algorithms

MDPI

Article

Packet-Level and Flow-Level Network Intrusion Detection Based on Reinforcement Learning and Adversarial Training

Bin Yang †, Muhammad Haseeb Arshad † and Qing Zhao *

Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 2R3, Canada
* Correspondence: qingz@ualberta.ca; Tel.: +1-780-492-5792
† These authors contributed equally to this work.

Abstract: Powered by advances in information and internet technologies, network-based applications have developed rapidly, and cybersecurity has grown more critical. Inspired by Reinforcement Learning (RL) success in many domains, this paper proposes an Intrusion Detection System (IDS) to improve cybersecurity. The IDS based on two RL algorithms, i.e., Deep Q-Learning and Policy Gradient, is carefully formulated, strategically designed, and thoroughly evaluated at the packet-level and flow-level using the CICDDoS2019 dataset. Compared to other research work in a similar line of research, this paper is focused on providing a systematic and complete design paradigm of IDS based on RL algorithms, at both the packet and flow levels. For the packet-level RL-based IDS, first, the session data are transformed into images via an image embedding method proposed in this work. A comparison between 1D-Convolutional Neural Networks (1D-CNN) and CNN for extracting features from these images (for further RL agent training) is drawn from the quantitative results. In addition, an anomaly detection module is designed to detect unknown network traffic. For flow-level IDS, a Conditional Generative Adversarial Network (CGAN) and the ε-greedy strategy are adopted in designing the exploration module for RL agent training. To improve the robustness of the intrusion detection, a sample agent with a complement reward policy of the RL agent is introduced for the purpose of adversarial training. The experimental results of the proposed RL-based IDS show improved results over the state-of-the-art algorithms presented in the literature for packet-level and flow-level IDS.

Keywords: reinforcement learning; intrusion detection; convolutional neural network; generative adversarial network

Citation: Yang, B.; Arshad, M.H.; Zhao, Q. Packet-Level and Flow-Level Network Intrusion Detection Based on Reinforcement Learning and Adversarial Training. *Algorithms* **2022**, *15*, 453. https://doi.org/10.3390/a15120453

Academic Editor: Mustafa Demetgül

Received: 5 October 2022
Accepted: 25 November 2022
Published: 30 November 2022

1. Introduction

Modernizing the economy requires a strong network: business, scientific research, entertainment, and education demand a dependable and secure network infrastructure. If the network infrastructure is broken, personal information will be disclosed, and society will be in disarray. In April 2020, the World Health Organization (WHO) unveiled an uptick in cyber intrusions, exposing 450 email addresses and passwords. Cybersecurity research is therefore vital for society's progress and stability. Traditionally, firewalls are used to safeguard network infrastructure and data confidentiality. An Intrusion Detection System (IDS), however, analyses network traffic in real-time and alerts or reacts to questionable traffic and thus can provide necessary information for preventing harmful assaults that could destroy IT infrastructure. Packet-level and flow-level analyses are two fundamental methods for designing IDS; however, flow-level-based IDS research is relatively more predominant than packet-level-based IDS research.

Conventional IDSs (Con-IDS) rely on pattern matching, which requires user-defined filtering criteria [1]. Setting up perfect filtering rules is challenging, even for specialists, thus limiting the use of Con-IDS to small networks. Hackers may easily bypass manually defined rules. Furthermore, there is no mechanism for proper rule updating. Seeing the success of

Artificial Intelligence (AI) in many research fields, network engineers and researchers are investigating its applications in cybersecurity. Using AI in IDS can effectively minimize issues related to Con-IDS [2–7]. Deep Neural Networks (DNNs) can learn typical properties of network assaults and apply categorization or detection, eliminating the need for explicit user-defined rules [8]. Because neural networks are a type of black box to everyone, hackers cannot acquire what kind of patterns and relationships are captured by the neural networks. Therefore, it is more challenging for hackers to create unique assaults that can evade detection. In particular, there has been a push towards agent-based IDS solutions that can learn from their surroundings and adapt to new threats in ways that humans cannot (e.g., by acting appropriately in response to a given or a novel threat) [9].

Reinforcement Learning (RL) is the prominent method of using automated agents for detection and classification problems. In many ways, RL and IDS are extremely compatible. RL algorithms are used for solving Markov Decision Process (MDP) problems [10]. In essence, network flow is a form of a dynamic process that a Markov process may represent. Furthermore, intrusion detection might be regarded as a distinct game. The RL agent in IDS may observe the patterns of attacks on various ecosystems and use that information to plan for future defenses that will be more effective. Depending on the nature of the input it receives from the environment, an RL technique may reward or punish a given activity, increasing its capacity to protect the environment (e.g., in a trial-and-error interaction, identify what works better with a specific environment). Throughout its lifetime, an RL agent can improve its powers. Several RL-based IDS have been presented to offer autonomous cyber protection solutions for various environments. The use cases range from the Internet of Things (IoT) and wireless networks to the cloud [11–15]. During the learning process, the RL agent can use its observations to execute self-learning capabilities without the need for human supervision or expert knowledge [16]. However, the majority of the currently available methods are either unable to cope with a huge dataset or cannot fully recognize authentic network activity reliably. This is because an RL agent often faces the state explosion problem when confronted with large learning states. In recent years, proposals have been made to overcome the major limitation of existing RL approaches by introducing Deep Reinforcement Learning (DRL) techniques capable of learning in an environment with a vast number of states. Using deep neural networks throughout the learning process, DRL approaches such as deep Q-learning have shown promise in solving the state explosion problem [17].

Contributions of the Proposed Work

Different intrusion datasets are used to train and assess the several DRL-based IDS approaches that have been reported in the recent research literature [18,19]. Nonetheless, the majority of the work aims to improve the detection performance compared to other methods of their kind. Many of them only scratch the surface of developing and implementing a DRL-based IDS approach without providing essential details, such as how the DQL agent can be formulated in light of an RL principle or how to fine-tune hyper-parameters for improved self-learning and interaction with the underlying network. Furthermore, in the existing literature, flow and packet-level intrusion detection is usually treated separately by using different methods. In this paper, we intend to exploit the application of RL to provide a unified framework for intrusion detection problems at both the packet level and flow level. We achieve this by strategically formulating intrusion detection problems into an RL problem and then designing the overall IDS structure in a DQN framework. The proposed method can not only detect malicious attacks but also classify their types. To summarize, the contributions of this paper are listed as follows:

1. By means of a proposed image embedding scheme, network traffic data are transformed into time-series first and then image data so that it can be processed by CNN and 1D-CNN-based architecture of the RL agent to tackle the intrusion detection problem.

2. Flow information is attached with the dataset used for the packet-level IDS to incorporate the temporal information for further performance improvement.
3. The exploration module is designed based on a CGAN and the ε-greedy policy for the flow level IDS.
4. An anomaly detection model is designed for the packet-level and flow-level IDS to detect unknown novel attacks.
5. In order to improve the robustness of the final detection, a sample agent with a complement reward policy of the RL agent is introduced in the RL framework for adversarial training.
6. Finally, since RL is known to be sensitive to the hyper-parameters chosen, in this paper, a Bayesian search is conducted to find the optimal parameters.

In the remainder of the paper, Section 2 includes a brief review of related work. Sections 3 and 4 introduce in detail the proposed intrusion detection frameworks at the packet level and flow level, respectively. Section 5 includes results for both packet-level and flow-level experiments. Finally, Section 6 draws conclusions and discusses future work.

2. Related Work

Q-learning, which does not require a model, has been lauded as a promising research strategy, mainly applied to complex decision-making problems. This strategy is useful when others, such as classical optimization and supervised learning methods, are inapplicable [20]. The strength of Q-learning lies in its efficient output, its adaptability via learning, and its potential for integration with other models.

Research into the use of machine learning in cybersecurity, including DRL [21–23], both supervised and unsupervised learning, has been conducted in previous work [19,20,24,25]. The authors of [24] presented a systematic overview of DRL regarding cybersecurity. They discussed the DRL-based security solutions for autonomous and non-autonomous physical systems as well as the game theory-based multi-agent DRL cyber-attack defense system. The studies were conducted in light of real-world and virtual settings. In [25], the authors demonstrated how adversarial learning can help improve the robustness of the DRL-based IDS model used for cyber defense. In their approach, adversarial training was performed by randomly sampling a session from the dataset and generating a positive reward based on the prediction at first, and then the environment was forced to predict wrong. Doing this, the environment acts as an adversarial agent and causes the classifier to further learn better underlying features that ultimately improve the robustness of the DRL-based IDS system. In [19], the authors applied the NSL-KDD [26] and AWID [27] datasets to the study of many DRL algorithms for intrusion detection, including Double Deep Q-Network (DDQN), Deep Q-Network (DQN), Policy Gradient (PG), and Actor-Critical (AC) methods. The supervised machine learning algorithms were trained on those datasets to categorize intrusion occurrences. They used the idea of a virtual environment and sampling methods to replace the interaction module of RL training. It was demonstrated how DRL may outperform alternative machine learning methods when applied to today's data networks requiring immediate attention and reaction. The results show that DDQN is superior to the competing methods in terms of classification performance and learning speed.

