Journal of Big Data

RESEARCH Open Access

MAFSIDS: a reinforcement learning-based intrusion detection model for multi-agent feature selection networks

Kezhou Ren[1], Yifan Zeng[1], Yuanfu Zhong[1], Biao Sheng[1] and Yingchao Zhang[1*]

*Correspondence:
zhangych68@mail.sysu.edu.cn

[1] School of Systems Sciences and Engineering, Sun Yat-Sen University, Guangzhou 510006, China

Abstract

Large unbalanced datasets pose challenges for machine learning models, as redundant and irrelevant features can hinder their effectiveness. Furthermore, the performance of intrusion detection systems (IDS) can be further degraded by the emergence of new network attack types. To address these issues, we propose MAFSIDS (Multi-Agent Feature Selection Intrusion Detection System), a DQL (Deep Q-Learning) based IDS.

MAFSIDS comprises a feature self-selection algorithm and a DRL (Deep Reinforcement Learning) attack detection module. The feature self-selection algorithm leverages a multi-agent reinforcement learning framework, which redefines the feature selection problem by converting the traditional 2^N feature selection space into N agent representations. This approach reduces model complexity and enhances the search strategy for feature selection. To ensure accurate feature representation and expedite the feature selection process, we have also developed a GCN (Graph Convolutional Network) method that extracts deeper features from the data. The DRL attack detection module utilizes the Mini-Batchs technique to encode the data, allowing reinforcement learning to be applied in a supervised learning context. This integration improves accuracy. Additionally, the policy network in this module is designed to be minimalist, enhancing model efficiency. To evaluate the performance of our model, we conducted comprehensive simulation experiments using Python. We tested the model using the CSE-CIC-IDS2018 and NSL-KDD datasets, achieving impressive accuracy rates of 96.8% and 99.1%, as well as F1-Scores of 96.3% and 99.1%, respectively. The selected feature subset successfully eliminates approximately 80% of redundant features compared to the original feature set. Furthermore, we compared our proposed model with other popular machine-learning models.

Introduction

In recent years, the Internet has developed rapidly globally, becoming an integral part of people's daily lives. This widespread adoption of network information systems has encompassed various industries, such as online social networking, traffic planning, business transactions, the Internet of Things, and military surveillance operations [1, 2]. The scale of network data on a global level is projected to reach 400 exabytes per month, a significant increase compared to the 120 exabytes per month recorded in 2017. This

exponential growth in network data necessitates the implementation of effective information systems to safeguard against threats like information leakage and cyber-attacks, ensuring the secure transmission of network information and thereby supporting the sustainable development of human society and the economy [3, 4]. However, there are currently two main issues that need to be addressed:

Feature Selection Challenges: Network traffic data often possess many features, many of which might need to be more relevant or relevant. Feature selection is crucial in extracting meaningful attributes while eliminating unnecessary ones. However, traditional feature selection methods might need to improve on large-scale and imbalanced datasets, consequently affecting the performance of intrusion detection systems (IDS).

Emergence of New Attack Types: With the continuous advancement of technology, novel network attack types continue to emerge, posing challenges for existing IDS that may need help to promptly and accurately identify these new attack categories. As a result, IDS requires a certain level of adaptability to adjust and recognize unknown attack types swiftly.

Researchers have developed the Intrusion Detection System (IDS) to maintain network security and guard against evolving and covert network threats. This system can identify anomalous network behaviors by monitoring the network for abnormal activity and intrusion threats and directing information systems to respond to the threats. Denning presented an intrusion detection system based on audit data and statistical approaches as early as 1987 [5].

Studies have identified two primary forms of IDS [6]. The first is signature-based IDS, which compares network traffic to known attack signatures. However, it is less effective against new threats. The second is anomaly-based IDS, which detects unknown threats by establishing a normal network model and flagging deviations from it.

Anomaly-based intrusion detection systems have gained dominance in IDS by incorporating machine learning techniques such as decision tree (DT), deep learning, and reinforcement learning [7–12]. While machine learning improves the accuracy of identifying unusual traffic, traditional techniques like DT lack the ability to extract deep network information. In recent years, deep learning has emerged as a powerful approach, extracting crucial information using multilayer neural networks and continuously improving the training model to detect network attacks [13]. However, deep learning's effectiveness in identifying unknown network attacks is limited and highly dependent on data [14, 15]. Reinforcement learning allows real-time interaction, autonomous decision-making, and learning from environmental incentives. Combining reinforcement learning with deep learning results in a deep reinforcement learning approach, which can enhance the effectiveness of IDS in detecting network threats [12].

Our study is based on this technique, where Emmons et al. [16] utilized a two-layer feedforward MLP combined with supervised learning from offline reinforcement learning. Their approach, known as RvS, was compared with complex methods like a transformer for sequence modeling and showed competitive results. Dong et al. [9] proposed an optimization approach for network anomaly traffic detection using semi-supervised reinforcement learning DDQN (SSDDQN). They compared it with conventional machine learning techniques and demonstrated that SSDDQN could accurately identify unknown network threats.

Feature selection is a crucial technique in IDS models to address challenges such as high false alarm rates and poor computational efficiency caused by duplicated characteristics in network data [17]. By selecting the most relevant subset of features, feature selection enhances the performance of IDS models by eliminating redundancy and improving prediction accuracy and efficiency [18–21]. Ren et al.'s ID-RDRL model [22], which combines Recursive Feature Elimination (RFE) with Deep Q-Network (DQN) reinforcement learning, effectively reduces redundant features in network data by approximately 80% and improves the accuracy of identifying network attacks. This integration of feature selection and reinforcement learning demonstrates the potential for optimizing IDS models and enhancing network threat detection capabilities.

Graph Convolutional Network (GCN) is a highly effective neural network for node classification tasks, particularly in capturing graph semantics and extracting topology information [23]. In the realm of network intrusion detection, where data features exhibit correlations, researchers have harnessed the power of GCN to construct graph structures that capture deep connections between these features [8, 24]. Notably, Liu et al. introduced GCNID, a specialized graph convolutional neural network for multiclassification intrusion detection, which showcased improved detection accuracy through simulations [25]. Similarly, Zhang et al. proposed an intrusion detection framework based on GCN for industrial IoT applications, verifying its accuracy and robustness through experiments on a publicly available dataset [26]. These advancements highlight the potential of GCN in enhancing intrusion detection performance by leveraging graph-based approaches.

Reinforcement learning is a powerful approach that emphasizes a model's ability to independently explore and solve problems, finding applications in various decision-making scenarios. Applying deep reinforcement learning (DRL) in Intrusion Detection Systems (IDS) offers advantages such as handling uncertain environments, enhancing the ability to identify unknown attacks, and possessing strong generalization capabilities. In the era of big data and complex network environments, integrating multi-agent reinforcement learning holds the promise of enabling Intrusion Detection Systems (IDS) to intelligently recognize unfamiliar network settings, thereby enhancing network security and efficiency [27]. Researchers, such as Liu et al. [28], have proposed multi-agent frameworks that leverage reinforcement learning for feature selection, showing superior performance compared to traditional methods. Additionally, Wang et al. [29] have suggested a multi-agent body-based recommendation system Auto-ML framework, which adaptively selects feature fields and incorporates performance enhancements. These advancements demonstrate the potential of multi-agent reinforcement learning to improve IDS by addressing complex network challenges and selecting optimal features, ultimately strengthening network security and operational effectiveness.

In this paper, we address the problem of automatic feature selection and subspace exploration in Intrusion Detection Systems (IDS). We propose a novel IDS called MAFSIDS (Multi-Agent Feature Selection Intrusion Detection System) that leverages Deep Q-Learning (DQL) and the MAFS (Multi-Agent Feature Selection) method for improved feature selection and network attack classification. By adopting a multi-agent reinforcement learning framework, we redefine the feature selection problem and reduce model complexity by representing the feature space with N feature agents

instead of the traditional 2^N representation. To extract deeper features, we utilize the Graph Convolutional Neural Network (GCN) method. The model is trained using mini-batches, combining reinforcement learning with supervised learning to enhance accuracy. The policy network is designed with a minimalist neural network architecture for improved efficiency. We conduct comprehensive simulation experiments on the CSE-CIC-IDS2018 and NSL-KDD datasets, achieving impressive results. The model achieves high accuracy rates of 96.8% and 99.1%, along with F1-Scores of 96.3% and 99.1% on the respective datasets. Our proposed feature selection method reduces redundant features by approximately 80% compared to the original set. Comparative evaluations with other popular machine learning models demonstrate the superior performance of our approach in terms of AUC and F1-Score. Overall, our study presents an efficient IDS solution that combines feature selection, deep learning, and reinforcement learning to enhance network security. The abbreviations used in this paper are as shown in Table 1.

