mathematics

MDPI

Article

Intrusion Detection Based on Adaptive Sample Distribution Dual-Experience Replay Reinforcement Learning

Haonan Tan, Le Wang *, Dong Zhu and Jianyu Deng

Cyberspace Institute of Advanced Technology, Guangzhou University, Guangzhou 510006, China; hntan@e.gzhu.edu.cn (H.T.); zhudong@e.gzhu.edu.cn (D.Z.); dengjianyu@e.gzhu.edu.cn (J.D.)
* Correspondence: wangle@gzhu.edu.cn

Abstract: In order to cope with ever-evolving and increasing cyber threats, intrusion detection systems have become a crucial component of cyber security. Compared with signature-based intrusion detection methods, anomaly-based methods typically employ machine learning techniques to train detection models and possess the capability to discover unknown attacks. However, intrusion detection methods face the challenge of low detection rates for minority class attacks due to imbalanced data distributions. Traditional intrusion detection algorithms address this issue by resampling or generating synthetic data. Additionally, reinforcement learning, as a machine learning method that interacts with the environment to obtain feedback and improve performance, is gradually being considered for application in the field of intrusion detection. This paper proposes a reinforcement-learning-based intrusion detection method that innovatively uses adaptive sample distribution dual-experience replay to enhance a reinforcement learning algorithm, aiming to effectively address the issue of imbalanced sample distribution. We have also developed a reinforcement learning environment specifically designed for intrusion detection tasks. Experimental results demonstrate that the proposed model achieves favorable performance on the NSL-KDD, AWID, and CICIoT2023 datasets, effectively dealing with imbalanced data and showing better classification performance in detecting minority attacks.

Keywords: cyber security; intrusion detection; reinforcement learning

MSC: 68T42

Citation: Tan, H.; Wang, L.; Zhu, D.; Deng, J. Intrusion Detection Based on Adaptive Sample Distribution Dual-Experience Replay Reinforcement Learning. *Mathematics* **2024**, *12*, 948. https://doi.org/10.3390/math12070948

Academic Editor: Cheng-Chi Lee

Received: 22 February 2024
Revised: 16 March 2024
Accepted: 19 March 2024
Published: 23 March 2024

1. Introduction

With the rapid development of network technologies, concepts such as smart homes, smart vehicles, and smart cities have become a reality, bringing more convenience and comfort to our daily lives. People are increasingly reliant on the Internet for information transfer and data processing [1–3]. However, the threat to cyber security is also on the rise, particularly with the proliferation of intrusion incidents. These attacks can lead to significant economic losses, personal privacy breaches, and damage to critical infrastructure [4]. Therefore, specialized intrusion detection systems have become a vital component of cyber security to safeguard against such threats [5].

Intrusion detection technology is designed to monitor and identify anomalous behavior within a network, enabling the early detection of intrusion activities through the monitoring and analysis of network traffic, system logs, and other relevant data. Research on intrusion detection can mainly be classified into two categories: signature-based intrusion detection and anomaly-based intrusion detection [6]. Signature-based intrusion detection, also known as rule-based intrusion detection, detects intrusion behavior through predefined rules or patterns. This technology is usually responsive to known attack types, requiring the maintenance of a rule or database to store and manage the required rules and patterns for detection. It also relies on experienced network security experts to create rules

and requires regular updates of the rule repository to ensure timeliness and performance in detection. With the rapid development and continuous evolution of network threats, the cost of maintaining the rule repository is increasing. Therefore, more and more research is considering anomaly-based intrusion detection. This type of approach typically employs machine learning techniques, whereby modeling and training detection models on network traffic, system logs, or other related data enables the detection of abnormal behaviors within the network. As this approach is not based on predefined rules and patterns, it has the capability to detect unknown attacks.

However, a major challenge in intrusion detection tasks is the imbalance of sample distribution, with a significantly larger number of normal samples compared with abnormal samples. Due to the large number of normal samples, it is difficult for the model to learn effective classification patterns from abnormal samples. Currently, common methods to address sample distribution imbalance include data resampling techniques [7,8], which involve increasing the number of abnormal samples or reducing the number of normal samples to balance the dataset. Another approach is ensemble learning [9,10], such as boosting and stacking, which compensates for the poor prediction of a single classifier on minority class attacks by combining predictions from multiple classifiers. Additionally, the use of generative adversarial networks to generate synthetic samples of minority classes [11,12] can increase the proportion of minority class attacks in the dataset, improving the model's generalization ability. Although these methods alleviate the impact of sample distribution imbalance on intrusion detection to some extent, they also have limitations. Data resampling techniques may lead to information loss, as the addition or removal of normal or abnormal samples can affect the true distribution of data in a real network. While ensemble learning enhances the robustness of intrusion detection models, it introduces computational and storage overhead in practical applications. Furthermore, the quality and diversity of the generated synthetic samples in generative adversarial networks are points of concern.

In addition to applying supervised or unsupervised learning methods to anomaly-based intrusion detection, reinforcement learning, a machine learning approach that involves interacting with the environment to gather feedback and improve, has become a new intrusion detection paradigm. Reinforcement learning has been successfully validated in various domains [13–18], these applications have showcased the remarkable capabilities and potential of reinforcement learning. In the field of intrusion detection, for example, Caminero et al. [19] proposed an intrusion detection model based on adversarial reinforcement learning, which integrates the supervised learning paradigm with reinforcement learning algorithms to form an intrusion detection framework. Lopez-Martin et al. [20] proposed a method of intrusion detection using labeled datasets based on various deep reinforcement learning (DRL) algorithms. They conceptually modified the DRL model, which is based on interaction with a live environment, and replaced the environment with a sampling function. Existing studies have provided insightful ideas for using reinforcement learning in intrusion detection, but there are still some challenges that have not been adequately addressed. For one thing, the current method's environment design is not sufficient. Reinforcement learning typically does not directly rely on existing datasets; instead, it requires training within an environment tailored to a specific task. This environment simulates the interaction between the agent and the real world, allowing the agent to learn the optimal actions or strategies through iteration and refinement within the environment. Therefore, designing a universal environment is an important condition for applying reinforcement learning to intrusion detection in different scenarios. And for other scenarios, the existing methods for handling imbalanced data are still not ideal. Reinforcement learning algorithms typically require a large number of samples for training to learn accurate strategies, which further exacerbates the impact of data imbalance and results in poor detection performance for minority class attacks.

In response to these challenges, this paper offers another feasible intrusion detection solution based on reinforcement learning. By integrating the designed adaptive sample

distribution dual-experience replay method to address the problem of data imbalance and developing a reinforcement learning environment specifically for intrusion detection, the proposed method enables reinforcement learning to be on par with supervised learning methods and provides a promising intrusion detection paradigm. In addition, labeled data offers a great pretraining sample for the strategy of reinforcement learning, which enables the cold start of crucial parameters through learning with labeled data. This provides reinforcement learning with an initial optimization space with good performance, providing a solid foundation for future work interacting with dynamic environments. The main contributions of our work are as follows:

1. We proposed a novel intrusion detection method based on an improved reinforcement learning algorithm. We design the dual-experience replay buffer as the data source for experience replay and adaptively adjust the weights of each classification class to adjust the proportion of samples from each class in the replay buffer, aiming to effectively address imbalanced datasets and achieve good detection performance.
2. We developed a reinforcement learning environment suitable for intrusion detection tasks. This environment presents a versatile interface for training and evaluating the performance of agents, providing a foundation for applying reinforcement learning to intrusion detection tasks in different scenarios.
3. We conducted experiments on three datasets to evaluate the proposed method. The results of the experiments demonstrate the proposed method has the capability to efficiently detect and identify network intrusion behaviors. Furthermore, when dealing with unbalanced data, the proposed method shows better classification performance in detecting minority attacks.

The remaining sections of this paper are organized as follows: Section 2 introduces related works. Section 3 presents a detailed description of the proposed method. Section 4 covers the experiments and results. This paper concludes with a summary in Section 5.

2. Related Works

In the field of cyber security, intrusion detection has always been an important task that has attracted much attention [21–23]. There have been many important studies on dealing with imbalanced data. In the research based on traditional machine learning, for example, by comparing random oversampling, random undersampling, and adaptive synthetic sampling methods on multiple datasets, Bagui et al. [24] found that oversampling and undersampling can significantly improve recall only when the data are extremely unbalanced. Among them, oversampling is more helpful in detecting attacks on minority classes, but its temporal complexity is higher. Cao et al. [25] proposed an intrusion detection model that integrates a convolutional neural network and gated recurrent unit. The model utilizes a hybrid sampling algorithm consisting of adaptive synthetic sampling (ADASYN) and repeated edited nearest neighbor (REN) to address the issue of imbalanced samples in intrusion categories. Shafieian et al. [26] conducted a comparison of the applications of bagging, boosting, and stacking techniques in the field of intrusion detection. They presented the advantages of multilevel stacking over other ensemble techniques in detecting low-footprint network intrusions. Thakkar et al. [27] proposed employing a deep neural network as the foundational estimator for a bagging classifier in order to address the issue of imbalanced intrusion detection data. The bagging method and class weights are utilized to harmonize the class distribution within the training set. Ren et al. [28] proposed a new ensemble method combining dynamic undersampling and boosting. The dynamic undersampling mechanism enhances the importance of boundary samples during the sampling process, providing a relatively balanced training set for each iteration, thus achieving better ensemble performance. Chui et al. [29] proposed a three-stage data generation algorithm that combines synthetic minority oversampling technique with generative adversarial networks (GANs) and variational autoencoder to produce high-quality data, aiming to address the issue of generating additional training samples for minority class in adversarial attacks. In addition to utilizing random oversampling, Dina et al. [30] employed a data synthesis

approach called a conditional generative adversarial network (CTGAN) to balance the data and improve the accuracy of various machine learning classifiers on the NSL-KDD and UNSW-NB15 intrusion detection datasets. Furthermore, Gaggero et al. [31] proposed an anomaly detection algorithm based on neural network autoencoders by analyzing the typical architecture of the storage system in microgrids and potential vulnerabilities therein. And observing the physical behavior of the system and combining it with autoencoders aims to replace human operator actions, enabling rapid the detection of possible dangerous working conditions and the implementation of countermeasures. Al-Abassi et al. [32] proposed a method called Ensemble Stacked AutoEncoder (ESAE) based on deep representation learning to construct a new balanced representation for the data imbalance problem in smart power systems. They combined this with a random forest classifier to detect attacks from the new representation. Fausto et al. [33] proposed an anomaly detection method based on machine learning. This method is different from those that only use information from the physical or cyber domains; it detects potential anomaly events by integrating data related to both the physical and cyber domains, making the method more universally valuable. These studies have provided new solutions and unique insights for intrusion detection tasks under unbalanced data.