Based on the stateful MDP, deep Q-learning was suggested in [28,29]. They contrasted their method's efficacy with conventional planning-based and DRL-based IDS, using metrics such as execution time, batch size, and cumulative reward to measure performance.

In [30], the authors advocated installing an RL agent on routers so that they might learn from network traffic and divert it away from the target server. They also proposed the novel idea of using the Coordinated Team Learning (CTL) approach rather than the classical multi-agent RL-training. Their proposed algorithm was able to handle the scalability issue with the network traffic in a much better way. In addition, Hidden Markov Models (HMMs) were presented in [31] as a machine learning technique for detecting multi-step assaults by predicting the attacker's next move. The authors used the preliminary offline training where the IDS prediction was compared with the stored database information to avoid

over-fitting. The authors of [32] presented a decision-theoretic framework based on the cost-sensitive and self-optimizing performance, called Anomaly Detection And Response System (ADRS), and they also examined its operation in autonomous networks.

The authors of [33] created an effective IDS by fusing the multi-objective decision problem with evolutionary algorithms. They treated an IDS as a multi-attribute decision-making issue that must consider factors such as implementation cost, resource limitation, time efficiency, and modification costs. This multi-objective problem sought a solution to bring down these functions' values before reacting to threats. A weighted linear combination method was used for preprocessing the network traffic. The idea was to map the different network traffic to a similar metric that could improve the classification accuracy.

Recently, in [34], an efficient way of fine-tuning the hyper-parameters for DQ-learning was suggested by the authors. They evaluated on the NSL-KDD dataset that a small exploration rate achieves better classification accuracy.

An online version of DQN-based IDS was proposed in [35]. The idea was to use the auto-encoders in the Q-network of RL as function approximators. It helped to improve the classification accuracy while real-world data were processed through the RL-trained network. The use of auto-encoders greatly reduces the effect of human interaction and thus achieves significantly higher efficiency during the online training mode.

Most previous IDS research work is focused on how to improve the performance, or how the suggested strategies fare in contrast to others of a similar kind that rely on Machine Learning (ML). Based on the literature review, in this paper, our goal is to provide a general RL framework for both packet- and flow-level IDS problems with improved robustness. The proposed method can not only detect the attack traffic but can also match the attack traffic into separate classes, including normal traffic and different types of attacks, which, to the best of the authors' knowledge, has not been widely studied in the literature. The structure of the IDS at the packet and flow level is clearly laid out, and the design of each module is elaborated on in detail. The robustness of the proposed algorithm is improved by using the notion of a negative (sample) agent during the RL training. The exploration rate is adjusted based on the extensive search and bound method. Quantitative results of the proposed RL algorithms are shown to verify that the presented framework can effectively classify the regular traffic with higher accuracy and explicitly identify each attack separately using both packet and flow level information.

3. The Proposed IDS Framework at the Packet Level

In the packet-level intrusion detection, messages and headers transported by network packets are primarily extracted for detecting malignant traffic [2,3]. A packet structure consists of messages generated in the application layer and three headers generated in the network layers. The content of messages generated in the application layer is different for different protocols. For example, HTTP generates particular HTTP request messages. Other headers carry further information. TCP and UDP, with TCP and UDP headers, are the two most common protocols in the transportation layer. IP, with the IP header, is the most common protocol in the network layer. Similarly, an Ethernet header is generated in the Physical/Ethernet layer. The knowledge carried inside these headers is called a field.

The proposed IDS framework at packet-level is shown in Table 1. The framework consists of two major modules and a number of sub-modules. The Preprocessing module is devised for data transformation and feature engineering. The Reinforcement Learning module is devised for training the IDS with RL approaches. This module further comprises a training module and an interaction module. An additional anomaly detection module is designed in this paper to detect those novel attacks blind to the training module.

Table 1. Intrusion Detection Framework at Packet-level.

Module	Sub-Module	Function
Preprocessing	None	Data Transformation and Feature Engineering
Reinforcement Learning	Interactive Training	Store batch data into replay buffer [1] Train the agent
Anomaly Detection	None	Detect attacks blind to the training module

[1] The replay buffer is used to facilitate training. It stores batch data when an agent is interacting with the environment. In the training stage, training data is then sampled from the replay buffer.

3.1. Preprocessing Module

Inspired by the adoption of natural language processing (NLP) techniques, such as word2vec, in processing massive text documents and metadata files [36,37], an embedding method, namely the image embedding of network packets for the RL-based IDS, is proposed in this section. Table 2 shows the main steps of the preprocessing module.

Table 2. The Preprocessing Module at Packet-level.

Step	Input	Method	Output
1	Raw Network Traffic	Session-based Rule	Seperated Sessions
2	Seperated Sessions	Image Embedding	Session Images
3	Session Images	Labeling with Log Files	Images and Label
4	Images and Label	Normalization	Applicable Dataset

In step 1, the network traffic recorded in the 'pcap' files is separated into small traffic files according to the session-based partition rules. One common rule is that if two packets share the same 5-tuple knowledge (source IP, source port, destination IP, destination port, transportation protocol), then they will be categorized in the same session. After partition, several separated sessions stored in 'pcap' files containing various packets recorded in the order of capture time are obtained.

In step 2, the separated sessions are converted into images using the proposed image embedding method. Generally, a network packet in a session consists of an Ethernet header, a TCP or UDP header, an IP header and application messages. The total field length of the first three headers (except the application message) is 54 bytes. The application message field is dropped during image embedding because of its varying length. The 54-byte packet with selected fields is converted to a line of the image, with each byte representing one pixel. Using one session as an example, the whole procedure of image embedding is illustrated in Figure 1. Thus, a session is transformed into a standard image format with a fixed size of 54×54, an image consisting of only 54 packets, as shown in Figure 2. For sessions with more than 54 packets, additional images were created by repeating the embedding for the extra packets. Zero-padding is used if the number of remaining packets is less than 54 to ensure the fixed image size of 54×54. It is essential to embed packets in the order in which they appear in the session. This way, session information is implicitly attached to the packet-level experiments.

In step 3, after image embedding, each session is labeled according to the time stamp given in the log file of the raw dataset. Finally, in the last step, all generated images' pixels are normalized into [0, 1] from [0, 255].

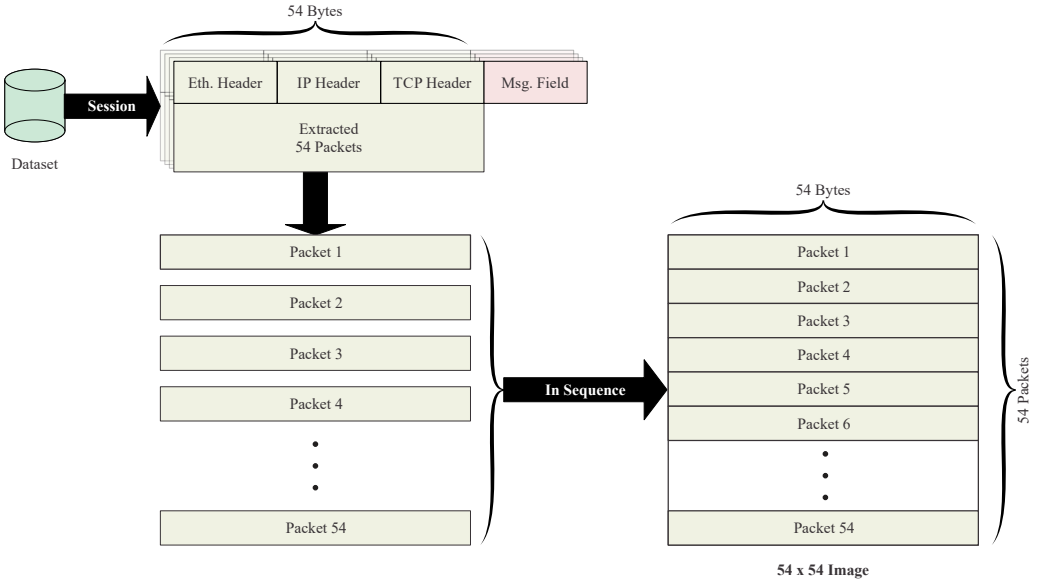

Figure 1. Procedure of the proposed image embedding, i.e., transforming packets to an image.

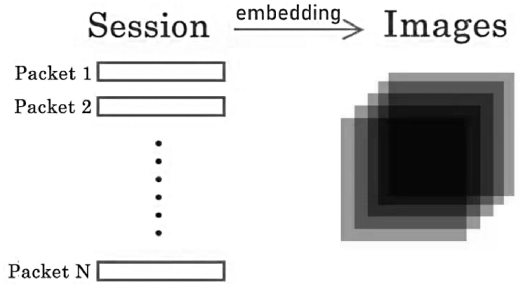

Figure 2. Transform a session into a batch of images. The number of images M is calculated by: $[(N-1)/54] + 1$.