The contributions of the paper are outlined below:

1. We introduce MAFSIDS, a DQN-based intrusion detection system, which incorporates the MAFS feature selection approach. MAFSIDS consists of two main components: a feature self-selection algorithm and a DRL attack detection module. The feature self-selection algorithm addresses the feature selection problem by leveraging a multi-agent reinforcement learning framework. This framework allows N features to compete with each other, effectively reducing the feature selection space from 2^N to a more manageable subset. The DRL attack detection module utilizes a simple policy network, enabling the model to provide fast responses, making it suitable for mod-

Table 1 List of abbreviations. (sorting according to the order of appearance in the text)

Abbreviation	Full form	Abbreviation	Full form
IDS	Intrusion Detection System	DRL	Deep Reinforcement Learning
DDoS	Distributed Denial of Service	SVM	Support Vector Machine
GCN	Graph Convolutional Networks	LASSO	Least Absolute Shrinkage and Selection Operator
CNN	Convolutional Neural Network	KNN	K-Nearest Neighbor
KDD99	KDD CUP 99 Dataset	RF	Random Forest
NSL-KDD	NSL-KDD Intrusion Detection Dataset	Auto-ML	Automated Machine Learning
ML	Machine Learning	LSTM	Long Short-Term Memory
RFE	Recursive Feature Elimination	DL	Deep Learning
DT	Decision Tree	ANN	Artificial Neural Network
IoT	Internet of Things	DoS	Denial of Service
IVN	In-vehicle Networking	DM	Data Mining
DQN	Deep Q-Network	DDQN	Double Deep Q-Network
PG	Policy Gradient	AC	Actor Critical
MLP	Multilayer Perceptron	MDP	Markov Decision Process
VANET	Vehicular Ad-hoc Network	ROC	Receiving Operating Characteristics Curve
AUC	Area Under the ROC Curve	RL	Reinforcement Learning
GBM	Gradient Boosting Machine	MAFSIDS	Multi-Agent Feature Selection Intrusion Detection System

ern IDSs. Furthermore, the model supports online learning, enabling it to adapt to changing network data environments.

2. The feature self-selection algorithm employs a multi-agent approach to control individual features separately, determining their final selection status. It can effectively identify the optimal subset of features. Additionally, we have developed a Graph Convolutional Network method to extract deeper features from the data, improving feature representation accuracy and expediting the feature selection process.
3. The DRL attack detection module utilizes the Mini-Batchs method to encode the data, allowing reinforcement learning to be applied within a supervised learning framework. The policy network employed in this module is designed to be minimalist, optimizing model efficiency.
4. To evaluate the performance of our proposed model, we conducted comprehensive simulation experiments using Python. We tested the model's efficacy using two datasets: the CSE-CIC-IDS2018 dataset, which includes novel cyber attacks, and the traditional NSL-KDD dataset. The results demonstrate high accuracy rates of 96.8% and 99.1%, as well as F1-Scores of 96.3% and 99.1%, respectively. Additionally, the selected feature subset reduces redundant features by approximately 80% compared to the original feature set. Furthermore, we compared our model against other popular machine-learning models.

Related works

In this research endeavor, our focus is on developing a multi-agent feature selection intrusion detection system. This innovative model utilizes multiple agents to perform feature selection. Consequently, our investigation encompasses the realm of feature selection and its connection to the Auto-ML model.

Feature selection is a prominent research topic [30, 31] that may eliminate duplicate features from datasets, minimize model computation, and enhance model performance. It has been extensively explored and used in academia and industry. The research on feature selection may be categorized into three groups: (1) Filtering approaches, such as univariate feature selection and correlation-based feature selection, that employ quantitative scores to determine the value of characteristics [32, 33]. This approach is appropriate for selecting features from high-dimensional data, but it may disregard the correlation and interaction between feature subsets of various feature combinations and predictor factors [34]. Wrapper approaches include evolutionary algorithms, RFE, and branch-and-bound algorithms, which combine particular search tactics with prediction objectives to select the optimal feature subset [35, 36]. This approach offers better performance than filtering methods, but it needs the computation of all potential feature subsets, which might result in a more significant computational burden [37]; (2) Embedded approaches, which include feature selection techniques into the model task and The optimal subset of features, are filtered according to the influence of the feature selection technique on model performance [38, 39], including LASSO and decision trees [40–42]. Unfortunately, this strategy depends on how the feature selection and prediction tasks are coupled and may be incompatible with classifiers. By incorporating Feature Selection (FS), IDS is enabled to choose the optimal subset from the feature set, thereby

enhancing performance. Moreover, by eliminating numerous redundant features, this not only reduces the computational load of the model but also significantly decreases testing time.

GCN is a neural network that excels in node classification tasks and can extract deep topological information among feature nodes to obtain better feature representations. Deng et al. [43] proposed a flow topology-based graph convolutional network (FT-GCN) label-constrained IoT intrusion detection method to reduce the dependence of IDS on labeled data, exploiting the potential traffic topology with limited labels to unlock the full potential of traffic flow data and obtained good results on three real-world datasets. Cheng et al. [44] proposed a framework called Alert-GCN to correlate alerts belonging to the same attack using graph convolutional networks (GCN), and the results showed that Alert-GCN outperformed traditional classification models in correlating alerts. Zhou et al. [45] proposed a new hierarchical adversarial attack (HAA) generation method to address the vulnerability of graph neural networks to unbalanced intrusion detection datasets and validated the model on the public dataset UNSW-SOSR2019 dataset, which can reduce the classification accuracy by 30%.

The Auto-ML model is a machine learning model based on reinforcement learning consisting of numerous agents interacting and learning in a complicated environment. Liu et al. [46] established a strategy for distinct task formulation that enables fast architecture search using gradient descent. They discovered high-performance convolutional and recursive architectures for image classification and language modeling. Lin et al. [47] suggested a strategy for resolving large-scale fleet management problems using reinforcement learning, which alleviates traffic congestion by reallocating traffic resources; an actual investigation demonstrated considerable increases in traffic efficiency. Zeynivand et al. [48] examined the key factors affecting the quality of VANET networks and used a multi-agent learning approach to improve quality of service; the results indicate that the proposed method outperforms previous methods in terms of packet delivery rate (PDR) and transaction success rate. Numerous research has explored ways, in which an agent is used to select a subset of data characteristics [49]. Unfortunately, this method needs the agent to decide whether to include or exclude each characteristic [28]. This results in an exponential increase of the explorable interval (2^N), similar to evolutionary algorithms [36, 38], and it is challenging to discover the optimal global solution and has a high computing cost.

Work description

This section introduces the MAFSIDS (Multi-Agent Feature Selection Intrusion Detection System) model proposed in this study. The framework overview outlines the structure of the model, while the subsequent sections focus on its key components. These components include the data preprocessing module, which prepares the input data for analysis, the GCN feature selection method, which extracts deep features from the data,

the multi-agent body feature selection algorithm, which selects optimal features, and the Mini-Batchs module, which handles data encoding. Lastly, the DRL module applies deep reinforcement learning to identify network attacks. Each of these components plays a crucial role in the overall functioning of the MAFSIDS model.

Framework overview

Figure 1 illustrates the framework of our proposed model. The model consists of three main sections and five modules: the data preprocessing module, the GCN feature selection module, the multi-agent body feature selection module, the Mini-Batchs data creation module, and the DRL module. These modules work together to handle various stages of the intrusion detection process, from data preparation to feature selection, selection, and finally, utilizing deep reinforcement learning techniques. The framework provides a comprehensive approach to effectively detect and classify network attacks.

The Data preprocessing module plays a crucial role in our framework as it focuses on preparing the network assault dataset. This module consists of four essential steps: Integration, Cleaning, Conversion, and Normalization. During Integration, the dataset is combined and organized to ensure a unified format. The Cleaning step involves removing any inconsistencies, errors, or missing values from the data. Conversion transforms the data into a suitable format for further analysis, while Normalization standardizes the data to a common scale. By performing these steps, the dataset becomes more refined, consistent, and ready for subsequent modules within the framework.

Our research is dedicated to the automated selection of features. In this process, we employ Graph Convolutional Network (GCN) to deeply extract data features. Although this step may demand considerable computational resources, its importance

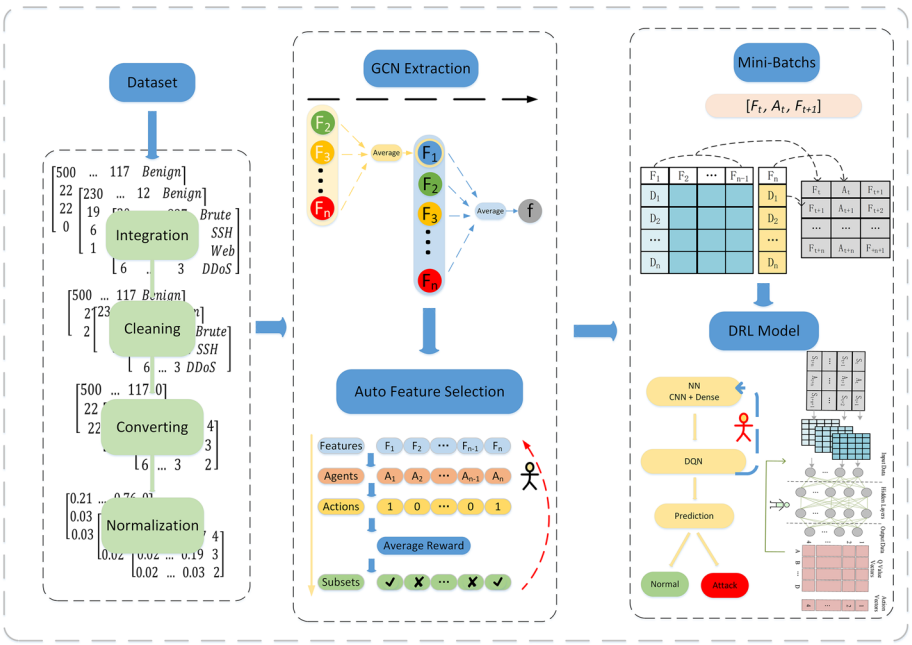

Fig. 1 Framework Overview

in accurately representing features cannot be overstated. Through GCN, we are able to capture richer and more abstract features within the data, providing a more precise and potent feature representation for subsequent tasks and serves as input to the feature selection component. Furthermore, the introduction of the multi-agent agent feature selection submodule transforms the traditional 2^N feature selection space into a competition among N agents collaborating to select features. This innovative approach, achieved through the synergy of agents, effectively reduces the feature space and enhances the efficiency of feature selection. We firmly believe that this multi-agent methodology is better equipped to address intricate feature selection challenges, thereby furnishing the model with a more robust feature subset.