In recent years, reinforcement learning as an adaptive learning framework has captured the attention of researchers due to its capacity for acquiring effective decision-making strategies through interaction with the environment, bringing new possibilities to the field of intrusion detection. Alavizadeh et al. [34] have combined Q-learning-based reinforcement learning with intrusion detection methods based on deep feedforward neural networks, proposing a reinforcement learning intrusion detection model based on DQL. By analyzing hyperparameters such as the discount factor and the number of learning cycles, they aim to identify the optimal fine-tuning strategy for the reinforcement learning agent. Benaddi et al. [35] introduced an IDS based on deep reinforcement learning that analyzes the interaction between normal IDS and attackers using non-zero-sum and random game simulations, enabling the game to achieve Nash equilibrium and converge to an optimal decision-making strategy. Mohamed et al. [36] proposed an intrusion detection model that combines the SARSA-based reinforcement learning algorithm with deep neural networks and verified its performance in achieving high-precision classification in unbalanced and modern traffic networks. Yang et al. [37] have proposed an intrusion detection method based on reinforcement learning that operates at both the packet level and flow level. Using image embedding techniques, the network session data are transformed into images that are processed by CNN and 1D-CNN for packet-level detection. As for flow-level detection, a conditional generative adversarial network (CGAN) and an ε-greedy strategy are employed to train the reinforcement learning agent. Dake et al. [38] have proposed a MADDPG framework for intrusion detection in an SDN environment that involves the utilization of two agents. This framework aims to efficiently optimize multipath routing and detect or prevent malicious DDoS traffic within an SDN. Both agents collaborate in the same environment to fulfill network optimization tasks. Sethi et al. [39] proposed a deep reinforcement learning intrusion detection system employing multiple distributed agents, incorporating the attention mechanism for attack detection and classification. Additionally, they integrated a denoising autoencoder (DAE) with the model to enhance its robustness against adversarial attacks. These works reflect the practical value and huge potential of reinforcement learning in intrusion detection scenarios.

3. Proposed Method

In this paper, we propose an intrusion detection method based on adaptive sample distribution dual-experience replay reinforcement learning. The method takes network traffic data and relevant network activity features as inputs, which are preprocessed and extracted for the reinforcement learning model. During the training phase, the model interacts with the reinforcement learning environment we developed for intrusion detection. The objective is to maximize cumulative rewards, enabling it to learn attack patterns in

network traffic characteristics and identify novel intrusions. Utilizing the attack patterns learned by the model, it classifies the input network traffic information, determining whether it corresponds to normal network behavior or potential intrusion activity.

3.1. Overall Structure

The overall structure of the proposed method is illustrated in Figure 1, which is divided into three components: data preparation, reinforcement learning environment, and detection model.

- Data preparation. We transform the raw data into a format that aligns with the input requirements of the model. Also, not all attributes and values in the dataset are conducive to model training; hence, it is necessary to extract meaningful data from the dataset. According to numerous studies and practical applications, it has been demonstrated that effective data preprocessing can have a substantial impact on the training and predictive performance of a model [40,41]. We undertake data preprocessing from the following three perspectives: (1) Data cleaning—eliminating redundant data, addressing missing values, and handling outliers. (2) Feature selection—involves analyzing and identifying effective features from the original dataset that exhibit high correlation with the target variable and can better contribute to model training and prediction. (3) Data transformation and encoding—values associated with different features exhibit distinct scales and distributions and may also encompass textual and categorical data types. As a result, it becomes necessary to perform transformations and encodings on diverse types of data.
- Reinforcement learning environment. The environment provides foundational conditions for applying reinforcement learning to intrusion detection, encompassing the necessary action space, state space, reward feedback, state transition model, task termination condition, etc. And the environment serves as the interface through which the agent interacts with the external world, providing the necessary information for the agent to perceive and act upon.
- Detection model. The detection model acts as an agent in the context of reinforcement learning, serving as an intrusion detection classifier. It receives the state of the environment, namely network traffic information, extracts network features, identifies potential intrusion behavior, and subsequently classifies the traffic. Utilizing the essence of reinforcement learning, the detection model engages in iterative experimentation and exploration of the environment, refining itself to obtain the utmost optimal strategy. In this paper, the detection model is based on our proposed adaptive sample distribution dual-experience replay reinforcement learning algorithm, which performs well with imbalanced samples.

3.2. Reinforcement Learning Environment for Intrusion Detection

To integrate reinforcement learning with intrusion detection, we developed a custom intrusion detection environment suitable for reinforcement learning based on OpenAI Gym [42]. Gym is an open-source toolkit widely used in the field of reinforcement learning. It provides a unified interface and standardized environment definition, allowing researchers to experiment and compare their proposed reinforcement learning algorithms on a standardized and unified platform. Specifically, the components included in our developed intrusion detection reinforcement learning environment are as follows:

- State space. In intrusion detection tasks, we consider network traffic features such as packet size, network protocols, and transmission information as the state, and all states form the state space. Due to the existence of discrete features such as network protocol types and port numbers, as well as continuous features such as packet size and transmission rate, the overall state space will be a hybrid state space. Therefore,

specific implementation requires the preprocessing of these features. The state space S is represented as follows:

$$S = \{(f_1, f_2, \ldots, f_n) | f_i \in \mathbb{R}\}, \tag{1}$$

Among these, n represents the quantity of network traffic features, with each f_i representing a distinct network traffic feature.

- Action space. The action space is responsible for labeling the incoming network traffic data for intrusion, i.e., categorizing them as either normal or attack. Typically, the output of a reinforcement learning algorithm is a probability distribution over actions. Therefore, in the context of intrusion detection, the model generates a vector of probabilities, with its shape matching the number of intrusion classes, corresponding to each class. The agent selects an action based on the generated classification probability vector. The action space A is represented as follows:

$$A = \{a_1, a_2, \ldots, a_m\}, \tag{2}$$

where m represents the number of intrusion categories (including the normal type), and each a_i represents an intrusion detection label.

- Reward function. The reward function represents the feedback on the intrusion detection results, which considers whether the attack was successfully detected or when there were false alarms. Therefore, if the agent's classification action is consistent with the actual results, a positive reward will be given; however, if the agent's action is inconsistent with the actual results, such as false positives or false negatives, a negative reward will be given. Thus, the intrusion detection agent will optimize the model according to the reward feedback, striving to obtain as much positive reward as possible to maximize cumulative rewards. In reinforcement learning, a cumulative reward is one of the metrics used to measure the effectiveness of learning. The reward function R is represented as follows:

$$R = \begin{cases} r_{t_{correct}}, & a_t = class_t \\ r_{t_{incorrect}}, & a_t \neq class_t \end{cases}, \tag{3}$$

where a_t is the classification action chosen by the agent at time step t, and $class_t$ is the true intrusion type of the detection sample input at time step t.

- State transition. The concept of state transition elucidates the pattern of changes in the environment following the actions taken by the agent. In the context of intrusion detection tasks, state transition is represented as the result of the current time step after classifying a certain traffic sample and then transitioning to the traffic sample of the next time step. The state transition model is represented as follows:

$$s_{t+1} = \begin{cases} next(s_t), & p \\ s_{t+1} \sim P(S), & 1-p \end{cases}, \tag{4}$$

s_t represents the state at the current time step, while s_{t+1} represents the new state transitioned to after executing the state transition to the next time step. The transition of states consists of two specific scenarios: one is the probability of transitioning to the next sample at the current time with p, and the other is the probability of randomly selecting a sample from the state set S with $1 - p$.

- Termination condition. The termination condition defines the circumstances under which the task should conclude. In our setting, the fixed time steps serve as the criterion for task completion, wherein the agent ceases to interact with the environment after handling a predetermined amount of traffic samples.