3.2. Reinforcement Learning Module

Before developing a reinforcement learning module, it is necessary to transfer the intrusion detection problem into an RL-based problem. For this purpose, the intrusion detection problem has been compared to the Atari games studied in [38], considering intrusion detection as a special game. The comparison is shown in Table 3.

Table 3. Comparison between Atari game and the intrusion detection game.

RL	Symbol	Atari	IDS	IDS Space
State	s	Image(s)	Image	Images in dataset
Action	a	Game Operation	Prediction	Label Space {0, 1, 2...}
Reward	r	Game Feedback	Reward Mechanism	{+1, −1}
Episode	–	Game Round	Session	–
Agent	–	Game User	Classifier	–

The agent can take the following trajectory until it reaches the end (last packet) of a session:

$$s_0, a_1, r_1, s_1, a_2, r_2, \ldots, s_i, a_i, r_i, \ldots, s_t$$

In the Atari game, the agent repeats until it reaches the terminated state (game over) of an episode. Similarly, the intrusion detection agent can also take the above trajectory until it reaches the session's end (last image). The state is the game screen in the Atari game, and in IDS, the state is also an image after image embedding on the sessions. The action space of the Atari game contains game operations, which can be expressed as discrete numbers,

e.g., 0 (one step left), 1 (one step right), 2 (one step up), etc. The action space of the IDS, however, contains the types of the traffic class, which can also be expressed as discrete numbers such as 0 (normal), 1 (attack 1), 2 (attack 2), etc. The reward system is different for the above two problems. For the IDS, the reward system has to be designed carefully. After conducting the research, it has been found that the reward scale significantly impacts the performance of reinforcement learning algorithms. Thus, inspired by the reward clip for the code-level optimization implemented in the Atari game agent [39], the following reward feedback rule has been designed: if the prediction made by the agent is correct, the reward is 1; otherwise, the reward is −1. Table 4 shows the significance of the trajectory and the transformation between RL and IDS.

Table 4. Conversion of IDS to RL at Packet-level.

RL	Symbol	IDS	IDS Space
State	s	Image	Session Images
Action	a	Prediction	Label Space {0, 1, 2, . . .}
Reward	r	Reward Mechanism	{+1, −1}
Episode	–	Session	–
Agent	–	Classifier	–
Step	i	Packet Position in a session	–
Terminated	t	Last Packet in a session	–

DQN learning is an excellent choice for solving discrete problems and is much easier to implement compared to Proximal Policy Optimization (PPO). After identifying these essential reinforcement learning ingredients, we start designing our reinforcement learning module. Convolutional Neural Networks (CNNs) are chosen as the network structure of the IDS agent since the input states are images. It should be noted that 1D-CNN is also appropriate for the IDS. Each session's images are treated as a type of time-series data since packets embedded in the image are arranged in chronological order according to the capture time. Therefore, 1D-CNN is chosen as an appropriate choice for comparison during the evaluation of the proposed IDS. The main structure of the two RL agents, i.e., DQN-CNN and DQN-1D-CNN, are shown in Figure 3.

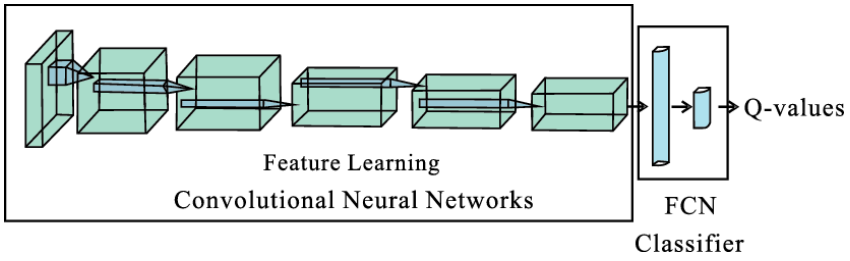

Figure 3. Main structure of two reinforcement learning agents DQN-CNN and DQN-1D-CNN. For DQN- CNN, the structure of the feature learning part is CNN. For DQN-1D-CNN, the structure of the feature learning part is 1D-CNN. FCN represents the fully connected network.

The input states of the RL agents are a finite batch of images. CNN and 1D-CNN are applied for feature extraction on DQN-CNN and DQN-1D-CNN, respectively. An additional fully connected layer is deployed for final classification and detection. The output is Q-values of the current state, and the prediction/action is decided based on the following set of equations.

$$Q^{new}(s_i, a_i) \leftarrow Q^{curr}(s_i, a_i) + \alpha\left(r_i + \gamma \underset{a}{max} Q(s_{i+1}, a_{i+1}) - Q(s_i, a_i)\right) \quad (1)$$

where α is the learning rate and γ is the reward discount factor. The final action is calculated as

$$a^* = \underset{a}{argmax} Q(s, a) \tag{2}$$

As shown in Table 1, the RL module in the IDS has two sub-modules, the interaction module and the training module. The complete procedure of the interaction module is shown in Algorithm 1. The complete procedure of the training module is shown in Figure 4 (taking DQN as an example). Two networks (the Update and the Target) operate together to achieve the approximate regression. The functionality of the target network is to improve the training stability [40] by fixing the regression target in N steps. The structure of the update network is the same as that of the agent, while the target network copies the structure from the update network and is initialized with the same parameters of the update network. The update network is updated through back propagation (BP), and the target network is updated by copying the parameters from the update network every N times.

Algorithm 1 Interaction Module

procedure START INTERACTING
top:
    Randomly sample a session and obtain its label
    Take the first image in the session as current state s_1
    Feed s_1 into agent and obtain action/prediction a_1
    Feed a_1 and label into reward mechanism, obtain reward r_1
    $i \leftarrow 0$
loop:
    $i \leftarrow i + 1$
    if Last image in the session **then**
        Store $(s_i, a_i, r_i, None)$ into replay buffer
        break
    else
        Take next image in the session as next state s_{i+1}
        Store (s_i, a_i, r_i, s_{i+1}) into replay buffer
    end if
    goto *loop*
    goto *top*
end procedure

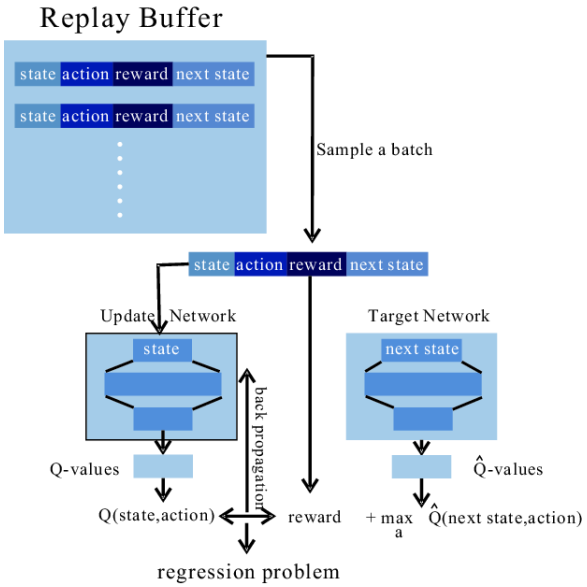

Figure 4. Deep Q-Learning Training Module.

The training module and the interaction module work alternately. For example, once the replay buffer is full, training for a specified number of epochs can start. After training, the new agent is set to interact with the environment and store new data into the replay buffer while removing the old data. The complete pack-level IDS algorithm, including both the interaction module and training module, is shown in Algorithm 2.

Algorithm 2 Deep Q-Learning Framework for Detection at Packet-level

procedure DEEP Q-LEARNING
  Extract sessions from raw traffic file
  Split sessions based on session-based rule
  Conduct image embedding on each session
  Initialize Q function for agent, set target Q function $\bar{Q} = Q$
  for 1 . . . m iterations **do**
    Start interacting ***until*** replay buffer is full
    for 1. . . n epochs **do**
      Sample a batch data (s_i, a_i, r_i, s_{i+1}) from replay buffer
      Target $y = r_i + \gamma \underset{a}{max} \bar{Q}(s_{i+1}, a)$
      Update Q function through back propagation to make $Q(s_i, a_i)$ close to y
      Every C steps reset $\bar{Q} = Q$
      Interact to replace old data
    end for
  end for
end procedure

3.3. Anomaly Detection Module

The purpose of the anomaly detection module is to detect novel attacks that are blind to the training set by considering them as anomaly classes. This is important for a robust IDS because, in reality, it is impossible to include all types of attacks in the training set.

