, the case where 14 features were extracted from the four layers and the hidden layer model using the TanH activation function and were then clustered into 128 clusters showed the best performance. In addition, the case where 17 features were extracted from the hidden layer model using six layers and the ReLU activation function and were then clustered into eight clusters showed the worst performance.

Looking at the three models, in the case of CICIDS2017, there are many types of attack data, and the complexity of the data is high, so clustering with many clusters is required to achieve good performance. If the number of clusters is large, the intrusion detection time may increase, but reducing the number of clusters does not significantly degrade performance. Therefore, it seems that the number of clusters can be adjusted slightly when data are excessively crowded and fast processing is required.

In the case of MQTTset, the optimal model can easily be found because the performance change is small according to the model change. In addition, because the number of clusters in the model having the best performance is not large, intrusion detection is expected to proceed quickly.

The merged dataset has an extensive range of fluctuations between the best and worst performance compared with CICIDS2017 and UNSW-NB15. Additionally, the worst performance on the merged dataset is worse than that on CICIDS2017 or MQTTset due to increased complexity. Because performance varies widely across model sets, it seems likely that hyperparameters will need to be fine-tuned to find the optimal model. However, the merged dataset confirmed that high performance could be obtained due to the wide output range of TanH despite the increased complexity.

To improve the performance of the IDS, we adjusted the hidden layer model, the number of output features of the DNN feature extractor, and the number of clusters obtained by the k-means clustering algorithm. In the case of the number of k-means clusters, performance generally improves when a large number of clusters are used, but the intrusion detection speed may be affected depending on the number of clusters. Therefore, it seems that the appropriate number can be selected differently according to the computing power of the IDS. However, the generation of hidden layers and the number of output features of the DNN feature extractor are important factors, and an appropriate adjustment

of the hidden layers and the number of output features through PPO makes it possible to build an IDS that can respond quickly to a changing network environment.

The DNN assigns weights to the input data according to the configuration of the hidden layer, and it outputs the output value by changing the output value to a non-linear structure using the activation function. The general DNN structure improves the DNN model by using the error-backpropagation function through repeated learning based on a pre-set answer, but the DNN feature extractor used in this study does not have an answer for the extracted feature. Therefore, it does not improve the DNN model through functions, such as error-backpropagation, but it improves the feature extractor by judging and adjusting the performance of the model itself through PPO. Although the DNN feature extractor plays a role, such as dimensionality reduction, that adjusts the number of features to an appropriate level, it is similar to a structure that generates data, such as the auto-encoder and GAN. In addition, the DNN feature extractor has a structure that creates new features from the input network data and detects intrusions using *k*-means with the newly created features.

If we analyze the structure of the model that achieves the best performance on each dataset, the number of clusters is generally set very high. If the number of clusters increases, the learning time is significantly affected, but it is judged that there is no problem in real-time detection. When the number of clusters increases during data learning, separating the training data and dividing the area takes a lot of time. However, the number of clusters does not affect real-time detection because the detection mode processes the extracted data as vector values and only checks whether an attack or normal in an already divided area. However, as the number of clusters increases, a lot of memory is needed. Therefore, in terms of memory, there is an inefficient part, so it is necessary to adjust the number of clusters to an appropriate level according to the computing power of the IDS.

5. Conclusions

In this study, we proposed a method to improve the performance of ID-HyConSys, which automatically updates the intrusion detection process and improves performance by controlling the hyperparameters of the DNN-based feature extractor and k-means cluster module using the existing PPO algorithm. By using PPO to flexibly cope with the network environment using the previously limited hidden layer configuration, ID-HyConSys is configured for use in the existing experimental environment as well as in the IoT environment.

To evaluate the system's performance, the CICIDS2017 and MQTTset datasets were used individually, and experiments were also conducted on a merged dataset.

As an experimental result, the F1-score was 0.9707 on CICIDS2017, 0.9973 on MQTTset, and 0.9827 on the merged dataset, and very high F1-scores were obtained in all experiments.

It is very important to extract meaningful data from an IDS. In many studies, feature selection or feature extraction and various deep learning algorithms have been used to find meaningful features among the many features of network data. In the experimental results on the limited intrusion detection dataset, meaningful results were obtained relatively easily, but these studies do not show that these experimental results are immediately generalizable [11–14]. In a real environment, it is very difficult to quickly analyze all data and extract important characteristics of an attack because the characteristics of the data change rapidly.

Cyber-attacks are rapidly developing and diversifying day by day. In contrast, intrusion detection datasets take a long time to study and are very limited. In order to solve these problems, this study proposes ID-HyConSys, which can respond flexibly to a changing network environment using the PPO algorithm. The IDS consists of a DNN feature extractor that extracts meaningful data from data having numerous features and a k-means cluster module that clusters similar data. These operations are automated by directly manipulating hyperparameters using the PPO.