In the Deep reinforcement learning to identify the attack section, the Mini-Batchs module recodes the feature subset dataset filtered by multiple agents into the data form $[Ft, At, Ft + 1]$ and feeds it to the DRL classifier module, which extracts the feature information of the input data by CNN and inputs it to the fully connected layer, and the agent can control the fully connected network to classify the input of network traffic for predictive classification.

Data preprocessing module

Integration of data

The CSE-CIC-IDS2018 dataset comprises raw traffic data encompassing 15 attack scenarios gathered over a 10-day period by ten networks. We performed an undersampling process on the combined dataset to equalize the percentage of normal and attack traffic, resulting in a dataset comprising around 8.9 million samples. The identical preparation procedure was applied to both this and the NSL-KDD datasets. Note that the CSE-CIC-IDS2018 dataset will be supplied with the data pretreatment procedure.

Cleaning of data

We removed about 2000 samples with missing or duplicate values from the original dataset to avoid invalid samples. Take the CSE-CIC-IDS2018 dataset as an example. The cleaned dataset had 77 features and 8,874,005 samples.

Conversion of data

Based on Fitni's work [50], we grouped the 15 attack categories in the original dataset into seven types: Benign, BruteForce, DoS, Bot, DDoS, Web Attacks, and Infiltration., where Benign represents normal traffic.

Normalization of data

Using a normalization procedure, we normalize all original features to a range of 0 to 1. This is because some features, such as "Port Number" and "Packet Size", have large and varying ranges of values that can affect the model's performance and increase the computational cost. As defined in Eq. 1.

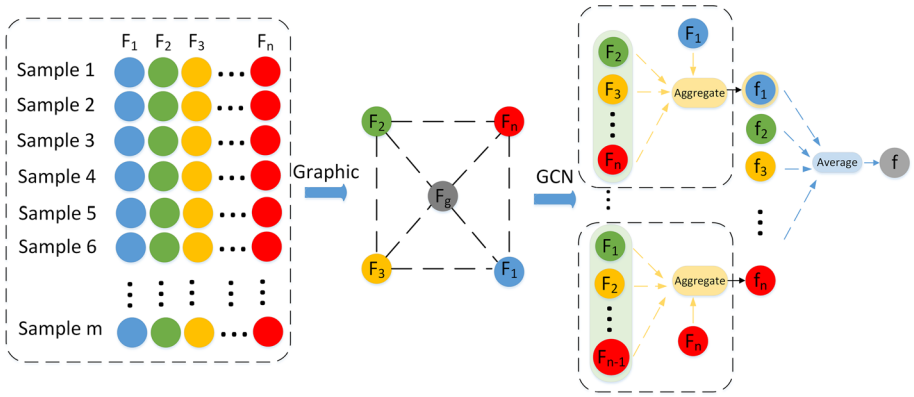

Fig. 2 GCN feature selection module

$$x^{'} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{1}$$

where $x^{'}$ represents the normalized eigenvalue, x represents the initial eigenvalue, x_{min}, x_{max} represents the minimal eigenvalue and the maximum eigenvalue, respectively.

Feature selection

GCN feature selection

Figure 2 illustrates the process of transforming the selected input data into a dynamic graph network to effectively capture the relationships between attributes. Initially, the input data is organized into a matrix of dimensions $[m \times n]$, where each column represents a random selection of features. By utilizing the feature correlation graph G, we establish a straightforward approach to represent the correlations between features. This method enables us to effectively analyze and understand the interdependencies among the selected attributes.

Since this study focuses on network attack identification for IDS, not on designing complex node embedding graph models, we choose the common GCN approach to extract the correlation between feature columns, a very advanced graph embedding model that has been applied in many graph-based tasks. The neural network layer formulation of GCN, as defined in Eq. (2) shows.

$$H^{(l+1)} = f(H^{(l)}, A) \tag{2}$$

where H^0 is the input data with $M * N$ dimensions, H^L is the feature node matrix of graph G, and A is the adjacency matrix of graph G. With the fast approximate convolution on the graph, we can further obtain the following layer-by-layer propagation multi-layer graph convolution network (GCN) formulation, as defined in Eq. 3.

$$f\left(H^{(l)}, A\right) = \sigma(\widehat{D}^{-\frac{1}{2}}\widehat{A}\widehat{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}) \tag{3}$$

where $\widehat{A} = A + I_N$ is the adjacency matrix of the undirected graph G with additional self-connections, I_N is the unit matrix, and $\widehat{D}_{ii} = \sum_j \widehat{A}_{ij}$ is the specific trainable weight matrix in the layer connections, and σ denotes the activation function.

In Fig. 2, the GCN (Graph Convolutional Network) feature selection module is depicted. On the left side of the figure, there are independent samples labeled as sample 1, sample 2, and m other randomly selected samples. The value of m is determined by the Batch-Size, and each sample consists of n features, resulting in a matrix of size $[m \times n]$. Each column of this matrix represents a separate feature column denoted as F_1, F_2, and F_n., with a shape of $[m \times 1]$.

These extracted feature columns are considered as nodes and collectively form a dynamic graph$G$. On the right side of Fig. 2, for the nodeF_1, all the remaining nodes in its neighborhood are treated as a whole, and their information is aggregated to create a new feature node associated withF_1. This process is repeated for each feature node, resulting in a set of feature nodes$[f_1, f_2, ..., f_n]$. Finally, these feature nodes are averaged to obtain the real feature node$f$, which represents the $[m \times n]$ data matrix in a condensed form.

Automatic feature selection module

In Fig. 3, we demonstrate the process of generating an input feature matrix $[M \times N]$ by randomly selecting M data samples from a dataset containing N features. The GCN feature selection module is then applied to extract a feature vector f, which becomes the input for the automated feature selection module. In this module, each feature is assigned an agent responsible for deciding whether to retain it based on the input data and the feedback from the environment. The automated feature selection module employs a multi-agent reinforcement learning framework, incorporating agents, states, actions, rewards, and policy functions, to form a decision process known as Markov Decision Process (MDP). By defining the MDP, the model learns the optimal policy function for each state through the agents and takes actions that maximize the overall reward R, as shown in Algorithm 1.

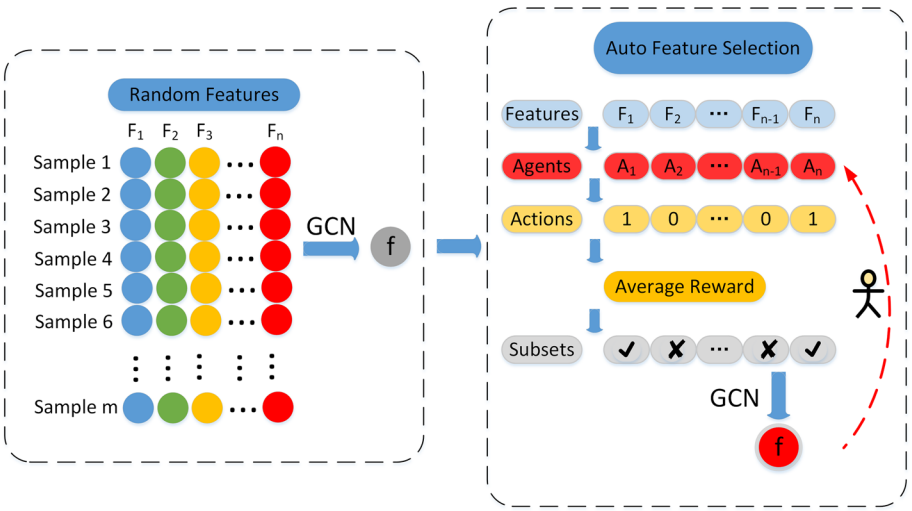

Fig. 3 Automatic feature selection module

Algorithm 1: Automatic feature selection algorithm

Input: all features set F in the dataset;

Output: the selected features subset F_i;

Step 1. Partition the data set into N parts and use F_n to represent the sub-data set

Step 2. Create M feature selection agents, a_i represents the i-th agent

Step 3. **for** each subset of F_i, $i = 1 \ldots N$ do

get the gcn_i of the F_i

for each agent of a_i, $i = 1 \ldots N$ do

$A_i =$ dqn - action[i]

Policy(s) = $\arg_a \max(Q(s,a))$, get Q function

get reward R,

$action_i =$ dqn(a_i, gcn_i),

for each agent of a_i, $i = 1 \ldots N$ do

calculate the reward R of the current agent and the accuracy,

$r_{agent} = r_t + \lambda * r_{t+1}$, get the r_{agent}

calculate the loss function and perform back propagation

end for

end for

get the current feature selection subsetF_i and calculate the current accuracy

if accuracy > $accuracy_{before}$ do

$F_{final} = F_i$

end if

end for

Step 4. Use the model corresponding to the optimal Fi feature subset and rank the features by importance, $F_i = (F_1 > F_2 > F_3 > \ldots)$

We detail the definition of each element in this paper framework.