The Q-values output generated by the agent is tested against the set manual threshold 'λ' for anomaly detection. If the confidence score (Q-value) is smaller than λ, the input session image will be determined as the 'anomaly' attack. Conversely, the class that belongs to the max confidence score is the expected detection result. Equation (3) shows the mathematical formulation of the anomaly detection module.

$$y = \begin{cases} \underset{a}{argmax}\ Q(s,a) & if\ max\ Q(s,a) > \lambda \\ anomaly & if\ max\ Q(s,a) < \lambda \end{cases} \tag{3}$$

4. The Proposed IDS Framework at Flow-Level

In flow-level intrusion detection, the traffic flow characteristics, which usually contain numerous packets, are extracted for detecting attacks [4–7]. Flow information includes the statistics of a flow, such as the number of packets, the duration, the average packet size, etc. A 5-tuple knowledge defines a flow that includes source IP, source port, transportation protocol, destination IP and destination port [41]. The number of packets in the flow is a valuable feature for flow-level IDS. The work in [4] contains evidence that Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks tend to transmit a large number of packets in a short time. The flow duration, the average packet size in the flow, and the transportation protocol can also be considered important features. Recently, many datasets have been collected and published for flow-level intrusion detection research.

The whole IDS framework at the flow level is shown in Table 5. The overall architecture is quite similar to the one proposed in Table 1 with some minor modifications in two modules, i.e., preprocessing and RL module.

Table 5. Intrusion Detection Framework at Flow-level.

Module	Sub-Module	Function
Preprocessing	None	Data Transformation and Feature Engineering
Reinforcement Learning	Interactive Training	Store batch data into replay buffer Train the agent and the sample agent
Exploration	None	A sub-module of Interaction Module

4.1. Preprocessing Module

The complete data preprocessing module is shown in Table 6. The dataset collected for flow-level research always involves some discrete and categorical features, such as protocols and packet size.

Table 6. The Preprocessing Module at Flow-level.

Step	Input	Method	Output
1	Raw Dataset	Encoding	Encoded Dataset
2	Encoded Dataset	Normalization	Applicable Dataset
3	Applicable Dataset	Feature learning with SAE	SAE's Encoder

In step 1, the discrete and categorical features are converted into continuous features, which can be processed by DNNs. With regard to the transformation of categorical features, the most simple and common approach is one-hot encoding. One-hot encoding has many advantages over other encoding approaches, one of which being the easy implementation. For the transformation of discrete features, an N-bit binary encoding approach can be used. The value of N, which ensures that N-bit binaries can encode all values of this discrete feature, is chosen based on the maximum value of the feature.

In step 2, max–min normalization is performed on the encoded dataset, converting all values to [0, 1]. Notably, after implementing one-hot encoding and binary encoding, data dimension considerably increases. A single hidden layer Stacked Auto Encoder (SAE) is then used with this encoded dataset for dimension reduction in the final step to counter this problem. The SAE-based approach helps in conducting dimension reduction and feature extraction. The SAE model is pre-trained, and then the encoder is extracted as the primary structure of the RL agent, as shown in Table 7.

Table 7. Stacked Autoencoder (Encoder and Decoder), Agent and Sample Agent. The encoder is shared with agent and sample agent.

Stacked Autoencoder	
Encoder	**Decoder**
InputLayer→ HiddenLayer→ LatentSpace→	HiddenLayer→ OutputLayer
Copy encoder to the first three layers of the agent and the sample agent	
Agent	
InputLayer→ HiddenLayer→ LatentSpace→	FCN $\xrightarrow{Softmax}$ Q-values
Sample Agent	
InputLayer→ HiddenLayer→ LatentSpace→	FCN $\xrightarrow{Softmax}$ $\overline{Q}$-values

4.2. Reinforcement Learning Module

Table 8 describes the important elements of the RL module of flow-level IDS. For brevity, only the parts that are different from the RL module of packet-level IDS are detailed in this section.

Table 8. RL Module for IDS Framework at the Flow Level.

RL	Symbol	IDS	IDS Space
State	s	Feature Vector	Traffic Feature
Action	a	Prediction	Label Space {0, 1, 2, . . .}
Reward	r	Reward Mechanism	{+1, −1}
Episode	–	User Defined	–
Agent	–	Classifier	–

The action represents the prediction performed by the agent. Furthermore, the flow level reward mechanism is the same as that of the packet level. However, the state and episode for the RL module, in the case of flow-level IDS, are defined differently. For the packet level, 'pcap' files are separated into different sessions; then, through image embedding, these sessions are transformed into images, which are considered as the states. However, network traffic features provided by the dataset are used directly as the states at the *flow level*. Moreover, at the packet level, packets embedded in the image and images in a session are arranged in chronological order (capture time). Thus, a session is viewed as an episode. Nonetheless, the data collected in the dataset has been shuffled at the *flow level*, so there is no apparent chronological order. The episode's length needs to be determined and is assumed to be fixed in the interaction process. Furthermore, the main structure of the update module is a Fully Connected Neural Network (FCN).

In the proposed RL module of flow-level IDS, in addition to the detection agent, we also design a sample agent to facilitate the adversarial training. The agent performs the intrusion class's correct prediction (the action) by achieving maximum rewards. The sample agent provides guidance (the action) for the next class to be sampled from. To improve the variability, the sample agent tends to counteract the agent. It chooses a class that is most likely to be misclassified and suggests it as the class to be sampled from for the next state.

For this reason, the reward feedback of the agent and the sample agent is the opposite. If the agent's prediction is wrong, the reward feedback of the sample agent is 1; otherwise, if the agent's prediction is correct, the reward feedback of the sample agent is -1: this way, the sample agent functions as the adversarial training agent. The objective of the sample agent is to ensure that those state-action pairs with high classification error rates can be adequately trained.

In the interaction stage, we should create a simulation environment. We take the preprocessed dataset as the simulation environment. In our experiments, we focus on episodic tasks; hence, we fix the length of an episode in advance. The complete interaction module is shown in Figure 5. All of the traffic features collected in the dataset can be considered as the states. In the beginning, a state is randomly sampled from the dataset. Feeding the state into the agent (classifier), the agent then outputs the prediction (action) of the current state. Feeding the action-label pair into the reward mechanism, we can obtain the reward. If the current state is not reaching the end of the present episode, we also feed the current state to the sample agent and obtain the next sample class. Afterward, we sample the next state that belongs to this class from the dataset. Next, we store the state, action, reward and next state in the replay buffer. Subsequently, treat the next state as the current state and repeat the above process. It should be noted that if the current state is reaching the end of the episode, we store state, action, reward into the replay buffer and then randomly sample a state from the dataset, which indicates that a new episode is launched.

The flow level IDS training approach is identical to that illustrated in Figure 4. The SAE-based feature extractor and the overall network design for the update and target networks are shown in Table 7. It should be emphasized that the agent and the sample agent must be trained at the same time.

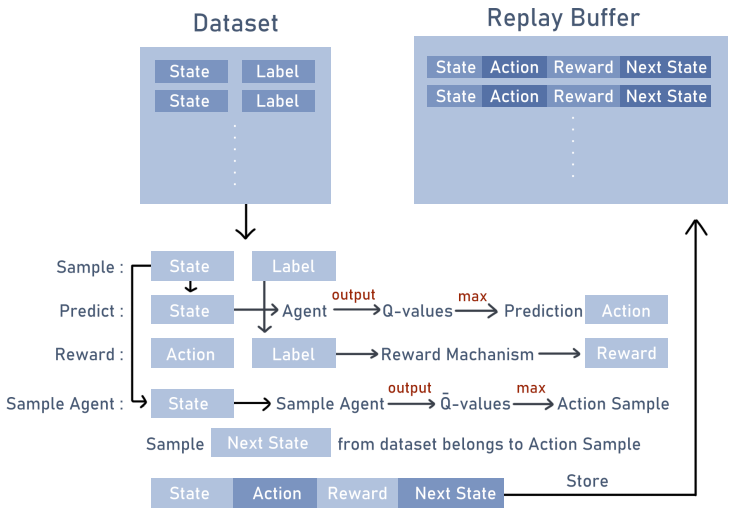

Figure 5. Interaction Module.

4.3. The Exploration Module

Most datasets collected for intrusion detection are unbalanced. This is because the vast majority of traffic in the real world is normal traffic, so it is easier to collect normal traffic than malignant traffic. In RL, the exploration module is conductive to solve these class imbalance problems. In the proposed flow-level IDS, both the ε-greedy policy and Conditional Generative Adversarial Network (CGAN) are used for the exploration. Equation (4) shows the ε-greedy policy applied to the sample agent, where ε controls the exploration degree.

$$a = \begin{cases} \underset{a}{argmax}\ \bar{Q}(s,a) & with\ probability\ of\ 1-\varepsilon \\ any\ action & with\ probability\ of\ \varepsilon \end{cases} \tag{4}$$

The purpose of CGAN is to generate some novel attack flows for each class, which will augment the fixed dataset. The CGAN exploration rate σ controls the extent of the exploration. The CGAN takes the class label and noise as the input and outputs a state that belongs to each class. The architecture of both the generator and discriminator is shown in Figure 6. The generator's functionality is to generate simulated states that can deceive the discriminator.