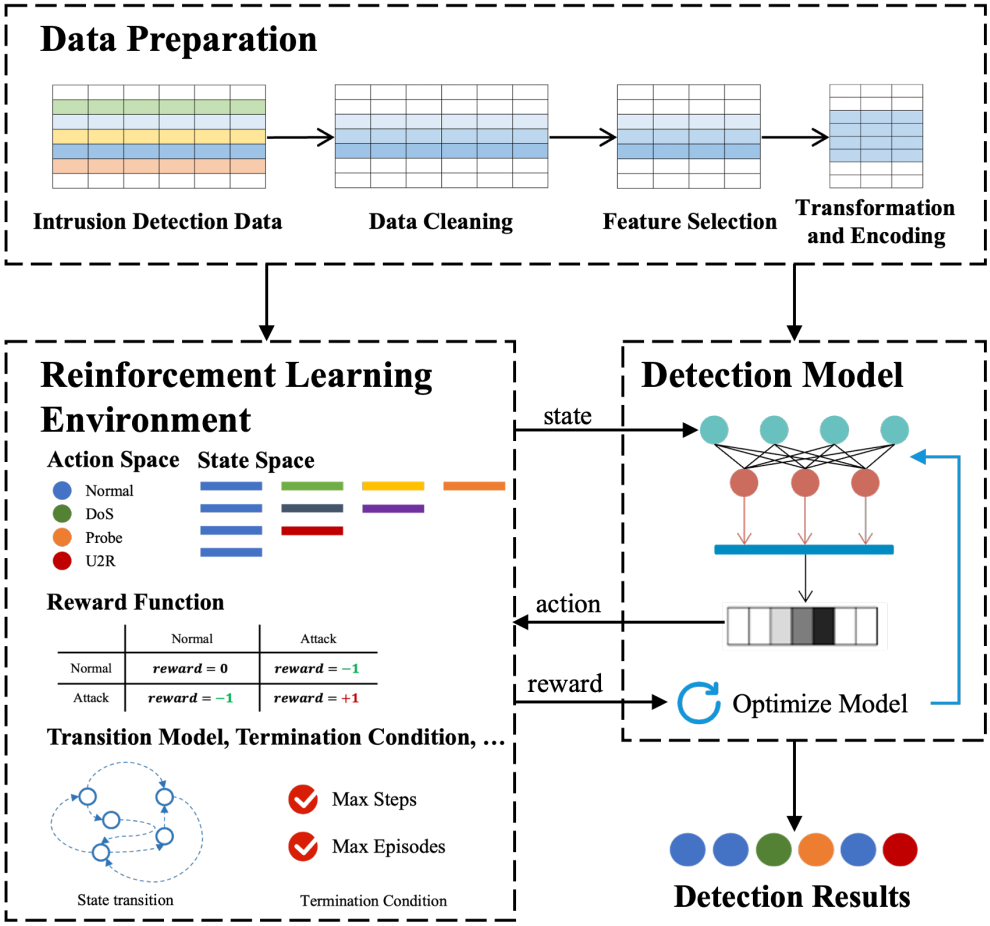


Figure 1. Overall structure of the proposed method. Data preparation is responsible for preprocessing network traffic data, while the reinforcement learning environment provides the necessary conditions for conducting intrusion detection tasks. The detection model, also referred to as the agent, interacts with the environment as a classifier and generates detection results.

In summary, the environment we developed presents a versatile interface for training and evaluating the performance of agents. The environment also supports customized reinforcement learning algorithms, customized reward function design, and configurable parameter options to facilitate model training and optimization.

The interaction process between the agent and the environment is depicted in Figure 2. During both the training and testing phases, the agent, acting as a reinforcement learning model for intrusion detection, interacts with the environment. During each time step, the agent receives the current state provided by the environment, which includes information such as current network traffic features. Based on the learned policy, the agent selects an action for classifying the current network traffic and applies it to the environment. The environment provides the agent with immediate rewards based on the action taken by the agent, while also returning updated information on network traffic status. The agent updates its knowledge and strategies based on environmental feedback rewards to better adapt to the environment. The process will continue over a duration of time, as the agent engages in continuous interactions with the environment, ultimately refining its strategy to attain more cumulative rewards. More cumulative rewards signify that the agent has acquired valuable knowledge from the characteristics of network traffic and is capable of accurately classifying network traffic information.

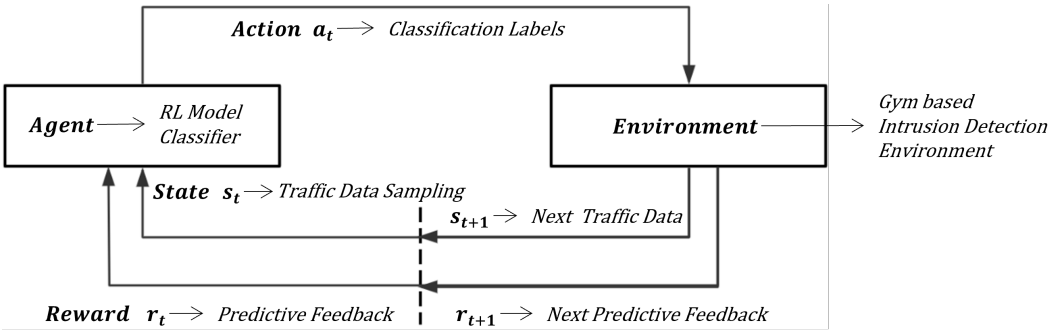


Figure 2. Interaction process between agent and environment. The environment provides the state space and action space, rewarding or penalizing the agent based on its actions. As the agent interacts with the environment, it continuously learns and adjusts its strategy to maximize long-term rewards.

3.3. Detection Model Based on Adaptive Sample Distribution Dual-Experience Replay RL

The adaptive sample distribution dual-experience replay reinforcement learning algorithm that we proposed is an improvement based on the Deep Q-Network (DQN) [43]. DQN is a reinforcement learning algorithm that combines the ideas of deep neural networks and Q-learning. It utilizes neural networks to approximate the Q-value function, enabling learning and decision making in complex environments, and the Q-value function is utilized to evaluate the value of taking a specific action in a given state. The realization of DQN involves two crucial techniques, namely experience replay and target networks. The DQN stores the agent's past experiences, which are the quadruples of state, action, reward, and next state obtained by the agent during each interaction with the environment, in an experience replay buffer. It then randomly samples from these experiences during the training process for learning. The purpose of doing so is to disrupt the temporal correlation between samples and enhance the efficiency of sample utilization. The target network is utilized to compute the target Q-value, contributing to the stability of the training process. This is a network with a structure that is the same as the primary network but with independent parameters, which are regularly updated based on the parameters in the primary network.

However, in intrusion detection scenarios, the highly imbalanced nature of network traffic data often results in models being biased towards the majority class samples. While using resampling techniques may alleviate the problem of imbalanced sample distribution, it could potentially result in the loss of crucial information. Therefore, we require an approach that can address imbalanced sample distribution and accurately reflect the true distribution of data samples.

We propose a reinforcement learning algorithm based on adaptive sample distribution dual-experience replay, wherein a second experience replay buffer is added to DQN. The experiences in the second buffer are distributed based on the weights of samples from each class, determining their proportion in the experience buffer. As depicted in Algorithm 1, in addition to initializing the primary experience replay buffer, we also initialized a secondary experience buffer and allocated initial weights for each class, while setting up a switch window for experience replay buffers. During the experience storage phase, we store the experience quadruples in both experience buffers simultaneously. However, unlike the primary buffer which collects all experiences, the secondary buffer decides whether to add the current experience to the buffer based on the weight of each class. During the experience replay phase, the algorithm will determine which experience buffer to use based on the experience replay buffer switch window. After completing a batch of neural network updates, the algorithm will examine the current batch of updates in comparison to the previous one to determine whether the loss has increased or decreased. We refer to this as the loss trend. We hypothesize that if the loss trend has increased, it indicates the presence of potentially difficult-to-classify minority class samples. As a result, we proceed with weight updates to adjust the distribution of samples in the secondary experience buffer.

Algorithm 1: Dual-Experience Replay DQN

Initialize primary replay buffer D_1 and second replay buffer D_2 to capacity N, M, respectively
Initialize sample distribution weights $W = \left\{w_1, w_2, \ldots, w_{len(labels)}\right\}$ and replay buffer switch window H
Initialize action-value function Q with random weights θ
Initialize target action-value function $\hat{Q}$ with random weights $\theta^{'} = \theta$
for *episode* = 1 **do**
  Initialize sequence $s_1 = x_1$ and preprocessed sequence φ_1=$\varphi(s_1)$
  for $t = 1, T$ **do**
    With probability ε, select a random action a_t otherwise select $a_t = argmax_a Q(\varphi(s_t), a; \theta)$
    Execute action a_t in the emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t$, a_t , x_{t+1} and preprocess $\varphi_{t+1} = \varphi(s_{t+1})$
    Store transition $(\varphi_t, a_t, r_t, \varphi_{t+1})$ in D_1
    Store transition $(\varphi_t, a_t, r_t, \varphi_{t+1})$ in D_2 according to weights W
    Sample random minibatch of transition $(\varphi_j, a_j, r_j, \varphi_{j+1})$ from D
    Set

$$y_i = \begin{cases} r_j, & \text{if episode terminate at step } j+1 \\ r_j + \gamma max_{a'} \hat{Q}(\varphi_{j+1}, a^{'}; \theta^{-}), & \text{otherwise} \end{cases}$$

    Perform a gradient descent step on $(y_j - Q(\varphi_j, a_j; \theta))^2$ with respect to the network parameters θ
    Update sample distribution weights W if loss trend increases
    Every C step reset to $\hat{Q} = Q$
  end
end

The weights of each class in the secondary experience buffer are subject to an adaptive and dynamically updating process, which we refer to as the adaptive sample distribution. Its update is illustrated in Algorithm 2. We initiated by computing a classify metric that combines the F1-Score of each class with the loss trend, aiming to assess the relative significance of samples in each class and thereby capture the impact level of each class's samples on the overall classification performance. The calculation of the classify metrics is shown as (5), where $L_{current}$ represents the loss from the current training batch, L_{last} represents the loss from the previous training batch, and the difference between the two indicates the loss trend. And f_i represents the F1-Score of class i; ϵ is a factor ensuring that the denominator is not zero, with a value of 1×10^{-7} .

$$m_i^c = \frac{(L_{current} - L_{last})^2}{f_i + \epsilon}, \tag{5}$$

In order to comprehensively consider the classification performance and the distribution of samples in different classes, thus finely adjusting the weights of each class, we propose an analysis metric. The formula for calculating the analysis metrics is shown as (6), where s_i represents the number of samples belonging to the class i, S is the total number of samples in each batch, and N represents the number of classes. This metric integrates the classify metric from the previous step and the proportions of samples in each class among all samples.