Multiple Agents: In this paper, there are N agents corresponding to N feature vectors one by one, and the corresponding current feature is selected or not based on the state of the environment and the environmental reward.

State: The state S is the medium through which the agent interacts with the environment, and in the MDP, S is the feature selection vector f obtained through the GCN feature selection module. In order to enable the neural network to train the agent correctly, we fix the dimension of f to $[1 * M]$.

Action: action A is the choice made by an agent to maximize the total reward R after interacting with the environment. In the MDP, each agent makes a yes or no choice (denoted by "1" or "0"), where "1" means to keep the current corresponding feature and "0" means to discard it. "0" means discarding the current corresponding feature.

Reward: Reward R is the feedback given by the environment to the agent after it has made an action. In the MDP, only the agent that has completed the policy adjustment and the action selection is "1" is assigned the reward R, equally distributed to each agent.

Policy function: The policy function Q plays a central role in the Markov Decision Process (MDP), as it guides the agent's actions by approximating the optimal policy function

for each state. In this study, we employ a straightforward approach of utilizing a two-layer fully connected neural network to approximate the fitted policy function. Through continuous updates, the model refines itself and provides guidance to the agent's decision-making process.

As shown in Fig. 3, in the MARFS approach, each feature is assigned to a feature agent, and the actions of the feature agent determine the selection/non-selection of its corresponding feature, as shown in Eq. 4.

$$S^{i+1} = represent(F^i) \tag{4}$$

where F_i is a subset of features selected at time t. And *represent* is a representation learning algorithm that converts the dynamically varying F_i into a fixed-length state vector S^{i+1}. Representation methods can be graph-based GCN. reward r^i is for the selected feature subset F_i, as shown in Eq. 5.

$$r^i = eval(F^i) \tag{5}$$

where *eval* is an evaluation function on F_i, either as a supervised measure of the machine-learning task with F_i as input or as a supervised measure method, rewards are assigned to each feature agent to train their policy. As more and more steps are explored and utilized, the policies become more intelligent and more innovative, so they can find better and better subsets of features.

In the training phase, the agent independently trains the strategy by experiencing playback. For agent a_i, at time t, newly created tuples $\{s_i^t, a_i^t, r_i^t, s_i^{t+1}\}$, including state s_i^t, action a_i^t, reward r_i^t and next state s_i^{t+1}, are stored in each agent's memory. agent a_i uses its corresponding small batch of samples to train its Deep Q-Network (DQN) to obtain the maximum long-term reward based on the Bellman equation, as shown in Eq. 6.

$$Q\left(s_i^t, a_i^t|\theta_t\right) = r_i^t + \gamma maxQ\left(s_i^{t+1}, a_i^{t+1}|\theta_{t+1}\right) \tag{6}$$

where θ is the parameter set of Q network, and γ is the discount factor.

Deep reinforcement learning to identify attack

Mini-Batchs module

The CSE-CIC-IDS2018 and NSL-KDD datasets are supervised datasets containing category labels, rendering them inappropriate for unsupervised or semi-supervised learning techniques such as DRL. To modify these datasets for DRL, we encode the data using the Mini-Batch method. In particular, we regard all data characteristics, except the label, as the current state represented by S_t. The label characteristics are considered the current action, represented by A_t. Except for the label, all following data characteristics are handled as the next state, represented by S_{t+1}. As seen in Fig. 4, these three representations are then combined to provide the input data $[S_t, A_t, S_{t+1}]$ for the DRL module.

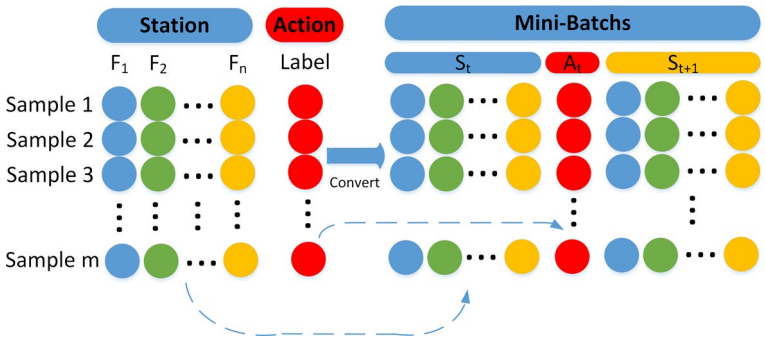

Fig. 4 Mini-Batchs module. The data encoding schematic for the DQN model

Mini-Batch randomly chooses a portion of the original dataset without duplicating the samples during each training session until the entire dataset has been explored. Figure 4 illustrates the Mini-Batch module's random selection and encoding of data from the dataset. In each training batch, $n + 1$ samples of $[S_t, A_t, S_{t+1}]$ are included. By applying the Mini-Batch method, we can adapt the supervised datasets to DRL to enable the IDS model to be efficiently trained. This method enables quick training of the model while guaranteeing that the training data correctly replicates the properties and features of the original dataset.

We modified the dataset to adapt to DRL and introduced the Mini-Batch method for preprocessing the raw data. This processing aligns the data format with the input requirements of the DRL model, enabling the application of unsupervised learning techniques to traditional supervised learning IDS, especially in intrusion detection classification tasks. We believe that this innovative attempt will bring new insights and approaches to the field of IDS, offering novel avenues for enhancing the performance of traditional IDS. The Mini-Batch process played a crucial role in data batch processing, streamlining data representation, and creating more favorable conditions for the application of DRL.

DRL module

DQN algorithm. The DQN algorithm is a reinforcement learning method that utilizes the Markov decision process (MDP) comprising State S, Action A, Reward R, and Value function Q. State S represents the current environment, which includes all input data except the Label feature. Action A is the agent's response to the environment feedback, represented by the Label feature. Reward R is the agent's action compared to the original Label, where a match is rewarded with 1, and a mismatch receives no reward (i.e., $R = 0 or 1$). The Q function estimates the value of each state and is continuously updated using the Agent process loss function. The DQN algorithm employs a deep neural network to approximate the Q-value function and estimate the Q-value of the current target, updating the network weights, as shown in Fig. 5.

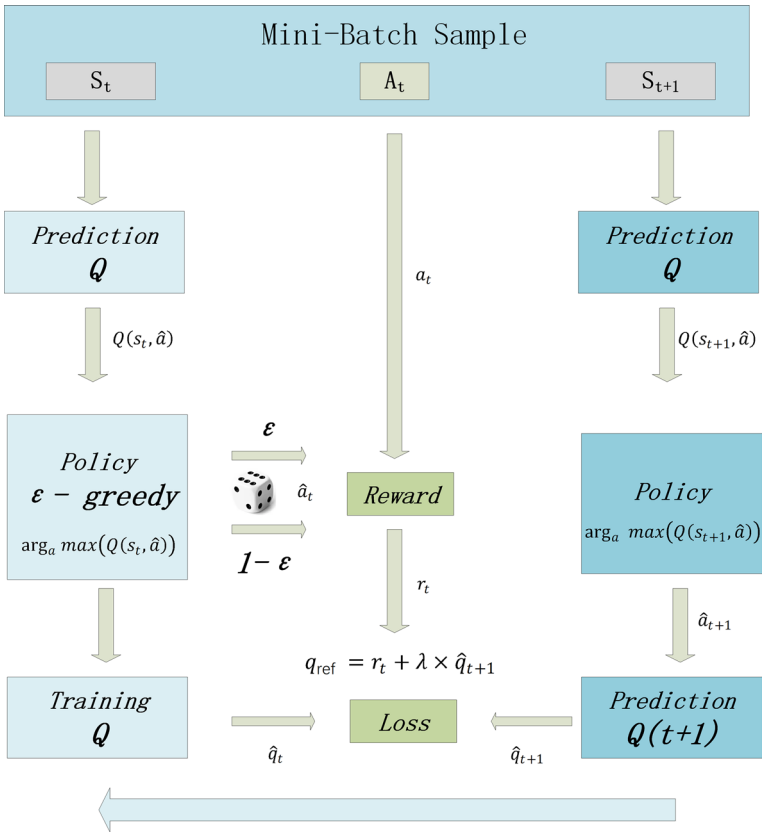

Fig. 5 Schematic diagram of DQN algorithm

Algorithm 2: Deep reinforcement learning-based algorithm for DRL module

Input: all features set F in the dataset;
Output: Modeling the identification of network attacks accuracy;
Step 1. Train the DQN model using all features
Step 2. Re-selection of the data by the subset of features obtained by Algorithm 1.
Step 3. **for** each subset of F_i, $i = 1 \ldots N$ do
train the DQN model using F_i features
Policy(s) = $\arg_a \max(Q(s,a))$, get Q function
get reward R,
Policy(s_{t+1}) = $\arg_a \max(Q(s_{t+1},a))$, get Q_{t+1} function
$q_{ref} = r_t + \lambda * q_{t+1}$, get the q_{ref}
find the loss of the DQN model
get the accuracy of the current model
backward propagation optimization model
end for
Step 4. Calculate the accuracy of the overall dataset by model