Because changes in the number of feature extractions, hidden layer model composition, and clustering number greatly affect intrusion detection performance, PPO can find the optimal value through a lot of learning and can safely protect the IoT environment from various attacks.

A testbed allows one to apply and explore various ML algorithms to show high levels of performance. However, in the real world, a more complex situation arises. The proposed ID-HyConSys shows the ability to solve these complex problems, such as general-purpose IDS, through iterative reinforcement learning.

In this study, an experiment was conducted using an open dataset for experimental reliability, and a more complex environment was constructed using a merged dataset to overcome the limitations of the dataset. However, various datasets need to be studied to analyze and prevent stealthy attacks, such as false data injection attacks [41,42].

In a future study, we will be able to examine how to design a more optimized model by automatically operating the system through reinforcement learning and improving the performance of the IDS, as well as learning the characteristics of the data itself.

Author Contributions: Conceptualization, H.H. and Y.K.; methodology, H.H.; software, H.H.; validation, H.H., H.K. and Y.K.; formal analysis, 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.; 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 2022.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The datasets used in the study are publicly available to everyone and can be accessed at: https://www.unb.ca/cic/datasets/ids-2017.html (accessed on 4 September 2022), https://www.kaggle.com/datasets/cnrieiit/mqttset (accessed on 4 September 2022), for CICIDS2017, and MQTTset dataset, respectively.

Acknowledgments: This work was supported by the Dongguk University Research Fund of 2022.

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**, *9*, 26–41. [CrossRef]
2. Debar, H.; Dacier, M.; Wespi, A. Towards a taxonomy of intrusion-detection systems. *Comput. Netw.* **1999**, *31*, 805–822. [CrossRef]
3. Liao, Y.; Vemuri, V.R. Use of K-Nearest Neighbor classifier for intrusion detection. *Comput. Secur.* **2002**, *21*, 439–448. [CrossRef]
4. Chen, W.; Hsu, S.; Shen, H. Application of SVM and ANN for intrusion detection. *Comput. Oper. Res.* **2005**, *32*, 2617–2634. [CrossRef]
5. Min, L.; Wang, D. Anormaly Intrusion Detection Based on SOM. In Proceedings of the 2009 WASE International Conference on Information Engineering, Taiyuan, China, 10–11 July 2009; pp. 40–43.
6. Sahu, S.; Mehtre, B.M. Network intrusion detection system using J48 Decision Tree. In Proceedings of the 2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Kochi, India, 10–13 August 2015.
7. Zhang, J.; Zulkernine, M.; Haque, A. Random-Forests-Based Network Intrusion Detection Systems. *IEEE Trans. Syst. Man Cybern. Part C (Appl. Rev.)* **2008**, *38*, 649–659. [CrossRef]
8. 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]
9. Gaikwad, D.P.; Thool, R.C. Intrusion Detection System Using Bagging with Partial Decision TreeBase Classifier. *Procedia Comput. Sci.* **2015**, *49*, 92–98. [CrossRef]
10. Masarat, S.; Taheri, H.; Sharifian, S. A novel framework, based on fuzzy ensemble of classifiers for intrusion detection systems. In Proceedings of the 2014 4th International Conference on Computer and Knowledge Engineering (ICCKE), Mashhad, Iran, 29–30 October 2014.
11. Prasad, M.; Tripathi, S.; Dahal, K. An efficient feature selection based Bayesian and Rough set approach for intrusion detection. *Appl. Soft Comput.* **2020**, *87*, 105980. [CrossRef]