$$m_i^a = \frac{m_i^c}{\sqrt{\frac{s_i}{S}} + \epsilon}, S = \sum_{i=1}^{N} s_i, \tag{6}$$

Algorithm 2: Adaptive Sample Distribution

Initialize historical factor y and class number N
Get training loss $L_{current}$ and L_{last}
for $i = 1, N$ **do**
    Calculate F1-Score f_i of class i
    Calculate the classify metric m_i^c of class i with F1-Score f_i and loss trend $L_{current} - L_{last}$
    Get total sample number S of current batch and count the sample number s_i of class i
    Calculate analysis metric m_i^a of class i with metric m_i^c and the sample proportions of class i
    Calculate historical weights sequences w_i^h combing the historical factor y
    Calculate weight w_i of class i according to historical weights w_i^h, analysis metric m_i^a, and historical factor y
end
Normalize the weights $W = \{w_1, w_2, \ldots, w_N\}$ as W'
Update weights $W = W'$

For each class, we computed its classify and analysis metrics, incorporated them as part of the weight's calculation, and introduced historical factors that adjust the impact of the current analysis metric and the previous weights on the new weight to balance the weight between current metrics and historical weights. Finally, we normalized the weights for all classes. The updated formula for the weight w_i in class i is shown below:

$$w_i = (1 - \lambda) \cdot m_i^a + \lambda \cdot w_i^h, \tag{7}$$

$$w_i^h = w_i^{k-1} + \lambda w_i^{k-2} + \cdots + \lambda^{k-1} w_i^1 \tag{8}$$

$$\sum_{i=1}^{N} w_i = 1, w \in W, W = \{w_1, w_2, \ldots, w_N\}, \tag{9}$$

where λ denotes the historical factor, k denotes the number of weight iterations, and w_i^h represents the historical weights sequence of the class i. When considering historical weights sequences, we weight the previous $k-1$ historical weights by gradually decaying them according to a weight factor and then compute the weights' sum. This process preserves the influence of historical weights on the new weight, but as the historical factor decays, the impact of earlier historical weights on the new weight gradually diminishes. This balancing mechanism allows the weight to retain the influence of historical information while also focusing more on the most recent situation.

By incorporating adaptive sample distribution dual-experience replay, the overall architecture of our algorithm is depicted in Figure 3. We input the network traffic features as the state into the model and approximate the state-action value function (i.e., Q-function) through a deep neural network. During the training process, the agent engages in interactions with the environment, classifying traffic samples and storing the experience in two separate experience replay buffers. Subsequently, the agent selects a buffer based on the switch window within the experience buffers and performs experience sampling. The target Q-value is computed using a target network, which represents the expected long-term cumulative reward for predicting a certain classification label given the network traffic feature information. Then, the loss function is optimized to update and upgrade the network parameters, continuously learning and updating the Q-value to approximate the optimal classification action strategy. In addition, predictive results and the training status of the neural network are used to generate corresponding classification and analysis metrics. Combined with the historical weights sequence, the experience distribution in the experience buffer is adaptively adjusted to achieve better intrusion detection classification effects.

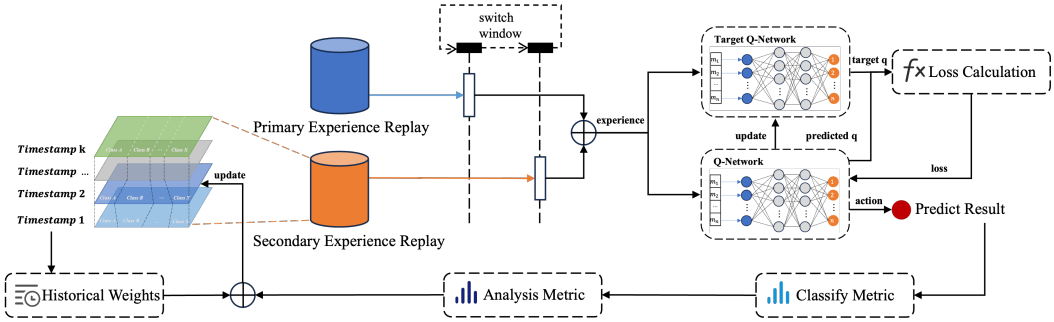


Figure 3. Adaptive sample distribution dual-experience replay RL.

We assume there are n traffic data samples and m discrete actions (representing different classification labels); s represents the feature information of the current network traffic, a represents the classification label predicted by the agent, r is the feedback on the correctness of the agent's prediction, γ is the discount factor, and $s^{'}$ represents the feature information of the next network traffic after the agent takes action a. The Q-value update formula can be summarized as the following equation:

$$Q(s,a) = (1-\alpha)\cdot Q(s,a) + \alpha\cdot(r + \gamma\cdot max_{a'}Q(s^{'},a^{'})), \tag{10}$$

Among which $s, s^{'} \in S, a, a^{'} \in A$. Additionally, we also employ the ε-greedy strategy to strike a balance between exploration and exploitation. When making decisions, the agent will choose a random action with a probability of ε, while selecting the currently known optimal action with a probability of $1-\varepsilon$. This approach serves to balance the likelihood of exploring unknown actions and exploiting the benefits of known optimal actions, aiding the agent in optimizing its strategy.

The proposed algorithm of adaptive sample distribution dual-experience replay reinforcement learning aims to address the issue of sample imbalance in intrusion detection. By introducing a second experience buffer, the algorithm focuses on the fewer and more challenging minority class samples, effectively preventing the minority class samples from being overshadowed by the majority class samples. We also implemented an adaptive dynamic updating mechanism, enabling the flexible adjustment of the distribution of samples in the experience buffer for each class. This allows the model to focus more on samples that are deemed challenging in the current stage, thereby enhancing the learning effectiveness for minority categories. Furthermore, the inclusion of a window for switching between experience buffers allows the model to seamlessly transition between the two, preventing an excessive focus on one experience buffer and the subsequent neglect of others. This ensures a comprehensive learning approach across all classes. This innovative design enhances the model's versatility and robustness, enabling it to adapt to evolving intrusion detection scenarios and more accurately reflect the true distribution of a sample.

4. Experiments and Results

4.1. Datasets

In order to validate the usability and effectiveness of the proposed method in intrusion detection research, we conducted experiments on three different intrusion detection datasets: NSL-KDD, AWID, and CICIoT2023. We chose these datasets from three perspectives: (1) they all have a considerable amount of data and cover a large amount of network information, thus providing a better simulation of real-world network scenarios, which is crucial for evaluating intrusion detection models; (2) they cover a comprehensive range of attack types, allowing for a full assessment and improvement of the accuracy and robustness of intrusion detection models for various types of attacks; (3) the datasets that have already been divided into training sets and test sets will be given priority consideration, which allows researchers to train and optimize models on the same training set and

evaluate the performance of models on the same testing set, thus making the comparison results between different models more objective and fair.

The NSL-KDD dataset [44] is one of the commonly used datasets for researching network intrusion detection. It is an improved version of the KDD'99 dataset [45], addressing issues such as redundancy in the training set and repetition in the test set. The NSL-KDD dataset consists of KDDTrain+, KDDTest+, KDDTrain+_20Percent, and KDDTest-21, with KDDTrain+_20Percent and KDDTest-21 being subsets of KDDTrain+ and KDDTest+, respectively. In this study, we use the KDDTrain+ and KDDTest+ datasets with 125973 and 22544 samples, respectively. Featuring 43 attributes, of which 39 are numeric and 3 are categorical, the dataset also includes a label indicating "normal" or "attack". The attack class in the dataset is divided into four major types: DoS (Denial of Service), Probe, R2L (Remote to Local), and U2R (User to Root). The sample distributions in the training and test datasets are shown in Table 1 and Table 2, respectively.

Table 1. The distribution of samples in the KDDTrain+ dataset.

Attack Type	Number of Samples	Proportion
Normal	67,343	53.46%
DoS	45,927	36.46%
Probe	11,656	9.25%
R2L	995	0.79%
U2R	52	0.04%

Table 2. The distribution of samples in the KDDTest+ dataset.

Attack Type	Number of Samples	Proportion
Normal	9711	43.08%
DoS	7458	33.08%
Probe	2421	12.22%
R2L	2754	10.74%
U2R	200	0.89%

AWID is the intrusion detection dataset based on 802.11 wireless networks released by the University of the Aegean [46], which consists of data collected from real-world wireless network environments. The dataset is available in two versions: the full dataset and a smaller dataset. It is worth noting that the smaller version is not a subset of the larger dataset but is rather created from different time periods, devices, and environments. Both versions are divided into training and testing sets. For this experiment, we utilize the smaller version of the dataset, with AWID-CLS-R-Trn serving as the training set and AWID-CLS-R-Tst as the testing set. Despite being the smaller version, the AWID dataset is still substantial, with 1,795,575 and 575,643 records in the training and testing sets, respectively. Each record in the dataset consists of 155 features, with attack labels categorized into three types: flooding, impersonation, and injection. The sample distribution in the training and testing sets can be found in Table 3 and Table 4, respectively.

Table 3. The distribution of samples in the AWID-CLS-R-Trn dataset.

Attack Type	Number of Samples	Proportion
Normal	1,633,190	90.96%
Injection	65,379	3.64%
Impersonation	48,522	2.70%
Flooding	48,484	2.70%

Table 4. The distribution of samples in the AWID-CLS-R-Tst dataset.

Attack Type	Number of Samples	Proportion
Normal	530,785	92.21%
Injection	20,079	3.49%
Impersonation	16,682	2.90%
Flooding	8097	1.41%

The CICIoT2023 dataset [47] constitutes a compendium of Internet of Things (IoT) attack data, spearheaded by the University of New Brunswick. The topology, comprising 105 authentic IoT devices, delineates a record of 33 categories of attack data emanating from IoT devices. These assaults are categorizable into seven distinct types, namely: DDoS, Brute Force, Spoofing, DoS, Recon, Web-based, and Mirai. The dataset is exceedingly vast, comprising a total of 46,686,579 instances across 46 features. The distribution of its samples is delineated in Table 5.