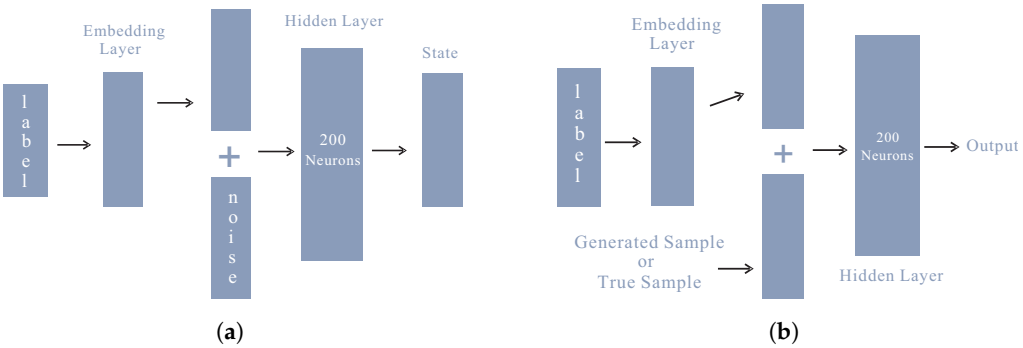

Figure 6. CGAN Architecture (**a**) Generator, (**b**) Discriminator.

The output of the discriminator is a numerical value ranging in [0, 1]. An output threshold of 0.5 was set to differentiate between the true and generated samples. If the output of the discriminator is greater than this threshold, the input label is marked as a true sample; otherwise, if the output is less than the pre-defined threshold, the input sample is marked as a generated or augmented sample.

The quality of the generated samples is assured by training an additional sample classifier for the dataset composed of generated samples. Only the generated samples correctly classified by the classifier are retained as generated samples to augment the overall dataset. The complete algorithm of Deep Q-Learning with an exploration module is shown in Algorithm 3.

Algorithm 3 Deep Q-Learning Framework for Detection at Packet-level

procedure DEEP Q-LEARNING
  Conduct preprocessing on dataset
  Pretrain a stacked autoencoder
  Train the CGAN
  Initialize Q function for agent, set target Q function $\overline{Q} = Q$
  Initialize the agent replay buffer and the sample agent replay buffer for training the agent and the sample agent, respectively
  Copy parameters to Q and $\overline{Q}$ from the encoder
  Set target Q-function $\widehat{Q} = Q$, $\widehat{\overline{Q}} = \overline{Q}$
  for each episode **do**
    Randomly choose s_0 from the dataset
    for each sample within the episode, $t \in [0, N]$ **do**
      Given state s_t , take action a_t based on Q
      Given state s_t , take action $\bar{Q}_t$ based on $\overline{Q}$ $(\varepsilon - greedy)$
      Compare a_t with the true label, obtain the reward r_t and $\bar{r}_t$
      Derive the next state: choose s_{t+1} for which the true label is $\bar{a}_t$ from dataset or CGAN, with a probability of 1-σ and σ, respectively
      Store s_t, a_t, r_t, s_{t+1} into the agent replay buffer
      Store $s_t, \bar{a}_t, \bar{r}_t, s_{t+1}$ into the sample agent replay buffer
      Sample a batch $s_t, a_t, r_t, s_{t+1}$from the agent replay buffer
      Sample a batch $s_t, \bar{a}_t, \bar{r}_t, s_{t+1}$from the sample agent replay buffer
      Target $y = r_t + \gamma \underset{a}{max} \widehat{Q}(s_{t+1}, a)$
      Update Q through back propagation to make $Q(s_t, a_t)$ close to y
      Target $\overline{y} = \bar{r}_t + \gamma \underset{a}{max} \widehat{\overline{Q}}(s_{t+1}, a_t)$
      Update $\overline{Q}$ through back propagation to make $\overline{Q}(s_t, \bar{a}_t)$ close to $\overline{y}$
      For each C steps reset $\widehat{Q} = Q, \widehat{\overline{Q}} = \overline{Q}$
    end for
  end for
end procedure

5. Experimental Results

5.1. Dataset

CICDDoS2019 is one of the latest datasets collected for the Distributed Denial of Service (DDoS) attack and has been widely used for intrusion detection research [42] since its publication. In a DDoS attack, the idea is to attack a single network or a machine from different locations. Various attackers can cause this attack from a different location, or a single attacker can take control of multiple machines in different locations and use them to attack victims simultaneously.

Eight different traffic types are used in the training set, namely, Normal, PortMap, NetBIOS, LDAP, MSSQL, UDP, UDP-Lag, and SYN. After the image embedding, a sample image for each of the eight classes is shown in Figure 7. The test set contains 12 types of traffic. In addition to the eight types collected in the training set, the other four types, including NTP, DNS, WebDDoS, and TFTP, are included in the test set. These additional traffic types help in the generality of the IDS because they are blind to IDS at the training stage. In the proposed work, an anomaly detection model detects these four additional attacks by considering them as an anomaly class.

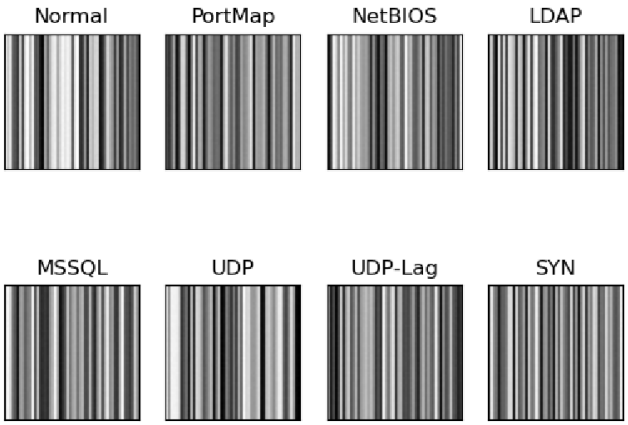

Figure 7. Images after image embedding for each class.

5.2. Results of the Proposed Packet-Level IDS

To evaluate the proposed packet-level IDS, two agents based on DQN-CNN and DQN-1D-CNN are trained. Another two agents, PG-CNN and PG-1D-CNN, are trained with the policy gradient learning method for comparison. The dataset is split into a ratio of 4:1 for the training and validation set. During the partition, the training and validation set's session length distribution is consistent, thus, allowing the image ratio to be approximately 4:1. After the partition, the statistics of the dataset are shown in Table 9.

Table 9. Training and Validation Set Details.

Traffic	Tranining Images	Validation Images	Class
Normal	25,012	6256	0
PortMap	20,200	5058	1
NetBIOS	23,652	5912	2
LDAP	18,019	5083	3
MSSQL	22,986	4709	4
UDP	19,159	6051	5
UDP-Lag	19,552	5198	6
SYN	22,640	4970	7

In the experiment, we compare three reward discount values (γ), namely 0.1, 0.5 and 0.9, on four agents. Then by using the value of accuracy averaged over each class on the validation set to evaluate the performance and select the optimal discount value. The results are shown in Table 10. We notice that when γ equals 0.1, all of the four agents attain the highest performance. With regard to reinforcement learning algorithms, DQN-1D-CNN slightly outperforms PG-1D-CNN over all discount factors, and DQN-CNN also marginally outperforms PG-CNN over all discount factors. Meanwhile, it is observed that PG-based agents are more sensitive to discount factors. Overall results indicate that Deep Q-Learning is a relatively better algorithm than policy gradient methods in our experiments. Hence, in the following case studies, we assume that the discount factor value is fixed as 0.1 and mainly adopt Deep Q-Learning algorithms for the agent.

Detailed experiment results of intrusion detection based on using DQN-1D-CNN with a discount value equal to 0.1 are given in Table 11. As shown in the table, our detection system can reach high accuracy at 98.78%. With respect to each class, 100% of normal traffic can be detected. The detection rate (Recall) of MSSQL is the least among all eight types, but still reaching 97.30%. Table 11 also lists the validation results of DQN-CNN. It is seen that the performance of DQN-CNN is slightly worse than that of DQN-1D-CNN, but still reaches a high accuracy of 96.07%. The experimental results prove that both DQN-1D-CNN and DQN-CNN can achieve high performances when dealing with image tasks. In addition, in our experiments, the image can also be readily treated as the time series

data, and DQN-1D-CNN achieves better performance in this case. In addition, the average computation times for intrusion detection based on DQN-CNN and DQN-1D-CNN are also given.

Table 10. Performances of different discount values on four agents.