Figure 6 shows the DRL model with the Auto-FS, in which we get the selected optimal feature subset from Auto-FS and recode it by Mini-Batch to form a new dataset. With redundancy and irrelevance removed, feed it into the convolutional network to extract the deep features, and then the obtained feature data is passed through two

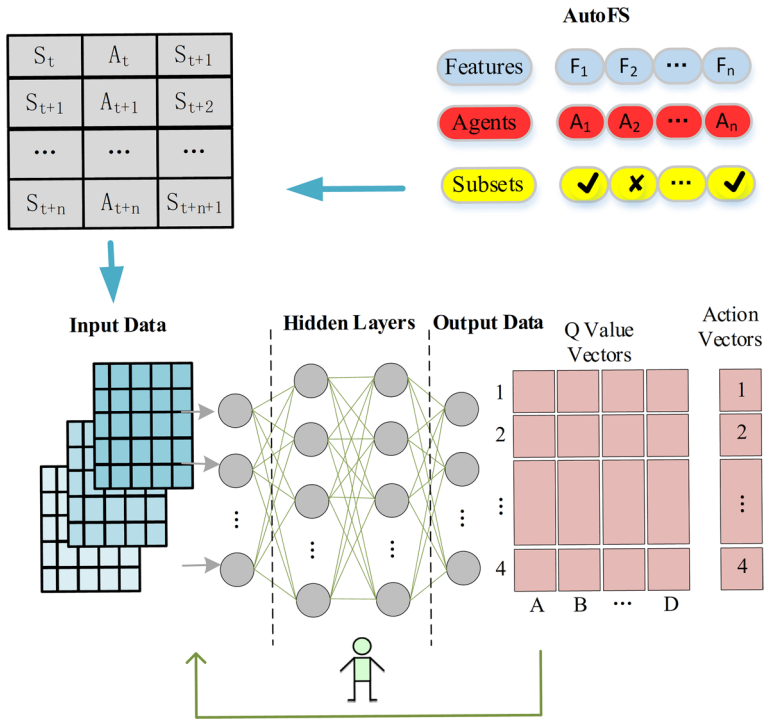

Fig. 6 Schematic diagram of DRL model with Auto-FS structure

simple fully-connected layers to make our DQN algorithm controls the classifier of the fully connected layer to implement the judgment of network attacks, and continuously feeds the data to train the whole DRL network model.

where the policy function is

$$Policy(s_{t+1}) = arg_a max(Q(s_{t+1}, a)) \tag{7}$$

Policy is the policy function, and we need to maximize the Q function of step S_{t+1} to approximate the policy function of the model, the correlation between different data is small for the supervised dataset. We calculate the Q function of two closed times t, as shown in Eq. 7.

$$q_{ref} = r_t + \lambda * q_{t+1} \tag{8}$$

where r is the reward at time t, and γ is the discount factor, as shown in Eq. 8.

Algorithm 2 demonstrates the application of the Environment, Agent, States, Actions, and Rewards ideas to network intrusion detection models based on the CSE-CIC-IDS2018 and NSL-KDD datasets. All characteristics except Label are employed as States in the DRL model, whereas Label features are used as Actions, and projected values serve as Outcomes. The dataset is preprocessed. Due to the supervised nature of the CSE-CIC-IDS2018 and NSL-KDD datasets, the Agent receives Rewards from the environment only after completing Actions and does not perform actual actions on the environment.

The Agent interacts with the environment by performing actions to forecast if the data indicates a network assault, using a greedy strategy to select between a random or optimum action with a probability of ϵ and $1 - \epsilon$, respectively. Actions and Rewards are crucial components of the DRL algorithm, allowing the agent and environment to communicate indirectly and reinforcement learning agents to acquire intelligence. The Label feature data columns are Actions, and *Q*-values created by the Agent are utilized to update the network and set *Q* thresholds that distinguish normal network traffic from network attacks.

Experiments and results

The "Hardware and Environment setting" section overviews the experimental hardware and environmental configurations. In the "Data Description" section, we introduce the two datasets used in this study, CSE-CIC-IDS2018, and NSL-KDD, employed to assess the model's performance. The "Evaluation Metrics" section discusses standard performance indicators for intrusion detection systems, including accuracy, F1 score, and ROC curve. In the "Results" section, we initially present the findings of ablation experiments, followed by an assessment of the overall performance of the MAFS model, including the confusion matrix, ROC curve, and an analysis of the impact of Lambda values. Finally, we compare our model to other conventional machine learning methods.

Hardware and environment setting

The experiments in our study were carried out on a Windows 11 operating system. The device's hardware configuration involved 32 GB of RAM, an AMD Ryzen 7 5800H processor, and an NVIDIA RTX3600 graphics card. For our experimental environment, we utilized Python 3.8, with model development performed using TensorFlow. The programming software used included VScode and PyCharm. To facilitate data processing and visualization, we employed various packages such as Scikit-Learn, NumPy, Pandas, and Matplotlib. For detailed information regarding the hardware and environment setup, please refer to Table 2.

Data description

CSE-CIC-IDS2018 dataset

The CSE-CIC-IDS2018 dataset is a recently produced intrusion detection system dataset containing seven attack scenarios, including Brute-force, Heartbleed, Botnet, DoS, DDoS, Web assaults, and network penetration from the inside. The dataset comprises

Table 2 Hardware and environment specification

Unit	Description
CPU	AMD Ryzen 7 5800H
RAM	32 GB
GPU	NVIDIA RTX3600
Operating System	Windows 11
Programming Software	VScode, Pycharm
Packages	Tensorflow 2.4.1, Sklearn 1.0.2, Pandas, Numpy and Matplotlib

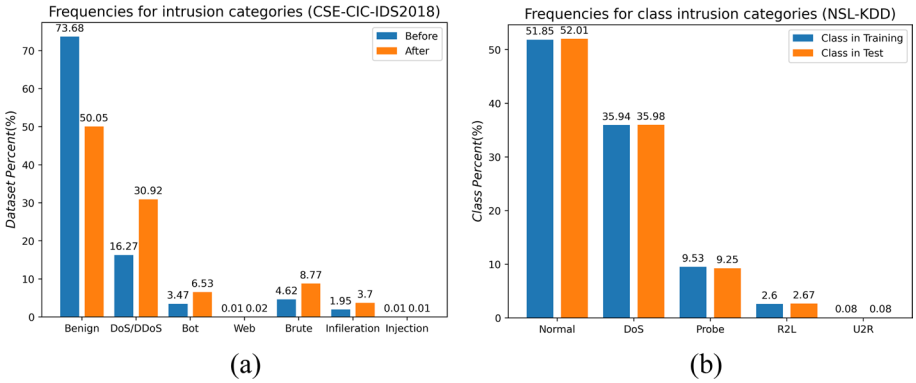

Fig. 7 Frequencies for intrusion categories. **a** CSE-CIC-IDS2018 dataset. **b** NSL-KDD dataset

Table 3 A part of features in the CSE-CIC-IDS2018 dataset

Feature name	Feature short description
Dst Port	Destination port of connection
Protocol	Protocol used during connection
Timestamp	Time that connection occurred
Flow Duration	Duration that connection occurred
Tot Fwd Pkts	Total number of forward packets
Tot Bwd Pkts	Total number of backward packets
TotLen Fwd Pkts	Total length of forward packets
Fwd Pkt Len Max	Maximum length of forward packets
Bwd Pkt Len Mean	Mean size of packet in backward direction
Flow IAT Std	Standard deviation time between two packets sent in the forward direction
Fwd Seg Size Min	Minimum segment size observed in the forward direction
.....	.....
Active Mean	Mean time a flow was active before becoming idle
Idle Std	Standard deviation time a flow was idle before becoming active
Idle Min	Minimum time a flow was idle before becoming active
Label	Describes if file is Attack or Benign

The first column is the name of the feature

The second column is the description corresponding to the feature

50 attacker machines, 420 victim machines, and 30 servers across five departments. It comprises network traffic and system logs from each machine and eighty characteristics extracted from the network traffic captured by CICFlowMeter-V3. Figure 7 illustrates the proportion and distribution of each traffic group (a). There are a disproportionate amount of negative samples in the dataset. Figure 7a illustrates how the Benign class is under-sampled to balance the dataset. Table 3 gives a subset of the dataset's 80 features, with feature names and brief descriptions in the first and second columns, respectively.

NSL-KDD dataset

The conventional KDD99 network traffic dataset has many duplicated data, which might negatively impact the training model's performance. To address the issue mentioned above, Tavallaee et al. proposed the NSL-KDD dataset, which removes redundant

information from the original dataset and adjusts the number of data in the training and testing sets relative to the original, thereby making the KDD dataset more reasonable and capable of enhancing the model's training performance.

The NSL-KDD training set contains 126,620 network traffic samples, with each sample containing 41 features, including four main feature categories, such as essential features (TCP/IP, etc.), event-based traffic features (count, retransmission rate, etc.), content features (Hot, Guest, etc.), and host-based traffic features (Dst host count, Srv count, etc.), as well as some additional features. The characteristics are shown in Table 4. All samples are categorized as either regular traffic or particular categories of attack traffic (e.g., DOS, R2L, Probe, and U2R attacks). Since some data sources in the training and test sets of the NSL-KDD dataset are inconsistent, and some network attacks in the test set do not appear in the training set, which has a significant impact on the performance of the model, we attempt to combine the training and test sets to create a new dataset [51]. Based on this, the new dataset is divided into training and test sets at a ratio of 8:2, and the resulting network assaults on the training and test sets are depicted in Fig. 7b.