Table 5. The distribution of samples in the CICIoT2023 dataset.

Attack Type	Number of Samples	Proportion
Normal	1,098,195	2.35%
DDoS	33,984,560	72.79%
Brute Force	13,064	0.03%
Spoofing	486,504	1.04%
DoS	8,090,738	17.33%
Recon	354,565	0.76%
Web-based	24,829	0.05%
Mirai	2,634,124	5.64%

4.2. Performance Metrics

The experiment in this paper is conducted using the mentioned two intrusion detection datasets for a multiclassification task. These datasets consist of various classes of attacks, and our objective is to effectively distinguish between normal and attack instances, while also identifying the corresponding attack categories. In the context of multiclassification tasks, *TP* (true positive) indicates the number of correctly classified positive samples by the classifier, *FP* (false positive) represents the number of samples incorrectly predicted as positive, *TN* (true negative) represents the number of samples correctly predicted as negative, and *FN* (false negative) signifies the number of samples incorrectly predicted as negative. Our experiment utilizes *TP*, *FP*, *TN*, and *FN* to compute the more intuitive performance metrics of Accuracy, Precision, Recall, and F1-Score.

Accuracy is the most frequently employed classification metric, signifying the proportion of correctly classified samples by the classifier to the total number of samples. Its computation is determined by the following formula:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{11}$$

Precision is a measurement of the proportion of true positive samples among the samples predicted as positive by the classifier. It is calculated as follows:

$$Precision = \frac{TP}{TP + FP} \tag{12}$$

Recall represents the proportion of true positives that are correctly predicted as positive by the classifier, and it is calculated as follows:

$$Recall = \frac{TP}{TP + FN} \tag{13}$$

F1-Score represents the harmonic mean of Precision and Recall, utilized for a comprehensive evaluation of the classifier's performance. It indicates the classifier's ability to make good predictions for both positive and negative cases. The calculation for F1-Score is as follows:

$$F1{-}Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{14}$$

4.3. Implementation

4.3.1. Dataset Preparation

We preprocess the data based on three aspects: data cleaning, feature selection, and data transformation and encoding.

- Data cleaning. Both datasets were cleaned by removing columns where all values were either the same or empty. For the AWID dataset, we replaced the numerous occurrences of "?" with NaN, rather than directly substituting with 0, as it signifies a missing value rather than a specific numerical value. This approach avoids potential interference with subsequent data analysis and processing.
- Feature selection. For the NSL-KDD dataset, we removed the "difficulty" feature, which represents the classification difficulty of samples, as it does not provide any valuable assistance to our model. The final set of features used is shown in Table 6. For the AWID dataset, due to a large number of missing values, we calculated the percentage of missing values for each column in the dataset and removed columns with missing values exceeding 60%. Additionally, we eliminated features with constant values. The final set of features used is shown in Table 7. For the CICIoT2023 dataset, we curated pertinent features through the mutual information. The final set of features used is shown in Table 8.
- Data transformation and encoding. Both datasets were normalized or standardized numerical features and encoded categorical features to transform the data into input acceptable for the model.

In addition, the scale of the dataset impacts the training time and computational resources required for the model. For the AWID dataset, we randomly sampled 10% of the normal class samples from the training set and combined them with the other three types of attacks to form a new training set. Similarly, due to the immense size of the CICIoT2023 dataset and constraints related to computational resources and time, we opted for its reduced version, consisting of 933,730 samples. The reduced dataset still adheres to the distribution of the original data and is divided into training and testing sets in an 80%:20% ratio.

Table 6. NSL-KDD Features.

#	Feature	Type	#	Feature	Type
0	duration	int64	21	is_guest_login	int64
1	protocol_type	object	22	count	int64
2	service	object	23	srv_count	int64
3	flag	object	24	serror_rate	float64
4	src_bytes	int64	25	srv_serror_rate	float64
5	dst_bytes	int64	26	rerror_rate	float64
6	land_f	int64	27	srv_rerror_rate	float64
7	wrong_fragment	int64	28	same_srv_rate	float64
8	urgent	int64	29	diff_srv_rate	float64
9	hot	int64	30	srv_diff_host_rate	float64
10	num_failed_logins	int64	31	dst_host_count	int64
11	logged_in	int64	32	dst_host_srv_count	int64
12	num_compromised	int64	33	dst_host_same_srv_rate	float64
13	root_shell	int64	34	dst_host_diff_srv_rate	float64

Table 6. *Cont.*

#	Feature	Type	#	Feature	Type
14	su_attempted	int64	35	dst_host_same_src_port_rate	float64
15	num_root	int64	36	dst_host_srv_diff_host_rate	float64
16	num_file_creations	int64	37	dst_host_serror_rate	float64
17	num_shells	int64	38	dst_host_srv_serror_rate	float64
18	num_access_files	int64	39	dst_host_rerror_rate	float64
19	num_outbound_cmds	int64	40	dst_host_srv_rerror_rate	float64
20	is_host_login	int64	41	labels	object

Table 7. AWID Features.

#	Feature	Type
0	frame.time_epoch	float64
1	frame.time_delta	object
2	frame.time_relative	object
3	frame.len	object
4	radiotap.mactime	int64
5	radiotap.datarate	int64
6	radiotap.channel.freq	int64
7	radiotap.channel.type.cck	float64
8	radiotap.channel.type.ofdm	float64
9	radiotap.dbm_antsignal	int64
10	wlan.fc.type_subtype	int64
11	wlan.fc.type	float64
12	wlan.fc.subtype	float64
13	wlan.fc.ds	int64
14	wlan.fc.frag	int64
15	wlan.fc.retry	float64
16	wlan.fc.pwrmgt	float64
17	wlan.fc.moredata	float64
18	wlan.fc.protected	float64
19	wlan.qos.priority	int64
20	wlan.qos.bit4	float64
21	class	float64

Table 8. CICIoT2023 Features.

#	Feature	Type
0	IAT	float64
1	Tot size	float64
2	Magnitue	float64
3	Max	float64
4	AVG	float64
5	Tot sum	float64
6	Min	float64
7	Header_Length	float64
8	Number	float64
9	Weight	float64
10	flow_duration	float64
11	Duration	float64
12	Std	float64
13	Radius	float64
14	Covariance	float64

Table 8. *Cont.*

#	Feature	Type
15	rst_count	float64
16	Variance	float64
17	urg_count	float64
18	Srate	float64
19	Rate	float64
20	TCP	float64
21	label	int64

4.3.2. RL Setting

Due to the inconsistent features and attack categories present in different datasets, it is necessary to configure different reinforcement learning parameters for each dataset. These parameters encompass both the reinforcement learning environment and the RL agent. The reinforcement learning environment parameters include episode number, max step, and reward. In this study, the specific reinforcement learning environment parameters utilized are displayed in Table 9. The maximum number of steps per episode during the training phase is set to 100. This signifies that within each episode, the agent will handle 100 samples from the dataset and classify them accordingly. In order to evaluate the capability of the designed reinforcement learning agent to effectively handle intrusion detection tasks, particularly in the case of limited training data, we set the episode number during training to 500. This means that the agent will receive a total of 50,000 samples for training throughout the entire training phase. As for testing, multiple iterations of episodes are not required. Therefore, the episode number for testing is set to 1, which means that it only needs to traverse through the testing dataset once, with the maximum number of steps dependent on the size of the testing dataset. Regarding the reward computation, since we are performing a classification task for intrusion detection, there is no delayed reward. We set the reward calculation to ±1, where a correct prediction is rewarded with +1 and an incorrect prediction is rewarded with -1. Although this may seem straightforward, its effectiveness is proven in the experiments.

Table 9. Reinforcement Learning Environment Parameters.

Parameter	Values
max step	100 steps (training phase) length of test dataset (testing phase)
episode number	500 (training phase) 1 (testing phase)
reward	+1 (correct predict) -1 (incorrect predict)

Our model is an improvement of the DQN; thus, the parameters of the RL agent include the discount factor gamma, the exploration rate epsilon of the epsilon-greedy policy, the minimum value of the exploration rate epsilon min, the decay factor of the exploration rate epsilon decay, the batch size used for training the network, and the size of the experience replay buffer (including the second one proposed in this paper). Additionally, the network architecture varies for different datasets, including the number of hidden layers, the number of neurons in each hidden layer, the optimizer used for training the network, and its learning rate. The specific values of these RL agent parameters are shown in Table 10.

Table 10. RL Agent Parameters.