Agent	Discount Factor	Validation Accuracy
DQN-1DCNN	**0.1**	**98.78%**
	0.5	95.21%
	0.9	87.12%
DQN-CNN	0.1	96.07%
	0.5	93.22%
	0.9	86.24%
PG-1DCNN	0.1	96.17%
	0.5	90.15%
	0.9	79.67%
PG-CNN	0.1	96.03%
	0.5	87.12%
	0.9	70.69%

Table 11. Validation results of DQN-1DCNN with discount factor value = 0.1.

Agent	Traffic	Accuracy	Precision	Recall	F1-Score
DQN-1DCNN	Normal	98.78%	99.32%	100%	99.66%
	PortMap		98.04%	98.95%	98.49%
	NetBIOS		98.87%	99.26%	99.06%
	LDAO		98.85%	98.37%	98.61%
	MSSQL		98.47%	97.30%	97.86%
	UDP		98.95%	99.62%	99.26%
	UDP-Lag		98.02%	98.31%	98.61%
	SYN		98.64%	97.83%	98.23%
DQN-CNN	Normal	96.07%	96.91%	99.68%	98.27%
	PortMap		95.67%	93.53%	94.59%
	NetBIOS		97.35%	95.75%	96.55%
	LDAO		95.76%	96.95%	96.35%
	MSSQL		95.06%	96.03%	95.54%
	UDP		95.32%	96.00%	95.66%
	UDP-Lag		95.05%	94.92%	94.99%
	SYN		97.14%	94.87%	95.71%

The averaged computation time for DQN-1DCNN detection is 0.112 s. The averaged computation time for DQN-CNN based detection is 0.071 s.

Afterward, we evaluate our DQN-1D-CNN agent on the test set. In this stage, we are required to set another important hyperparameter: λ, the threshold that controls the detection performance of (unknown) anomaly traffic. We treat the eight known types of attacks as the negative class, and the anomaly traffic as the positive class. We use precision and recall to evaluate the performances and select the optimal λ. Results are shown in Table 12. As λ is increasing, the precision is decreasing, which means that an increasing volume of network traffic is classified as the anomaly type. Conversely, as λ is increasing, the recall is also increasing, meaning that an increasing amount of anomaly traffic is detected. We can determine the value of λ based on the actual requirements of the intrusion detection system. In our experiments, we fix the λ as 0.7, which is a trade-off between 0.5 and 0.9.

Table 12. Anomaly Detection λ Selection.

λ	Precision	Recall
0.5	**93.94%**	19.17%
0.7	85.13%	71.05%
0.9	43.21%	**86.86%**

Finally, we evaluate our intrusion detection system on the test set with $\gamma = 0.1$ and $\lambda = 0.7$. Results are shown in Table 13. Due to the existence of anomaly traffics, the general accuracy has declined greatly to 84.27% compared with the validation accuracy at 96.07%. With respect to all classes, the DQN-1D-CNN agent can attain relatively high performances on those known types. With regard to anomaly types, although these anomaly types are completely blind to our system, 71.05% of them can still be detected.

Table 13. Test results of DQN-1D-CNN with discount value $\gamma = 0.1$ and $\lambda = 0.7$. The averaged computation time for detection is **0.143 s**.

Traffic	Accuracy	Precision	Recall	F1-Score
Normal		88.81%	93.64%	91.16%
PortMap		86.83%	91.38%	89.05%
NetBIOS		84.62%	90.69%	87.55%
LDAO		82.36%	88.31%	85.23%
MSSQL	84.27%	88.24%	92.14%	90.14%
UDP		83.55%	89.40%	86.38%
UDP-Lag		82.07%	87.92%	84.90%
SYN		84.67%	89.57%	87.05%
Anomaly		81.12%	71.05%	75.71%

5.3. Results of the Proposed Flow-Level IDS

Similarly, we conduct preprocessing on DDoS2019 at first. After transformation and normalization, the dimension of the state vector is 97. The structure of the stacked autoencoder is shown in Table 7. The dimension of the input layer, hidden layer and latent space is 97, 60, 25, respectively. The dimension of the fully connected layer is eight, equal to the number of classes. Figure 8 shows the performance of the stacked autoencoder, which is excellent on DDoS2019.

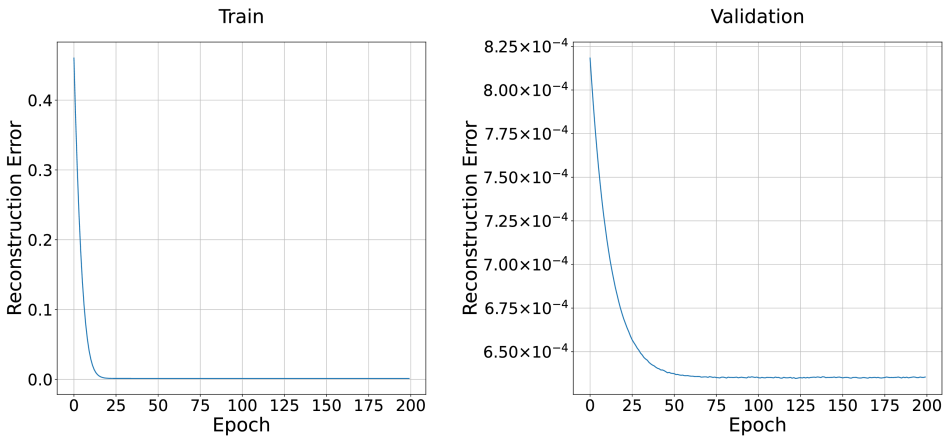

Figure 8. Performances of stacked autoencoder on DDoS2019.

In the first stage, we explore the influence of ε and γ when designing the agent and the sample agent. We also fix the length of the episode as 512. The experiment results are shown in Figures 9 and 10. We create a search space (0, 0.2) for both ε and γ.

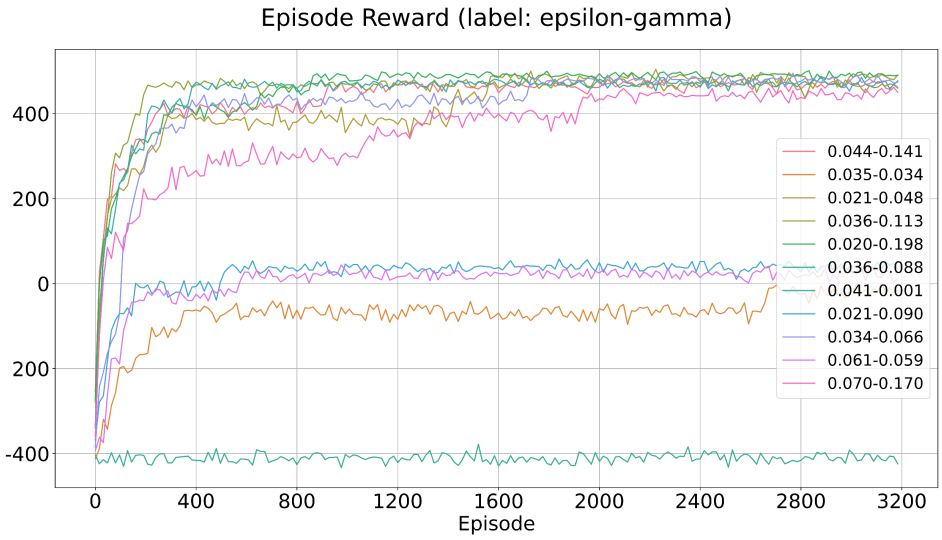

Figure 9. Performances on the validation set on DDoS2019 (Episode Reward). The averaged computation time for training on each episode is 6.912 s.

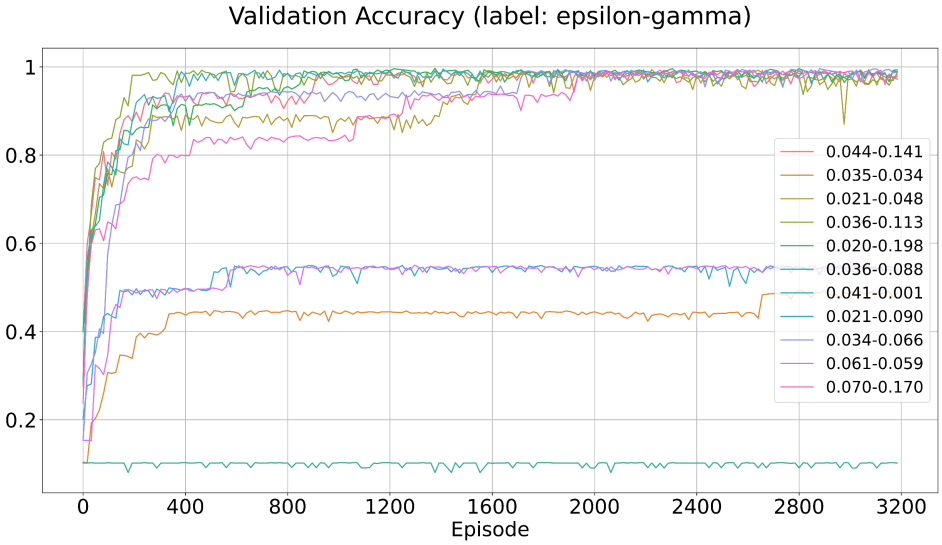

Figure 10. Performanceson the validation set on DDoS2019 (Validation Accuracy).