Evaluation metrics

Intrusion Detection Systems prioritize precisely identifying attack traffic above the proper classification of routine traffic. In addition to accuracy, several measures like F1-score, precision, recall, and ROC are used to evaluate the performance of a model. The CSE-CIC-IDS2018 and NSL-KDD datasets have an extreme imbalance in sample collection for different attack categories, making accuracy and F1-score more suitable for evaluating the model's performance.

Based on a confusion matrix containing TP, TN, FP, and FN, the performance metrics F1-score, precision, recall, and ROC are determined. The confusion matrix enables us to see the performance of the model as a grid. TP reflects the number of true positives, which shows that attack traffic was accurately predicted. TN reflects the number of true

Table 4 A part of features in the NSL-KDD dataset

Feature name	Feature short description
Duration	Length (number of seconds) of the connection
Protocol_type	Type of the protocol
Service	Network service on the destination
src_bytes	Number of data bytes from source to destination
dst_bytes	Number of data bytes from destination to source
Flag	Normal or error status of the connection
Land	1 if connection is from/to the same host/port; 0 otherwise
.....	.....
Urgent	Number of urgent packets
Num_failed_logins	Number of failed login attempts
Num_access_files	Number of operations on access control files
diff_srv_rate	Percent of connections to different services
Label	Network Traffic Attribution Categories

The first column is the name of the feature

The second column is the description corresponding to the feature

negatives, which implies that regular traffic was accurately forecasted. FP denotes the number of false positives or normal traffic wrongly identified as attack traffic. FN reflects the number of false negatives or attack traffic wrongly classed as normal traffic. FN is the most relevant of these indicators since a low number suggests that the IDS is less likely to misclassify attack traffic. Thus, our technique is to reduce the FN value.

As mentioned above, the basic notation of the metrics is described below.

Accuracy: the number of correct predictions made by the model as a percentage of the total number of predictions. As defined in Eq. (9).

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{9}$$

Precision: this metric measures the percentage of attack traffic correctly predicted as attack traffic and is mathematically defined as follows. As defined in Eq. (10).

$$\text{Precision} = \frac{TP}{TP + FP} \tag{10}$$

F1-Scores: This metric is a combined form of model accuracy and sensitivity and is a reconciled average of model accuracy and sensitivity. In an unbalanced dataset, better F1-Scores indicate fewer misclassified flows, and this metric is the focus of our study. As defined in Eq. (11).

$$F1 = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{11}$$

In the "one vs the rest" scenario, each class or label is compared to all other classes, resulting in a series of binary classifications, one for each class. In contrast, aggregated instances produce a single result that reflects the average or the aggregate of all classes. In addition, the aggregated method incorporates a variety of averaging strategies, such as micro, macro, weighted, and sampling, which yield mixed results. The performance metrics given in this work, including F1-score, accuracy, and recall, were assembled using the weighted technique described by Pedregosa et al. [52].

Receiving operating characteristics curve (ROC): ROC is a combination of response sensitivity and continuous specificity variables that may indicate the link between sensitivity and specificity; the greater the curve area, the better the model's performance. As defined in Eq. (12).

$$AUC_{ROC} = \int_0^1 \frac{TP}{TP + FN} d \frac{FP}{TN + FP} \tag{12}$$

Results

This section first conducts ablation experiments to assess the effects of different modules in the MAFS model. Subsequently, the model's performance is further evaluated on two datasets, CSE-CIC-IDS2018 and NSL-KDD. A comparison is made between the model and other prevalent machine-learning approaches. Firstly, three benchmark models are proposed, and their overall performance is assessed based on criteria such

as accuracy and F1 score, with visualization of PCA results. The overall performance, confusion matrices, ROC curves, etc., of the MAFS model on the two datasets, are presented, along with an evaluation of the impact of Lambda values on model performance. Finally, a comprehensive comparison is made between the model and other machine learning methods and relevant literature.

Ablation study

We conducted ablation experiments to investigate the impacts of the feature self-selection module and the Deep Reinforcement Learning (DRL) module within the MAFS (Multi-Agent Feature Selection) model on intrusion detection. To ensure a fair comparison, we established three baseline models named DT, DQN, and DQN + RFE. These models were employed to explore the contributions of the feature self-selection module and the DRL module within the MAFS framework. We excluded the feature subset generated by the feature self-selection module for the DT model. We replaced the DRL module with a traditional machine learning decision tree (DT) while keeping the other experimental settings unchanged. To assess the performance gain from the DRL module, we introduced a second DQN model. In this model, we omitted the feature self-selection module while retaining the DRL module, maintaining consistency with other experimental settings.

Further assessing the influence of the feature self-selection module, we introduced a third model named DQN + RFE. In this model, we employed the Recursive Feature Elimination (RFE) method instead of the feature self-selection module. This yielded a new feature subset while maintaining other experimental settings. These ablation experiments aimed to unveil the respective contributions of the components within the MAFS model to intrusion detection performance.

The experimental results are presented in Table 5. In the table, "Dataset ID" indicates the performance evaluation for specific datasets, where Dataset ID 1 corresponds to the CSE-CIC-IDS2018 dataset, and Dataset ID 2 corresponds to the NSL-KDD dataset. By comparing the test results of DT and DQN, we observed that DQN outperformed the DT model on both datasets. This suggests that DQN is more capable of accurately identifying network traffic than traditional machine learning methods.

Furthermore, the DQN + RFE model, which incorporates feature selection, outperformed the standalone DQN model on both datasets. This indicates that feature

Table 5 The experimental results for the four methods in terms of accuracy and F1-Score on the CSE-CIC-IDS2018 and NSL-KDD datasets are as follows

Model	Feature Selection	Accuracy	F1-Score	Dataset ID
DT	No	0.931	0.922	1
		0.971	0.969	2
DQN	No	0.941	0.925	1
		0.982	0.979	2
DQN + RFE	Yes	0.962	0.949	1
		0.988	0.986	2
MAFS	Yes	0.968	0.963	1
		0.991	0.991	2

Here, Dataset ID 1 represents the CSE-CIC-IDS2018 dataset, and Dataset ID 2 represents the NSL-KDD dataset

selection approaches have the potential to enhance intrusion detection system performance. Our MAFS model introduces a feature self-selection method on top of the DQN model. In comparison to the DQN + RFE model, MAFS demonstrated improvements in both accuracy and F1-Score, particularly with a more significant improvement in F1-Score. This further validates the effectiveness of our approach in the realm of feature selection.

Additionally, we conducted a visualization evaluation of the top-3 selected features of the DQN + RFE model and the MAFS model, as depicted in Fig. 8. Figure 8a and b display the visualization of the top three feature spaces chosen by the MAFS model on the CSE-CIC-IDS2018 and NSL-KDD datasets, respectively. Figure 8c and d illustrate the visualization of the top three feature spaces chosen by the DQN + RFE model on the same datasets. Analyzing Fig. 8a and c, we observed that two out of the top three features selected by MAFS align with those selected by RFE, namely "Dst Port" and "Int Fwd Win Bytes". However, the third-ranked feature differs. Notably, in Fig. 8a, the visualization of network data appears more dispersed and discernible, indicating that the feature self-selection module of the MAFS model holds stronger feature selection capabilities. As for Fig. 8b and d, both exhibit identical top three feature selections, with the

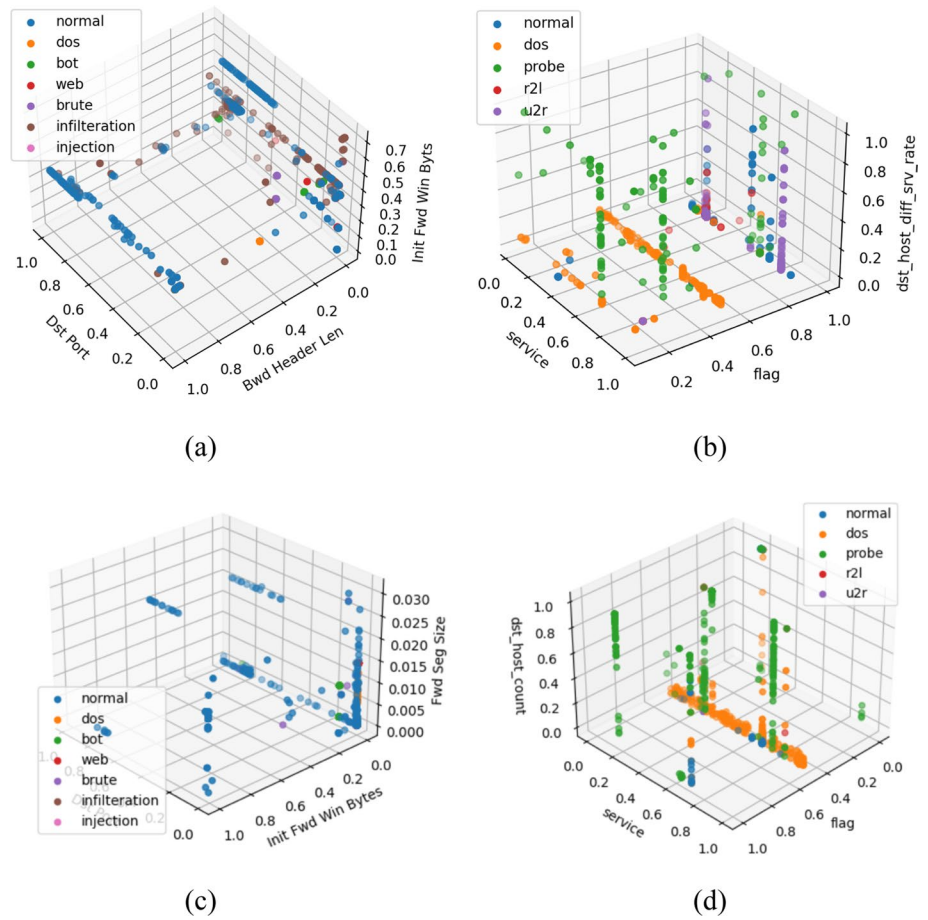

Fig. 8 PCA visualization diagram with top-3 selected features. **a** MAFS with CSE-CIC-IDS2018 dataset. **b** MAFS with NSL-KDD dataset. **c** RFE with CSE-CIC-IDS2018 dataset. **d** RFE with NSL-KDD dataset