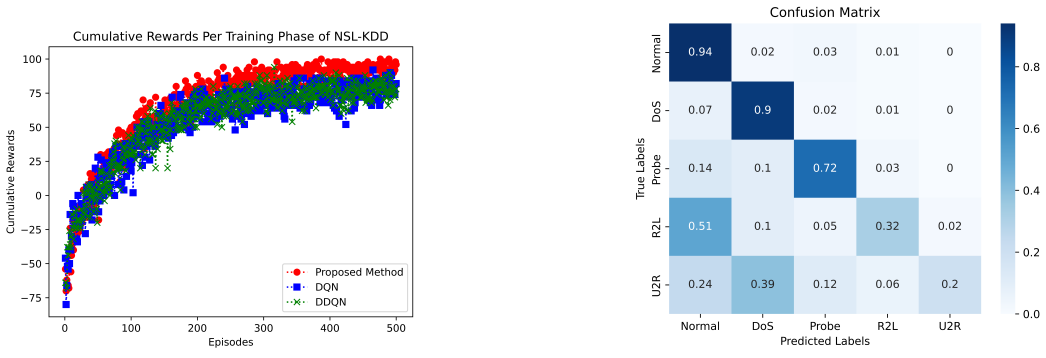
Parameter	Values
gamma	0.001
epsilon	0.8
epsilon min	0.01
epsilon decay	0.99
batch size	256
primary replay buffer size	1000
secondary replay buffer size	500
hidden layers	2
hidden units	2×100 (NSL-KDD) 2×50 (AWID and CICIoT2023)
optimizer	Adam (NSL-KDD and AWID) Adagrad (CICIoT2023)
learning rate	0.001 (NSL-KDD and AWID) 0.1 (CICIoT2023)

4.4. Results

Based on the data preprocessing and environment setup, we conducted experiments on three datasets, NSL-KDD, AWID, and CICIoT2023, to evaluate the proposed model. Furthermore, the experimental results were compared with other models, including Support Vector Machine (SVM), random forest, AdaBoost, and various other machine learning (ML) models, as well as the Deep Q-Network (DQN), Double Deep Q-Network (DDQN), and other reinforcement learning (RL) models.

4.4.1. NSL-KDD Result

The cumulative rewards for each episode during training using our proposed models on the KDDTrain+ dataset are shown in Figure 4a. It is evident that with each iteration, our reinforcement learning agent effectively learns the attack patterns present in the data samples. From an initial accumulation of negative rewards to a trajectory towards 100 (where positive rewards are obtained almost at every step), the agent gradually transitions from selecting actions through random exploration to selecting optimal actions (i.e., correct classification) based on the observation (i.e., sample's features) it receives. Furthermore, from the graph, we can also observe the cumulative reward curves of two other reinforcement learning models, DQN and DDQN. Although they also learned some attack patterns from the samples, it is evident that their final cumulative rewards did not reach the level of our proposed model.



(**a**) (**b**)

Figure 4. (**a**) Cumulative rewards per training phase (NSL-KDD); (**b**) confusion matrix of test set results (NSL-KDD).

Figure 4b showcases the confusion matrix of the proposed approach on the KDDTest+ dataset, while Table 11 displays the classification precision of different models in predicting each class on the same dataset. Given that there are 38 types of attacks in the KDDTest+ dataset, compared with the 23 types in the KDDTrain+ dataset, it presents a significant

challenge for intrusion detection models. There are 21 attack types shared between the training and testing sets, with 2 unique attack types present in the training set and a staggering 17 attack types in the testing set that were not encountered during training. Consequently, accurately assigning an unseen attack pattern to a specific class proves to be a formidable task for classification models. However, our model tackles this challenge by leveraging an additional experience replay buffer, enabling extra learning on attack class with limited samples and higher difficulty. As a result, unlike other approaches that heavily lean towards the majority class, our model demonstrates a more balanced classification performance.

Table 11. Classification precision of different models on each class of KDDTest+ dataset.

Model	Normal	DoS	Probe	R2L	U2R
SVM	0.9282	0.7485	0.6171	0.000	0.0000
Random Forest	0.6300	0.9510	0.7360	0.7320	0.0000
KNN	0.9278	0.8225	0.5940	0.0356	0.0350
DQN	0.7860	0.9012	0.5011	0.8826	0.0777
DDQN	0.7838	0.9278	0.6948	0.7792	0.0391
AE-RL [19]	0.8603	0.8183	0.4976	0.7930	0.0914
AEDNN [48]	0.9338	0.8121	0.7138	0.2763	0.1584
Proposed Model	0.8278	0.8937	0.7396	0.8016	0.3744

The overall performance and classification metrics of the proposed model and other detection models on the KDDTest+ dataset are shown in Figure 5. It can be observed that our model achieves the best performance in almost all metrics. Even in an extremely imbalanced dataset, our model can still achieve a classification accuracy of 82.03%.

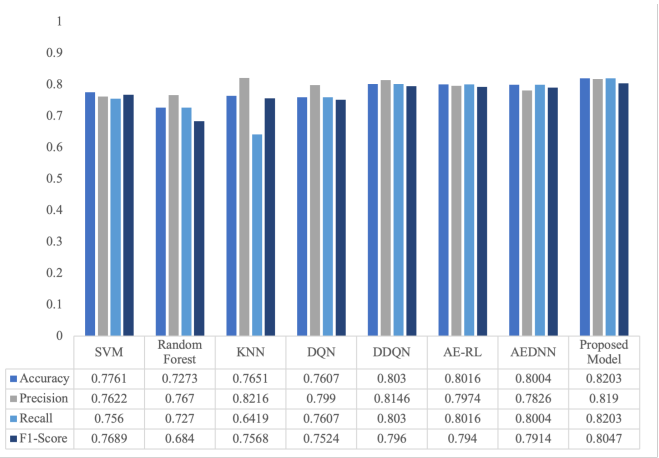
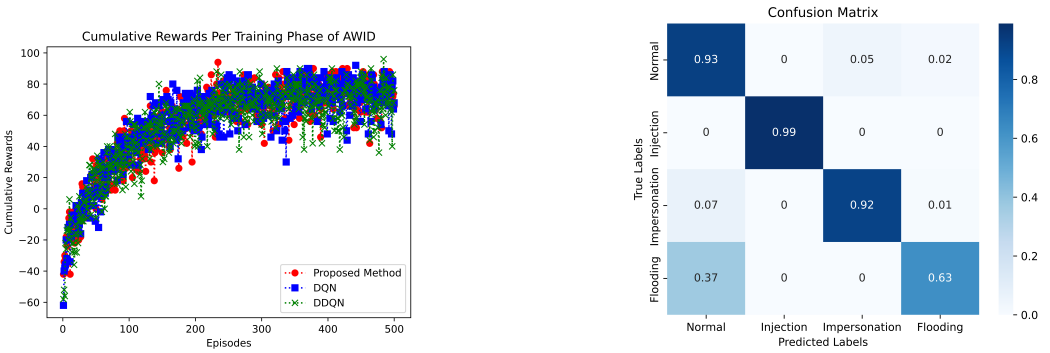


Figure 5. Comparison of detection results with other models (NSL-KDD).

4.4.2. AWID Result

The plot in Figure 6a depicts the cumulative reward curve of the proposed model and two reinforcement learning models, DQN and DDQN, on the training set AWID-CLS-R-Trn. During the training phase, the differences among these three reinforcement learning models are relatively small, and they can effectively learn different attack types. However, due to the significant imbalance between normal and attack samples in the AWID dataset, with a ratio as high as 10:1, the cumulative reward curves of the three models appear to be good, mainly because the input majority consists of normal samples. Table 12 compares the precision of different models on different types of samples in the AWID-CLS-R-Tst test set. As mentioned above, all models perform exceptionally well in classifying normal samples. However, there is a significant difference between the models' performance on attack samples, especially for impersonation and flooding attacks. Due to their low proportion in the dataset and the complexity of their attack patterns, most models struggle to classify these samples accurately. Combining this with the confusion matrix depicted

in Figure 6b, it is evident that our proposed model demonstrates better performance in detecting minority classes.



(**a**) (**b**)

Figure 6. (**a**) Cumulative rewards per training phase (AWID); (**b**) confusion matrix of test set results (AWID).

Table 12. Classification precision of different models on each class of AWID-CLS-R-Tst dataset.

Model	Normal	Injection	Impersonation	Flooding
Naive Bayes	0.9825	0.7982	0.3950	0.0260
AdaBoost	0.9950	0.9875	0.3640	0.1700
QDA	1.000	0.9980	0.5400	0.1200
DQN	0.9499	0.9042	0.0365	0.0574
DDQN	0.9823	0.2776	0.8000	0.2642
Proposed Model	0.9912	0.9267	0.5402	0.3463

Figure 7 presents the overall performance of the proposed model compared with different models such as Policy Gradient [20], Actor Critic [20], and IG-CH [49], etc., on the AWID-CLS-R-Tst test set. Our proposed model demonstrates balanced performance across all metrics while outperforming other models in terms of precision and F1-Score. This indicates that our model is capable of accurately identifying intrusions in intrusion detection, which is crucial as mislabeling normal activities as intrusions can lead to unnecessary alarms and interference. Additionally, our model maintains high recall while also achieving high precision, implying that it can capture more true intrusion events and minimize false negatives. As a result, our approach effectively balances both precision and recall.

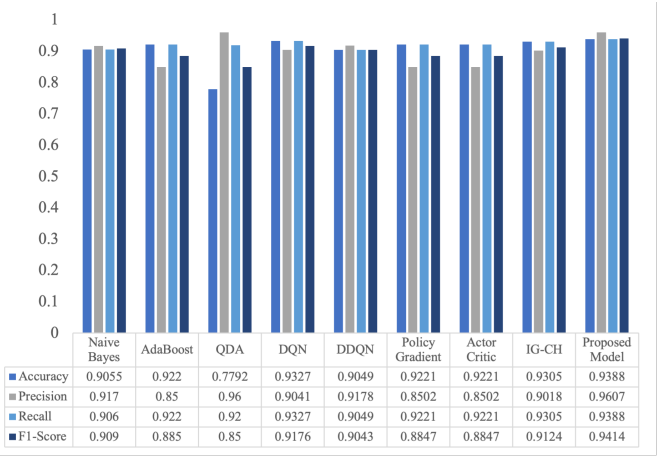
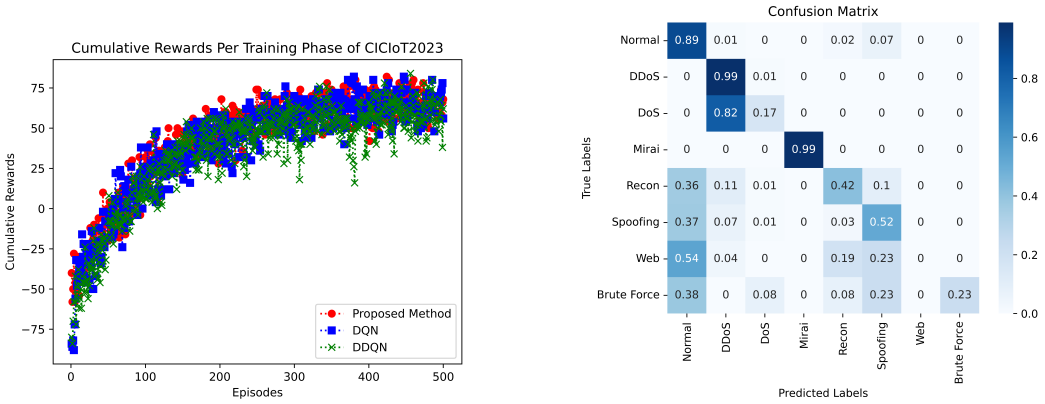


Figure 7. Comparison of detection results with other models (AWID).