From the plots, we find that despite some anomaly pairs generating worse performances, most pairs associated with relatively small ε and γ can reach high accuracy and obtain high episode reward. We extract some statistics from performance figures and include them in Table 14. The optimal pair is ε-0.034, γ-0.066, reaching 98.86% on the validation set. The worst pair is ε-0.036, γ-0.088. It is observed that a small variance of parameters can have a significant impact on the performance.

Table 14. Experiment results (averaged over the last 10 episodes) without CGAN.

ε	γ	Mean Episode Reward	Mean Validation Accuracy
0.034	**0.066**	**477.3**	**98.86%**
0.041	0.001	471.3	98.48%
0.070	0.170	449.8	97.73%
0.061	0.059	22.2	54.47%
0.036	0.088	–410.6	10.10%

Furthermore, we create another two search spaces for the pair ε and γ and run the Bayesian search algorithm for the optimal results, which are shown in Table 15. The results inform us that a large γ is not appropriate for our intrusion detection system. If we adjust γ to a relatively small value, we can ensure that the intrusion detection system can always achieve good performance.

Table 15. Performances (averaged over the last 10 episodes) on validation set. (a, b) denotes the search space for the Bayesian search algorithm.

ε	γ	Mean Episode Reward	Mean Validation Accuracy
0.034 (0, 0.2)	**0.066 (0, 0.2)**	**477.3**	**98.86%**
0.062 (0, 0.2)	0.529 (0.4, 0.6)	462.9	96.82%
0.102 (0, 0.2)	0.804 (0.8, 1)	40.9	54.22%
0.596 (0.4, 0.6)	0.015 (0, 0.2)	472.6	98.49%

In the second stage, we further strengthen the IDS with CGAN for exploration and data augmentation. Experiment results on the test set are shown in Table 16. Using CGAN, the IDS attains better performance, improved from 93.67% to 96.43%. It proves that CGAN can simulate a more realistic network environment, which leads to performance enhancement.

Table 16. Performances on test set with RL-CGAN.

Exploration Rate	Traffic	Accuracy	Precision	Recall	F1-Score
0	Normal	93.67%	96.23%	95.52%	95.87%
	PortMap		93.79%	93.96%	93.88%
	NetBIOS		92.14%	96.12%	94.09%
	LDAO		92.53%	92.72%	92.63%
	MSSQL		91.48%	95.50%	93.44%
	UDP		93.67%	89.98%	91.79%
	UDP-Lag		93.92%	91.70%	92.79%
	SYN		95.79%	93.82%	94.80%
0.5	Normal	96.43%	98.11%	97.54%	97.82%
	PortMap		96.45%	97.14%	96.79%
	NetBIOS		95.45%	98.20%	96.81%
	LDAO		95.98%	95.44%	95.71%
	MSSQL		94.86%	96.76%	95.80%
	UDP		96.52%	93.68%	95.08%
	UDP-Lag		96.75%	95.76%	96.25%
	SYN		97.29%	96.80%	97.04%

The averaged computation time for detection is **0.059 s** with '0' Exploration Rate. The averaged computation time for detection is **0.083 s** with '0.5' Exploration Rate.

5.4. Comparison and Discussion

In this section, the proposed RL-based IDS design is compared with some other commonly used ML algorithms, including random forest, support vector machine, AdaBoost [43], FCN, LSTM, CNN and 1DCNN on DDoS2019. Results are shown in Table 17, quantifying the performance of the proposed IDS for packet and flow level in comparison with other ML algorithms.

Considering flow-level approaches, ensemble methods, including random forest and AdaBoost, achieve great performances on DDoS2019. SVM is the worst one among all models, and it takes the longest time to train. Neural networks, including fully connected neural networks and LSTM, also attain great performances. The proposed RL-CGAN flow-level approach still outperforms all other flow-level models listed in Table 17 without increasing computation time.

Table 17. Comparison with different models.

Algorithm	Level	Accuracy	Precision	Recall	F1 Score	Detection Time
Random Forest	Flow	95.42%	96.72%	94.12%	95.40%	1.751 s
Adaboost	Flow	94.79%	93.98%	95.11%	94.54%	2.610 s
SVM	Flow	90.12%	92.15%	88.98%	90.54%	10.019 s
3-Layer FCN	Flow	94.12%	93.09%	95.24%	94.15%	0.034 s
3-Layer LSTM	Flow	93.90%	94.01%	91.99%	92.93%	0.192 s
RL-CGAN	Flow	96.42%	96.43%	96.42%	96.41%	0.092 s
1D-CNN	Packet	92.12%	94.12%	90.43%	92.29%	0.096 s
CNN	Packet	90.68%	91.36%	93.10%	91.84%	0.084 s
DQN-1DCNN	Packet	**97.69%**	**98.10%**	**96.65%**	**97.14%**	0.101 s
DQN-CNN	Packet	95.19%	95.94%	96.10%	96.01%	**0.079 s**

For packet-level approaches, the comparison of the proposed approach with conventional 1D-CNN and CNN approaches (without RL) is conducted. It can be seen that conventional deep learning methods can also achieve relatively high performance on the test set, which suggests that image embedding is a useful data preprocessing approach for raw network traffic analysis. Most importantly, the proposed RL-based approach is significantly better in terms of generalizability when compared with these traditional deep learning approaches.

Among all models (including both flow-level and packet-level), the proposed DQN-1DCNN at the packet level is the best one for all performance measures. By comparing it to RL-CGAN at the flow level, it should be noted that flow information is incorporated by the DQN-1DCNN agent through packet features, while RL-CGAN only utilizes flow features provided by the dataset. This is the main reason why DQN-1DCNN outperforms RL-CGAN in this comparison study.

6. Conclusions and Future Work

In this paper, an RL-based IDS at the packet level and flow level is proposed. With the help of image embedding, network traffic is transformed into images processed by CNN and 1D-CNN for the detection task at the packet level. In addition, unknown network traffic can be detected through an anomaly detection module. An enhanced RL-based IDS with an exploration module at the flow level is also designed. The experimental results verify the superior performance of the proposed packet-level and flow-level IDS with the most commonly used state-of-the-art algorithms. However, further research works can be undertaken based on the following future directives.

- GAN can be introduced for packet-level IDS to generate novel flows or sessions. Furthermore, we can use GAN to simulate a dynamic network environment for interaction.
- The idea of exploration can also be introduced into the packet-level reinforcement learning framework to employ the exploration policy for capturing more states in the interaction space.
- The proposed IDS at the packet level can detect the malignant image but can not identify the specific traffic session position embedded in the image. One possible solution is to use an N to N LSTM model for this problem.
- For flow-level IDS, to improve the generalizability of the RL agent, a new reward mechanism can be designed, where a more significant penalty can be assigned to the intrusion detection agent when some critical malignant traffics are not detected.
- Finally, to further validate the proposed IDS design, experiments can be conducted on different datasets with other types of attacks, and a thorough comparison can be performed with other RL-based IDS designs.

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

Funding: This research was funded by the Natural Science and Engineering Research Council (NSERC), Canada.