Table 6 Top-ranked selected features in the CSE-CIC-IDS2018 dataset

SN	Features	SN	Features
1	Dst Port	10	Fwd Header Len
2	Flow Duration	11	Fwd Pkts/s
3	Fwd Pkt Len Max	12	Bwd Pkts/s
4	Fwd IAT Mean	13	Bwd Header Len
5	Bwd IAT Tot	14	Pkt Len Mean
6	Bwd IAT Mean	15	RST Flag Cnt
7	Bwd IAT Std	16	Init Fwd Win Byts
8	Bwd IAT Max	17	Init Bwd Win Byts
9	Bwd IAT Min	/	/

Table 7 Top-ranked selected features in the NSL-KDD dataset

SN	Features	SN	Features
1	service	6	dst_host_diff_srv_rate
2	flag	7	diff_srv_rate
3	src_bytes	8	dst_host_count
4	dst_bytes	9	dst_host_srv_serror_rate
5	num_root	/	/

feature visualization displaying similarities. This similarity may be attributed to the high accuracy achieved by both MAFS and DQN + RFE models on the NSL-KDD dataset.

In addition, we present the selected feature subsets achieved by MAFS for both the CSE-CIC-IDS2018 dataset and the NSL-KDD dataset. For the CSE-CIC-IDS2018 dataset, MAFS identified 17 features that formed a new feature subset. Similarly, for the NSL-KDD dataset, MAFS selected nine features to create a new feature subset. These results are summarized in Tables 6 and 7. Notably, the implementation of the MAFS model successfully reduces approximately 80% of the redundant features compared to the original full feature set.

MAFSIDS results

Figure 9 illustrates the performance evaluation of the MAFSIDS model on the CSE-CIC-IDS2018 and NSL-KDD datasets. On the CSE-CIC-IDS2018 dataset, the MAFS model achieved an improved Accuracy of 0.968 and F1-Score of 0.963 compared to the DQN + RFE model, particularly with a significant enhancement in F1-Score. This showcases the MAFS model's strong performance in addressing imbalanced datasets. Similarly, on the NSL-KDD dataset, the MAFS model achieved an impressive Accuracy and F1-Score of 0.991 each. These results underscore the superiority of the MAFSIDS model in intrusion detection tasks.

By integrating the feature self-selection module and the deep reinforcement learning module, the MAFS model demonstrated enhanced performance on both datasets. In the case of the CSE-CIC-IDS2018 dataset, the MAFS model's combination of adaptive feature selection and deep reinforcement learning outperformed the DQN + RFE

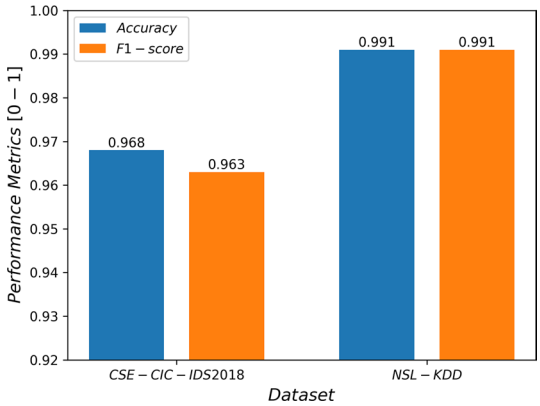

Fig. 9 Performance evaluation of the MAFS model on CSE-CIC-IDS2018 and NSL-KDD datasets

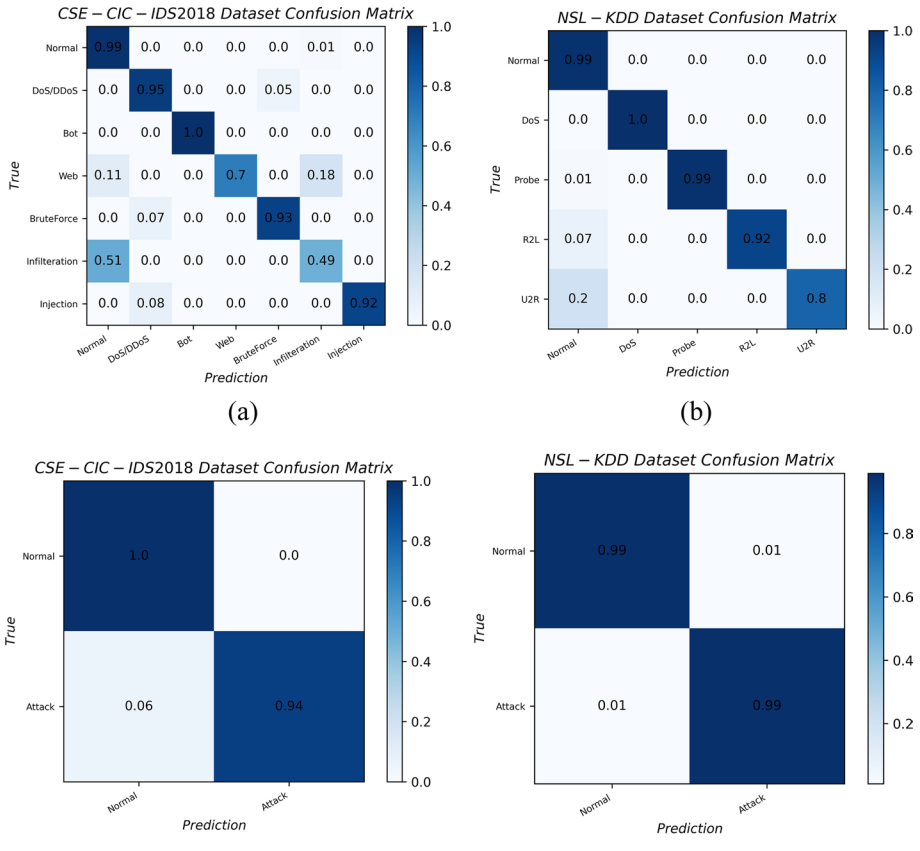

Fig. 10 Confusion matrix diagram of the model in different datasets. **a** Multi-category confusion matrix in CSE-CIC-IDS2018 dataset. **b** Multi-category confusion matrix in NSL-KDD dataset. **c** Binary confusion matrix in NSL-KDD dataset. **d** Binary confusion matrix in NSL-KDD dataset

model. Moreover, for the NSL-KDD dataset, the MAFS model approached performance limits, affirming its robust performance on intricate datasets.

Figure 10 depicts the confusion matrices for multiclassification and binary classification used to evaluate the model's performance on the two datasets. Figure 10a and

c depict the confusion matrices of the model for multiclassification and binary classification on the CSE-CIC-IDS2018 dataset. In contrast, Fig. 10b and d epict the confusion matrices of the model for multiclassification and binary classification on the NSL-KDD dataset, where the predicted values are on the horizontal axis, and the true values are on the vertical axis.

Figure 10a shows that the model can accurately identify the categories Normal, DoS/DDoS, Bot, and Brute Force on the CSE-CIC-IDS2018 dataset. However, it is less effective for the Web and Infilteration categories, and frequently misclassifies the latter as the Normal category, which may be due to the insufficient training samples for this category. Figure 10c illustrates the confusion matrix of the model for binary classification on the CSE-CIC-IDS2018 dataset, where the accuracy of the Normal category is one, and the accuracy of the Attack category is 0.94.

Figure 10b displays the confusion matrix resulting from the model's multi-categorization of the NSL-KDD dataset. Using the NSL-KDD dataset, it can be shown that the model has high identification rates for the Normal, DoS, and Probe categories (all over 0.99) but low recognition rates for the R2L and U2R categories, with the latter having a rate of only 0.8%. This might be due to the minimal quantity of U2R samples in the NSL-KDD dataset, which prevented the model from learning about this type of network threat. Due to the minimal amount of U2R category samples in the NSL-KDD data set, this may be the result of the model's poor learning of this class of network assaults. Figure 10d depicts the confusion matrix results of the model for binary classification on NSL-KDD. It can be seen that the model has high recognition rates for both the Normal and Attack categories (0.99), showing that it can distinguish between normal traffic and attack traffic flawlessly.