4.4.3. CICIoT2023 Result

Figure 8a and Figure 8b, respectively, show the comparison of cumulative rewards for our proposed model and two other reinforcement learning models, as well as the confusion

matrices evaluated on the test set. Unlike the two datasets mentioned earlier, none of the three reinforcement learning models achieved a cumulative reward of 100. This is due to the vast size of the CICIoT2023 dataset and the significant disparity between the majority and minority classes. After we used a downscaled version of the dataset, the minority samples available for learning became even fewer. Despite not perfectly learning all attacks, our proposed method can still achieve a cumulative reward of around 75, which is slightly higher than both DQN and DDQN.



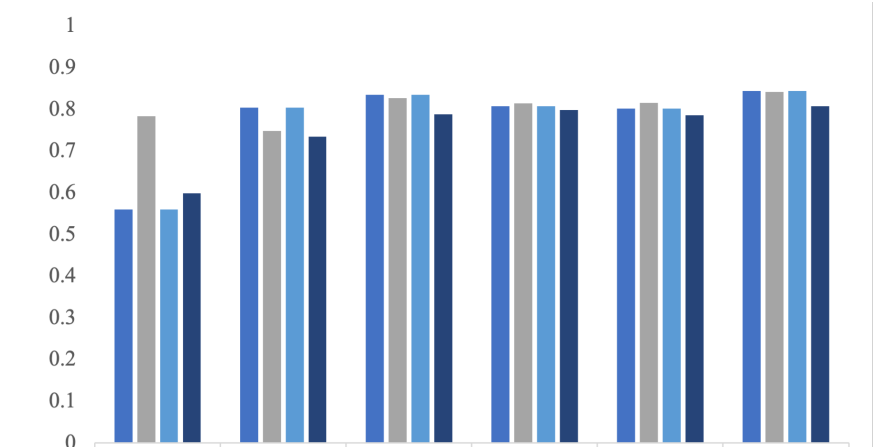
(**a**) (**b**)

Figure 8. (**a**) Cumulative rewards per training phase (CICIoT2023); (**b**) confusion matrix of test set results (CICIoT2023).

The precision of different models for classifying different classes of attacks on the test set is shown in Table 13. We can observe that for categories such as DDoS and Mirai, most models have good precision. However, the classification performance for Web-based and Brute Force attacks is noticeably lower. As previously mentioned, there are very few minority class samples available for learning. However, even in such extreme circumstances, our proposed model maintains a more balanced precision for most classes compared with other models. For Web-based attacks, our method did not effectively detect them. We speculate that this is due to the reduced version of the training set having too few samples in this class, causing the secondary experience buffer to be unable to collect enough samples for the agent. Additionally, a comparison of the overall performance of different models shown in Figure 9 reveals that our proposed method still maintains the best performance on different metrics. Our method, while maintaining leading accuracy, also maintains better F1-Score. This suggests that our model is more adept at dealing with the imbalance between different attack classes, while also maintaining higher accuracy.

Table 13. Classification precision of different models on each class of CICIoT2023 dataset.

Model	Normal	DDoS	DoS	Mirai	Recon	Spoofing	Web	Brute Force
DT	0.6222	0.9072	0.2679	0.9831	0.3045	0.3939	0.0729	0.0800
LR	0.6310	0.8052	0.4755	0.9848	0.4167	0.5312	0.0000	0.3705
GBM	0.6313	0.8203	0.9575	1.0000	0.3616	0.2032	0.0659	0.2308
DQN	0.3609	0.8593	0.6892	0.9502	0.3733	0.3353	0.0116	0.0000
DDQN	0.6172	0.8525	0.7356	0.8973	0.0885	0.1613	0.0079	0.0033
Proposed Model	0.6945	0.8321	0.8385	0.9985	0.7376	0.6921	0.0000	0.7500



	Decision Tree	Logistic Regression	GBM	DQN	DDQN	Proposed Model
Accuracy	0.5606	0.8044	0.8346	0.808	0.8025	0.8439
Precision	0.7832	0.7478	0.8274	0.814	0.8156	0.8417
Recall	0.5606	0.8044	0.8346	0.808	0.8025	0.8439
F1-Score	0.5993	0.7351	0.7877	0.7983	0.7863	0.8078

Figure 9. Comparison of detection results with other models (CICIoT2023).

4.4.4. Result Summary

We conducted experiments on the intrusion detection datasets, NSL-KDD, AWID, and CICIoT2023, widely used in the field of network security and covering modern intrusion attempts. Both datasets suffer from the issue of imbalance sample distribution; as in real-world networks, the number of normal traffic samples far exceeds that of intrusion samples. Therefore, our experimental goal is not only to pursue higher overall detection accuracy but also to emphasize the detection performance for each class, particularly for the minority classes. We employed a series of evaluation metrics, including accuracy, precision, recall, and F1-Score, to comprehensively assess the performance of our model when confronted with imbalanced data distribution in intrusion detection tasks. Through experimentation and analysis, our model demonstrates excellent performance on different datasets and effectively handles imbalanced data samples, particularly excelling in detecting attacks from minority classes. But due to limited computational resources and time, our experiment had to use a downscaled version of the dataset when dealing with large-scale datasets. However, as the scale of the dataset became smaller, the minority samples available for learning decreased, leading to less than ideal detection results for some minority class attacks. So, the proposed method still has room for improvement. In the next step, we will improve the current method based on the aspects of the design of the reward function, the improved reinforcement learning algorithm, and the design of classify indicators and analysis indicators that can better reflect the minority class samples.

5. Conclusions

The present study proposes a novel intrusion detection method based on adaptive sample distribution dual-experience replay reinforcement learning, aiming to address the issue of imbalanced distribution in intrusion detection data samples. By introducing a secondary experience buffer in DQN specifically designed to store less, yet more challenging, minority class samples, the model effectively prevents the minority class samples from being overwhelmed by the majority class samples. Additionally, an adaptive dynamic update mechanism is employed, allowing for the flexible adjustment of the distribution of samples in the experience buffer, thus enabling the model to focus on the samples that are deemed difficult in the current stage, resulting in improved learning performance for the minority class. This innovative design enhances the generality and robustness of the model, allowing it to adapt to varying intrusion detection scenarios and better reflect the true distribution of the data. Experimental evaluation is conducted on three highly imbalanced

intrusion detection datasets, and the proposed model achieves an accuracy of 82.03% on NSL-KDD, 93.88% on AWID, and 84.39% on CICIoT2023, outperforming other models in terms of precision, especially for the minority class. Therefore, the method proposed in this paper serves as an effective approach to address the significant data distribution disparities in intrusion detection tasks. However, there is still room for improvement in the overall accuracy of our model, as well as its detection of minority class intrusion behaviors. Moving forward, we will continue enhancing the algorithm and RL environment to achieve better intrusion detection performance.

In this work, we provide feedback to the agent based on the true labels in the dataset and the agent's prediction; these labels are typically manually marked by network engineers or researchers. As a foundational work, this paper validates the feasibility and effectiveness of our method based on a labeled dataset, offers a promising intrusion detection paradigm, and lays the foundation for constructing an agent with good initial optimization space.

Our subsequent work will not rely on labeled datasets but will focus on self-collected traffic or log data, which are closer to actual application scenarios. Under these conditions, the feedback received by the agent originates from the positive and negative impacts of its behavior on the network environment. The calculation of these impacts depends on a predefined rule engine and pretrained machine learning model, generating feedback in an automated way. Ultimately, combined with the pretraining samples and initialized crucial parameters generated from the work in this paper, we will achieve adaptive learning and adjustment of the optimal detection strategy in a dynamic environment using reinforcement learning methods. Meanwhile, we respect the current state of the cyber security field. As one of the key areas, the combination of AI and manual methods is currently feasible and more easily accepted by the industry, but gradual development towards more comprehensive automation requires more effort.

Author Contributions: Methodology, H.T. and L.W.; Validation, H.T., D.Z. and J.D.; Formal analysis, H.T., D.Z. and J.D.; Investigation, H.T., D.Z. and J.D.; Resources, L.W.; Writing—original draft, H.T.; Writing—review & editing, H.T. and L.W.; Visualization, H.T.; Supervision, L.W.; Project administration, L.W.; Funding acquisition, L.W. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by Guangdong Basic and Applied Basic Research Foundation (2023A1515011698), Guangdong High-level University Foundation Program (SL2022A03J00918), Major Key Project of PCL (PCL2022A03), and National Natural Science Foundation of China (Grant No.62372137).

Data Availability Statement: The data will be made available by the authors on request.