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

References

1. Smaha, S. Haystack: An intrusion detection system. In Proceedings of the [Proceedings 1988] Fourth Aerospace Computer Security Applications, Orlando, FL, USA, 12–16 September 1988; pp. 37–44.
2. Hwang, R.H.; Peng, M.C.; Nguyen, V.L.; Chang, Y.L. An LSTM-based deep learning approach for classifying malicious traffic at the packet level. *Appl. Sci.* **2019**, *9*, 3414. [CrossRef]
3. Ge, M.; Fu, X.; Syed, N.; Baig, Z.; Teo, G.; Robles-Kelly, A. Deep learning-based intrusion detection for iot networks. In Proceedings of the 2019 IEEE 24th Pacific Rim International Symposium on Dependable Computing (PRDC), Kyoto, Japan, 1–3 December 2019; pp. 256–265.
4. Doshi, R.; Apthorpe, N.; Feamster, N. Machine learning ddos detection for consumer internet of things devices. In Proceedings of the 2018 IEEE Security and Privacy Workshops (SPW), San Francisco, CA, USA, 24 May 2018; pp. 29–35.
5. Radford, B.; Apolonio, L.; Trias, A.; Simpson, J. Network Traffic Anomaly Detection Using Recurrent Neural Networks. *arXiv* **2018**, arXiv:1803.10769.
6. Choraś, M.; Pawlicki, M. Intrusion detection approach based on optimised artificial neural network. *Neurocomputing* **2021**, *452*, 705–715. [CrossRef]
7. Ieracitano, C.; Adeel, A.; Morabito, F.C.; Hussain, A. A novel statistical analysis and autoencoder driven intelligent intrusion detection approach. *Neurocomputing* **2020**, *387*, 51–62. [CrossRef]
8. Goodfellow, I.; Bengio, Y.; Courville, A.; Bengio, Y. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016; Volume 1.
9. Bodeau, D.; Graubart, R. *Cyber Resiliency Design Principles Selective Use Throughout the Lifecycle and in Conjunction with Related Disciplines*; Technical Report; MITRE CORP: Bedford, MA, USA, 2017.
10. Beal, M.; Ghahramani, Z.; Rasmussen, C. The infinite hidden Markov model. In Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 3–8 December 2001.
11. Toyoshima, K.; Oda, T.; Hirota, M.; Katayama, K.; Barolli, L. A DQN based mobile actor node control in WSAN: Simulation results of different distributions of events considering three-dimensional environment. In Proceedings of the International Conference on Emerging Internetworking, Data & Web Technologies, Kitakyushu, Japan, 24–26 February 2020; pp. 197–209.
12. Saito, N.; Oda, T.; Hirata, A.; Hirota, Y.; Hirota, M.; Katayama, K. Design and implementation of a DQN based AAV. In Proceedings of the International Conference on Broadband and Wireless Computing, Communication and Applications, Yonago, Japan, 28–30 October 2020; pp. 321–329.
13. Alavizadeh, H.; Hong, J.B.; Kim, D.S.; Jang-Jaccard, J. Evaluating the effectiveness of shuffle and redundancy mtd techniques in the cloud. *Comput. Secur.* **2021**, *102*, 102091. [CrossRef]
14. Sethi, K.; Kumar, R.; Mohanty, D.; Bera, P. Robust adaptive cloud intrusion detection system using advanced deep reinforcement learning. In Proceedings of the International Conference on Security, Privacy, and Applied Cryptography Engineering, Kolkata, India, 17–21 December 2020; pp. 66–85.
15. Sethi, K.; Kumar, R.; Prajapati, N.; Bera, P. Deep reinforcement learning based intrusion detection system for cloud infrastructure. In Proceedings of the 2020 International Conference on COMmunication Systems & NETworkS (COMSNETS), Bengaluru, India, 7–11 January 2020; pp. 1–6.
16. Sethi, K.; Sai Rupesh, E.; Kumar, R.; Bera, P.; Venu Madhav, Y. A context-aware robust intrusion detection system: A reinforcement learning-based approach. *Int. J. Inf. Secur.* **2020**, *19*, 657–678. [CrossRef]
17. Cappart, Q.; Moisan, T.; Rousseau, L.M.; Prémont-Schwarz, I.; Cire, A.A. Combining reinforcement learning and constraint programming for combinatorial optimization. In Proceedings of the AAAI Conference on Artificial Intelligence, Virtual, 2–9 February 2021; Volume 35, pp. 3677–3687.
18. Ma, X.; Shi, W. Aesmote: Adversarial reinforcement learning with smote for anomaly detection. *IEEE Trans. Netw. Sci. Eng.* **2020**, *8*, 943–956. [CrossRef]
19. Lopez-Martin, M.; Carro, B.; Sanchez-Esguevillas, A. Application of deep reinforcement learning to intrusion detection for supervised problems. *Expert Syst. Appl.* **2020**, *141*, 112963. [CrossRef]
20. Stefanova, Z.S.; Ramachandran, K.M. Off-policy q-learning technique for intrusion response in network security. *Int. J. Comput. Inf. Eng.* **2018**, *12*, 266–272.
21. François-Lavet, V.; Henderson, P.; Islam, R.; Bellemare, M.G.; Pineau, J. An introduction to deep reinforcement learning. *Found. Trends Mach. Learn.* **2018**, *11*, 219–354. [CrossRef]

22. Hu, B.; Li, J. Shifting deep reinforcement learning algorithm toward training directly in transient real-world environment: A case study in powertrain control. *IEEE Trans. Ind. Inform.* **2021**, *17*, 8198–8206. [CrossRef]
23. Sethi, K.; Madhav, Y.V.; Kumar, R.; Bera, P. Attention based multi-agent intrusion detection systems using reinforcement learning. *J. Inf. Secur. Appl.* **2021**, *61*, 102923. [CrossRef]
24. Nguyen, T.T.; Reddi, V.J. Deep reinforcement learning for cyber security. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, 1–17. [CrossRef]
25. Caminero, G.; Lopez-Martin, M.; Carro, B. Adversarial environment reinforcement learning algorithm for intrusion detection. *Comput. Netw.* **2019**, *159*, 96–109. [CrossRef]
26. Tavallaee, M.; Bagheri, E.; Lu, W.; Ghorbani, A.A. A detailed analysis of the KDD CUP 99 data set. In Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, Ottawa, ON, Canada, 8–10 July 2009; pp. 1–6.
27. Kolias, C.; Kambourakis, G.; Stavrou, A.; Gritzalis, S. Intrusion detection in 802.11 networks: Empirical evaluation of threats and a public dataset. *IEEE Commun. Surv. Tutor.* **2015**, *18*, 184–208. [CrossRef]
28. Iannucci, S.; Barba, O.D.; Cardellini, V.; Banicescu, I. A performance evaluation of deep reinforcement learning for model-based intrusion response. In Proceedings of the 2019 IEEE 4th International Workshops on Foundations and Applications of Self* Systems (FAS*W), Umea, Sweden, 16–20 June 2019; pp. 158–163.
29. Iannucci, S.; Cardellini, V.; Barba, O.D.; Banicescu, I. A hybrid model-free approach for the near-optimal intrusion response control of non-stationary systems. *Future Gener. Comput. Syst.* **2020**, *109*, 111–124. [CrossRef]
30. Malialis, K.; Kudenko, D. Distributed response to network intrusions using multiagent reinforcement learning. *Eng. Appl. Artif. Intell.* **2015**, *41*, 270–284. [CrossRef]
31. Holgado, P.; Villagrá, V.A.; Vazquez, L. Real-time multistep attack prediction based on hidden markov models. *IEEE Trans. Dependable Secur. Comput.* **2017**, *17*, 134–147. [CrossRef]
32. Zhang, Z.; Naït-Abdesselam, F.; Ho, P.H.; Kadobayashi, Y. Toward cost-sensitive self-optimizing anomaly detection and response in autonomic networks. *Comput. Secur.* **2011**, *30*, 525–537. [CrossRef]
33. Fessi, B.A.; Benabdallah, S.; Boudriga, N.; Hamdi, M. A multi-attribute decision model for intrusion response system. *Inf. Sci.* **2014**, *270*, 237–254. [CrossRef]
34. Alavizadeh, H.; Alavizadeh, H.; Jang-Jaccard, J. Deep Q-Learning based Reinforcement Learning Approach for Network Intrusion Detection. *Computers* **2022**, *11*, 41. [CrossRef]
35. Kim, C.; Park, J. Designing online network intrusion detection using deep auto-encoder Q-learning. *Comput. Electr. Eng.* **2019**, *79*, 106460. [CrossRef]
36. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient estimation of word representations in vector space. In Proceedings of the Workshop at International Conference on Learning Representations (ICLR), Scottsdale, AZ, USA, 2–4 May 2013
37. Forestiero, A.; Papuzzo, G. Agents-Based Algorithm for a Distributed Information System in Internet of Things. *IEEE Internet Things J.* **2021**, *8*, 16548–16558. [CrossRef]
38. Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; Zaremba, W. Openai gym. *arXiv* **2016**, arXiv:1606.01540.
39. Dhariwal, P.; Hesse, C.; Klimov, O.; Nichol, A.; Plappert, M.; Radford, A.; Schulman, J.; Sidor, S.; Wu, Y.; Zhokhov, P.; et al. OpenAI Baselines, a Set of High-Quality Reinforcement Learning Algorithms. 2017. Available online: https://github.com/openai/baselines (accessed on 24 November 2022).
40. Hasselt, H. Double Q-learning. In Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 6–9 December 2010; Volume 23, pp. 2613–2621.
41. Wang, W.; Zhu, M.; Wang, J.; Zeng, X.; Yang, Z. End-to-end encrypted traffic classification with one-dimensional convolution neural networks. In Proceedings of the 2017 IEEE International Conference on Intelligence and Security Informatics (ISI), Beijing, China, 22–24 July 2017; pp. 43–48.
42. Sharafaldin, I.; Lashkari, A.H.; Hakak, S.; Ghorbani, A.A. Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy. In Proceedings of the 2019 International Carnahan Conference on Security Technology (ICCST), Chennai, India, 1–3 October 2019; pp. 1–8.
43. Schapire, R.E. Explaining AdaBoost. In *Empirical Inference: Festschrift in Honor of Vladimir N. Vapnik*; Schölkopf, B., Luo, Z., Vovk, V., Eds.; Springer: Berlin/Heidelberg, Germany, 2013; pp. 37–52. [CrossRef]