12. Park, K.; Song, Y.; Cheong, Y. Classification of Attack Types for Intrusion Detection Systems Using a Machine Learning Algorithm. In Proceedings of the 2018 IEEE Fourth International Conference on Big Data Computing Service and Applications (BigDataService), Bamberg, Germany, 26–29 March 2018.
13. Tasi, C.; Lin, C. A triangle area based nearest neighbors approach to intrusion detection. *Pattern Recognit.* **2010**, *43*, 222–229.
14. Rababah, B.; Srivastava, S. Hybrid Model for Intrusion Detection Systems. *arXiv* **2020**, arXiv:2003.08585.
15. 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. [CrossRef]
16. 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.
17. Vaccari, I.; Chiola, G.; Aiello, M.; Mongelli, M.; Cambiaso, E. MQTTset, a New Dataset for Machine Learning Techniques on MQTT. *Sensors* **2020**, *20*, 6578. [CrossRef] [PubMed]
18. Murder, A.; Alia, A.; Amer, A.; Saleh, A.; Abdul, R. Deep recurrent neural network for IoT intrusion detection system. *Simul. Model. Pract. Theory* **2020**, *101*, 102031.
19. Yang, A.; Zhuansun, Y.; Liu, C.; Li, J.; Zhang, C. Design of Intrusion Detection System for Internet of Things Based on Improved BP Neural Network. *IEEE Access* **2019**, *7*, 106043–106052. [CrossRef]
20. Liang, C.; Shanmugam, B.; Azam, S.; Jonkman, M.; Boer, F.D.; Narayansamy, G. Intrusion Detection System for Internet of Things based on a Machine Learning approach. In Proceedings of the 2019 International Conference on Vision towards Emerging Trends in Communication and Networking (ViTECoN), Vellore, India, 30–31 March 2019.
21. Jan, S.U.; Ahmed, S.; Shakhov, V.; Koo, I. Toward a Lightweight Intrusion Detection System for the Internet of Things. *IEEE Access* **2019**, *7*, 42450–42471. [CrossRef]
22. Manimurugan, S.; AI-Mutairi, S.; Aborokbah, M.M.; Chilamkurti, N.; Ganesan, S.; Patan, R. Effective Attack Detection in Internet of Medical Things Smart Environment Using a Deep Belief Neural Network. *IEEE Access* **2020**, *8*, 77396–77404. [CrossRef]
23. Yang, L.; Moubayed, A.; Shami, A. MTH-IDS: A Multitiered Hybrid Intrusion Detection System for Internet of Vehicles. *IEEE Internet Things J.* **2022**, *9*, 616–632. [CrossRef]
24. Rachmadi, S.; Mandala, S.; Oktaria, D. Detection of DoS Attack using AdaBoost Algorithm on IoT System. In Proceedings of the 2021 International Conference on Data Science and Its Applications (ICoDSA), Bandung, Indonesia, 6–7 October 2021.
25. Freund, Y.; Schapire, R.E. A Short Introduction to Boosting. *J. Jpn. Soc. Artif. Intell.* **1999**, *14*, 1–14.
26. Agarap, A.F. Deep Learning using Rectified Linear Units (ReLU). *arXiv* **2019**, arXiv:1803.08375v2.
27. Clevert, D.; Unterthiner, T.; Hochreiter, S. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). *arXiv* **2016**, arXiv:1511.07289v5.
28. Klambauer, G.; Unterthiner, T.; Mayr, A.; Hochreiter, S. Self-Normalizing Neural Networks. *arXiv* **2017**, arXiv:1706.02515v5.
29. Yan, B.; Han, G. Effective Feature Extraction via Stacked Sparse Autoencoder to Improve Intrusion Detection System. *IEEE Access* **2018**, *6*, 41238–41248. [CrossRef]
30. 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.
31. 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]
32. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal Policy Optimization Algorithms. *arXiv* **2017**, arXiv:1707.06347v2.
33. Khraisat, A.; Gondal, I.; Vamplew, P.; Kamruzzaman, J. Survey of intrusion detection systems: Techniques, datasets and challenges. *Cybersecurity* **2019**, *2*, 1–20. [CrossRef]
34. Maseer, Z.K.; Yusof, R.; Bahaman, N.; Mostafa, S.A.; Foozy, C.F.M. Benchmarking of Machine Learning for Anomaly Based Intrusion Detection Systems in the CICIDS2017 Dataset. *IEEE Access* **2021**, *9*, 22351–22370. [CrossRef]
35. 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. *Computer* **2019**, *52*, 41–51. [CrossRef]
36. MQTT Protocol Market Share. Available online: https://talks.navixy.com/reviews/mqtt-protocol/ (accessed on 4 September 2022).
37. CICIDS2017, Intrusion Detection Evaluation Dataset. Available online: https://www.unb.ca/cic/datasets/ids-2017.html (accessed on 4 September 2022).
38. Ghazanfar, S.; Hussain, F.; Rehman, A.U.; Fayyaz, U.U.; Shahzad, F.; Shah, G.A. IoT-Flock: An Open-source Framework for IoT Traffic Generation. In Proceedings of the 2020 International Conference on Emerging Trends in Smart Technologies (ICETST), Karachi, Pakistan, 26–27 March 2020.
39. Vaccari, I.; Aiello, M.; Cambiaso, E. SlowITe, a Novel Denial of Service Attack Affecting MQTT. *Sensors* **2020**, *20*, 2932. [CrossRef]
40. Keras the Python Deep Learning API Home Page. Available online: https://keras.io (accessed on 4 September 2022).
41. Pang, Z.; Fan, L.; Sun, J.; Liu, K.; Liu, G. Detection of stealthy false data injection attacks against networked control system via active data modification. *Inf. Sci.* **2021**, *546*, 192–205. [CrossRef]
42. Pang, Z.; Fan, L.; Dong, Z.; Han, Q.; Liu, G. False Data Injection Attacks Against Partial Sensor Measurements of Networked Control Systems. *IEEE Trans. Circuits Syst. II Express Briefs* **2022**, *69*, 149–153. [CrossRef]