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

References

1. Kim, H.; Choi, H.; Kang, H.; An, J.; Yeom, S.; Hong, T. A systematic review of the smart energy conservation system: From smart homes to sustainable smart cities. *Renew. Sustain. Energy Rev.* **2021**, *140*, 110755. [CrossRef]
2. Bhatti, G.; Mohan, H.; Singh, R.R. Towards the future of smart electric vehicles: Digital twin technology. *Renew. Sustain. Energy Rev.* **2021**, *141*, 110801. [CrossRef]
3. Kirimtat, A.; Krejcar, O.; Kertesz, A.; Tasgetiren, M.F. Future trends and current state of smart city concepts: A survey. *IEEE Access* **2020**, *8*, 86448–86467. [CrossRef]
4. Kaur, J.; Ramkumar, K. The recent trends in cyber security: A review. *J. King Saud Univ.-Comput. Inf. Sci.* **2022**, *34*, 5766–5781. [CrossRef]
5. Ahmad, Z.; Shahid Khan, A.; Wai Shiang, C.; Abdullah, J.; Ahmad, F. Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Trans. Emerg. Telecommun. Technol.* **2021**, *32*, e4150. [CrossRef]
6. Khraisat, A.; Gondal, I.; Vamplew, P.; Kamruzzaman, J. Survey of intrusion detection systems: Techniques, datasets and challenges. *Cybersecurity* **2019**, *2*, 20. [CrossRef]
7. Abdelkhalek, A.; Mashaly, M. Addressing the class imbalance problem in network intrusion detection systems using data resampling and deep learning. *J. Supercomput.* **2023**, *79*, 10611–10644. [CrossRef]

8. Gonzalez-Cuautle, D.; Hernandez-Suarez, A.; Sanchez-Perez, G.; Toscano-Medina, L.K.; Portillo-Portillo, J.; Olivares-Mercado, J.; Perez-Meana, H.M.; Sandoval-Orozco, A.L. Synthetic minority oversampling technique for optimizing classification tasks in botnet and intrusion-detection-system datasets. *Appl. Sci.* **2020**, *10*, 794. [CrossRef]
9. Tama, B.A.; Lim, S. Ensemble learning for intrusion detection systems: A systematic mapping study and cross-benchmark evaluation. *Comput. Sci. Rev.* **2021**, *39*, 100357. [CrossRef]
10. Fitni, Q.R.S.; Ramli, K. Implementation of ensemble learning and feature selection for performance improvements in anomaly-based intrusion detection systems. In Proceedings of the 2020 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT), Bali, Indonesia, 7–8 July 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 118–124.
11. Lee, J.; Park, K. GAN-based imbalanced data intrusion detection system. *Pers. Ubiquitous Comput.* **2021**, *25*, 121–128. [CrossRef]
12. Andresini, G.; Appice, A.; De Rose, L.; Malerba, D. GAN augmentation to deal with imbalance in imaging-based intrusion detection. *Future Gener. Comput. Syst.* **2021**, *123*, 108–127. [CrossRef]
13. Silver, D.; Huang, A.; Maddison, C.J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. Mastering the game of Go with deep neural networks and tree search. *Nature* **2016**, *529*, 484–489. [CrossRef]
14. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 5998–6008.
15. Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language models are few-shot learners. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 1877–1901.
16. Kiran, B.R.; Sobh, I.; Talpaert, V.; Mannion, P.; Al Sallab, A.A.; Yogamani, S.; Pérez, P. Deep reinforcement learning for autonomous driving: A survey. *IEEE Trans. Intell. Transp. Syst.* **2021**, *23*, 4909–4926. [CrossRef]
17. Wu, T.; Zhou, P.; Liu, K.; Yuan, Y.; Wang, X.; Huang, H.; Wu, D.O. Multi-agent deep reinforcement learning for urban traffic light control in vehicular networks. *IEEE Trans. Veh. Technol.* **2020**, *69*, 8243–8256. [CrossRef]
18. Bae, H.; Kim, G.; Kim, J.; Qian, D.; Lee, S. Multi-robot path planning method using reinforcement learning. *Appl. Sci.* **2019**, *9*, 3057. [CrossRef]
19. Caminero, G.; Lopez-Martin, M.; Carro, B. Adversarial environment reinforcement learning algorithm for intrusion detection. *Comput. Netw.* **2019**, *159*, 96–109. [CrossRef]
20. 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]
21. Idrissi, M.J.; Alami, H.; El Mahdaouy, A.; El Mekki, A.; Oualil, S.; Yartaoui, Z.; Berrada, I. Fed-anids: Federated learning for anomaly-based network intrusion detection systems. *Expert Syst. Appl.* **2023**, *234*, 121000. [CrossRef]
22. Asif, M.; Abbas, S.; Khan, M.; Fatima, A.; Khan, M.A.; Lee, S.W. MapReduce based intelligent model for intrusion detection using machine learning technique. *J. King Saud-Univ.-Comput. Inf. Sci.* **2022**, *34*, 9723–9731. [CrossRef]
23. Zhang, Z.; Wang, L.; Chen, G.; Gu, Z.; Tian, Z.; Du, X.; Guizani, M. STG2P: A two-stage pipeline model for intrusion detection based on improved LightGBM and K-means. *Simul. Model. Pract. Theory* **2022**, *120*, 102614. [CrossRef]
24. Bagui, S.; Li, K. Resampling imbalanced data for network intrusion detection datasets. *J. Big Data* **2021**, *8*, 6. [CrossRef]
25. Cao, B.; Li, C.; Song, Y.; Qin, Y.; Chen, C. Network intrusion detection model based on CNN and GRU. *Appl. Sci.* **2022**, *12*, 4184. [CrossRef]
26. Shafieian, S.; Zulkernine, M. Multi-layer stacking ensemble learners for low footprint network intrusion detection. *Complex Intell. Syst.* **2023**, *9*, 3787–3799. [CrossRef]
27. Thakkar, A.; Lohiya, R. Attack classification of imbalanced intrusion data for IoT network using ensemble learning-based deep neural network. *IEEE Internet Things J.* **2023**, *10*, 11888–11895. [CrossRef]
28. Ren, H.; Tang, Y.; Dong, W.; Ren, S.; Jiang, L. DUEN: Dynamic ensemble handling class imbalance in network intrusion detection. *Expert Syst. Appl.* **2023**, *229*, 120420. [CrossRef]
29. Chui, K.T.; Gupta, B.B.; Chaurasia, P.; Arya, V.; Almomani, A.; Alhalabi, W. Three-stage data generation algorithm for multiclass network intrusion detection with highly imbalanced dataset. *Int. J. Intell. Netw.* **2023**, *4*, 202–210. [CrossRef]
30. Dina, A.S.; Siddique, A.; Manivannan, D. Effect of balancing data using synthetic data on the performance of machine learning classifiers for intrusion detection in computer networks. *IEEE Access* **2022**, *10*, 96731–96747. [CrossRef]
31. Gaggero, G.B.; Caviglia, R.; Armellin, A.; Rossi, M.; Girdinio, P.; Marchese, M. Detecting cyberattacks on electrical storage systems through neural network based anomaly detection algorithm. *Sensors* **2022**, *22*, 3933. [CrossRef]
32. Al-Abassi, A.; Sakhnini, J.; Karimipour, H. Unsupervised stacked autoencoders for anomaly detection on smart cyber-physical grids. In Proceedings of the 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Toronto, ON, Canada, 11–14 October 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 3123–3129.
33. Fausto, A.; Gaggero, G.B.; Patrone, F.; Girdinio, P.; Marchese, M. Toward the integration of cyber and physical security monitoring systems for critical infrastructures. *Sensors* **2021**, *21*, 6970. [CrossRef] [PubMed]
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. Benaddi, H.; Ibrahimi, K.; Benslimane, A.; Jouhari, M.; Qadir, J. Robust enhancement of intrusion detection systems using deep reinforcement learning and stochastic game. *IEEE Trans. Veh. Technol.* **2022**, *71*, 11089–11102. [CrossRef]
36. Mohamed, S.; Ejbali, R. Deep SARSA-based reinforcement learning approach for anomaly network intrusion detection system. *Int. J. Inf. Secur.* **2023**, *22*, 235–247. [CrossRef]

37. 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. [CrossRef]
38. Dake, D.K.; Gadze, J.D.; Klogo, G.S.; Nunoo-Mensah, H. Multi-agent reinforcement learning framework in sdn-iot for transient load detection and prevention. *Technologies* **2021**, *9*, 44. [CrossRef]
39. 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]
40. García, S.; Luengo, J.; Herrera, F. Tutorial on practical tips of the most influential data preprocessing algorithms in data mining. *Knowl.-Based Syst.* **2016**, *98*, 1–29. [CrossRef]
41. Ramírez-Gallego, S.; Krawczyk, B.; García, S.; Woźniak, M.; Herrera, F. A survey on data preprocessing for data stream mining: Current status and future directions. *Neurocomputing* **2017**, *239*, 39–57. [CrossRef]
42. Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; Zaremba, W. Openai gym. *arXiv* **2016**, arXiv:1606.01540.
43. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv* **2013**, arXiv:1312.5602.
44. 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 October 2009; IEEE: Piscataway, NJ, USA, 2009; pp. 1–6.
45. KDD Cup 1999 Data. Available online: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html (accessed on 17 October 2023).
46. 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]
47. Neto, E.C.P.; Dadkhah, S.; Ferreira, R.; Zohourian, A.; Lu, R.; Ghorbani, A.A. CICIoT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment. *Sensors* **2023**, *23*, 5941. [CrossRef] [PubMed]
48. Yang, H.; Zeng, R.; Xu, G.; Zhang, L. A network security situation assessment method based on adversarial deep learning. *Appl. Soft Comput.* **2021**, *102*, 107096. [CrossRef]
49. Thanthrige, U.S.K.P.M.; Samarabandu, J.; Wang, X. Machine learning techniques for intrusion detection on public dataset. In Proceedings of the 2016 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), Vancouver, BC, Canada, 15–18 May 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 1–4.