In Fig. 11, we displayed the dichotomous ROC curves and computed the AUC area of each curve for each of the four models DT, DQN, DQN + RFE, and MAFSIDS FS on the two datasets, which may typically assist us in understanding the quality of the model predictions. Figure 11a depicts the ROC curves for the CSE-CIC-IDS2018 dataset. The DQN + RFE and OURS models with feature selection techniques have greater AUC areas than the DT and DQN models without feature selection methods, which have AUC areas of only 0.961 and 0.961, respectively. The ROC curves and AUC areas for the

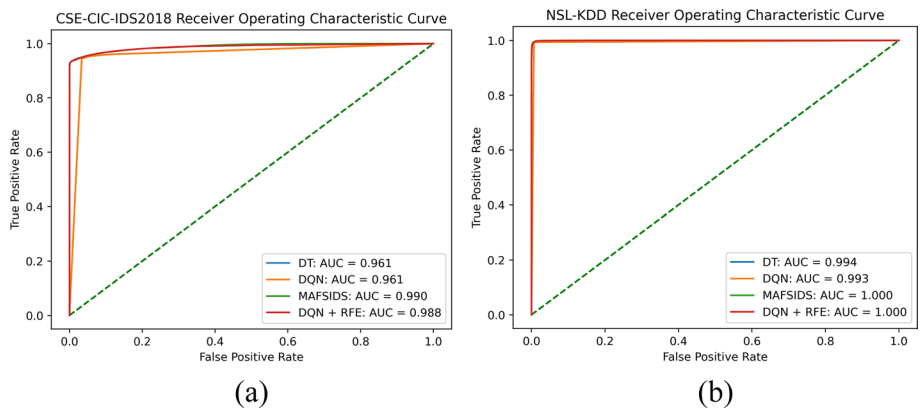

Fig. 11 ROC of the different datasets. AUC is shown on the right. **a** CSE-CIC-IDS2018 dataset. **b** NSL-KDD dataset

four models on the NSL-KDD dataset are all near 1.0, as shown in Fig. 11b. The models have strong detection ability on this dataset, and the ROC curves represent the detection capacity of various models.

Using the two datasets, we drew the binary ROC curves of the four models DT, DQN, DQN + RFE, and MAFSIDS, and computed the AUC area under each curve, as seen in Fig. 11. Figure 11a displays the results of the models on the CSE-CIC-IDS2018 dataset, and it can be shown that the DQN + RFE and MAFSIDS models using the feature selection approach have greater AUC areas than the DT and DQN models without the feature selection method, which are only 0.961. Figure 11b depicts the model results on the NSL-KDD dataset, and it can be seen that all four models have high AUC areas (all near to 1) on this dataset, showing that the models have excellent detection capacity on this dataset. The ROC curves also represent the variations in detecting abilities between models.

To examine the impact of the discount factor in the DRL algorithm on the performance of the model prediction, we utilized three distinct discount factors of 0.01, 0.50, and 0.99 to assess the performance of the model. Figure 12 illustrates the predictive performance of the model for two datasets with various discount factors. Figure 12a illustrates the results of the model on the CSE-CIC-IDS2018 dataset. The model prediction performance decreases gradually as the discount factor increases, and the model has the best prediction performance when the discount factor is 0.01. This may be because the DRL model cannot learn more useful contextual information from the supervised learning dataset, and the data in the supervised learning dataset has low relevance. Figure 12b depicts the model NSL-KDD dataset findings, demonstrating a similar pattern to Fig. 12a.

Comparison

In this part, the performance of the MAFSIDS model is compared to that of other ML models applied to the CSE-CIC-IDS2018 dataset, including MLP, CNN, Logistic Regression, DDQN, DQN + RFE, and SVM ML models. To compare the performance of our proposed model with that of other traditional machine learning techniques on the CSE-CIC-IDS2018 dataset, we give the Accuracy, F1-Score metrics and the test time cost(ms) for detecting a single data for each model in Table 8. We only compare

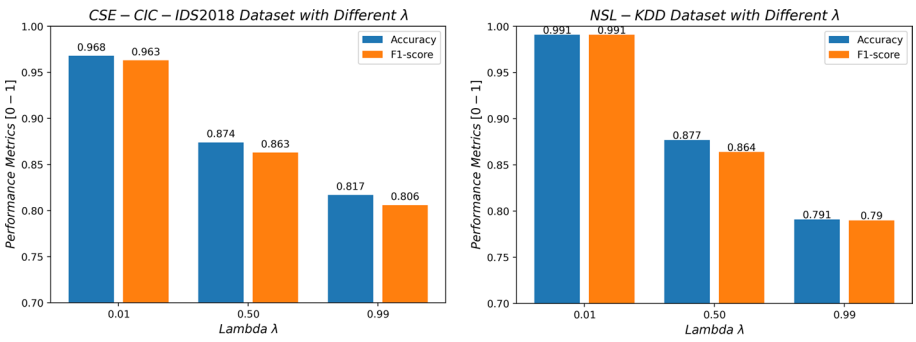

Fig. 12 The impact of different discount factors (λ). **a** CSE-CIC-IDS2018 dataset. **b** NSL-KDD dataset

Table 8 Comparison of performance of multiple models in CSE-CIC-IDS2018 dataset

Detection Model	Accuracy	F1-Score	Test time(ms)
logistic regression	0.881	0.782	6.294
KNN	0.927	0.907	317.929
Random forest	0.836	0.735	26.870
GBM	0.934	0.921	24.717
Gaussian-NB	0.796	0.389	4.738
Bernoulli-NB	0.728	0.589	5.032
Multinomial-NB	0.558	0.498	4.496
AdaBoosts	0.946	0.906	59.217
Neural Network	0.901	0.809	30.525
XGBoost	0.947	0.933	130.918
DT	0.931	0.922	80.301
CNN-1D	0.929	0.918	98.374
DQN	0.941	0.925	110.327
DDQN	0.939	0.928	142.392
Our model	0.968	0.963	34.286

the CSE-CIC-IDS2018 dataset since the NSL-KDD dataset is a regularly used dataset for which several recognition techniques greatly perform.

As shown in Table 8, our suggested model achieves ideal performance in Accuracy and F1-Score, with respective values of 0.968 and 0.963. In comparison, the XGBoost model, a fresh ML model launched in recent years, is behind our suggested model in terms of performance by 2.1% and 3%. There is a 2.2% improvement in F1-Score compared to the DQN model, and test time was reduced significantly. The three subclasses of naive Bayes models, Gaussion-NB, Bernoulli-NB, and Multinomial-NB, perform badly, although the performance of other models is very excellent. In addition to the accuracy of each model's predictions, we also considered its test time for detecting a single data, as indicated in the final column of Table 8. It has been observed that the prediction performance of models with shorter run periods is often inferior, whereas the prediction performance of models with longer run times is considerably superior. The random forest model required the largest amount of time (300 ms), two orders of magnitude longer than the NB series model, which required the least amount of time (about 3 ms). This suggests that runtime and prediction performance are positively correlated. Our suggested model employs a feature selection technique to minimize duplicate features and lower the testing cost to about 34.2 ms for single data.

In addition, we performed a comparative analysis of our model's AUC and F1-Score against other existing approaches using the CSE-CIC-IDS2018 dataset, as presented in Table 9. The results indicate that our model surpasses other methods in terms of performance, exhibiting superior AUC and F1-Score values. Traditional machine learning techniques like Decision Tree and Random Forest demonstrate relatively lower AUC values, although they exhibit higher F1-Scores. Conversely, novel research approaches such as ID-RDRL, which utilize deep reinforcement learning techniques for network attack identification, achieve higher AUC values but slightly lower F1-Scores. These findings suggest that our model excels in handling the challenges posed by imbalanced datasets, making it a more effective solution for intrusion detection.

Table 9 Comparing our model's AUC and F1-Score with other studies in the literature in the CSE-CIC-IDS2018 dataset

Reference	Method	FS Method	AUC	F1-Score
Leevy [53]	CatBoost	/	0.956	0.936
	Decision Tree	/	0.911	0.887
	LightGBM	/	0.951	0.929
	Naive Bayes	/	0.553	0.314
	Random Forest	/	0.955	0.932
	XGBoost	/	0.913	0.889
DKNN [54]	Kronecker neural network	CRDO	/	0.960
MP-CVAE [55]	Variational Autoencoder	/	0.94	0.95
ICVAE-BSM [56]	Variational Autoencoder	BPSO	0.964	0.955
ID-RDRL [22]	Deep Q-Learning	RFE	0.983	0.948
Our model	Deep Q-Learning	MAFS	0.990	0.963

Conclusion

This paper addresses the challenge of automatic feature selection and subspace exploration in intrusion detection systems (IDS). We propose MAFSIDS, a Deep Q-Learning (DQL) based IDS that leverages the MAFS (Multi-Agent Feature Selection) method to enhance feature selection and network attack classification. MAFSIDS introduces a multi-agent reinforcement learning framework to redefine the feature selection problem, reducing the complexity by representing the feature space as N feature agents instead of the traditional 2^N space. To improve the accuracy and speed of feature selection, we employ a Graph Convolutional Neural Network (GCN) to extract deeper features from the data. Additionally, we utilize Mini-Batchs for data recoding, allowing reinforcement learning to be applied to supervised learning, thereby enhancing accuracy. Furthermore, we optimize the model's efficiency by fitting the policy network with a minimalist neural network. In our comprehensive simulation experiments conducted using Python, we evaluate the performance of MAFSIDS on the CSE-CIC-IDS2018 and NSL-KDD datasets. The results demonstrate impressive accuracy (96.8% and 99.1%) and F1-Score (96.3% and 99.1%), with the selected feature subset reducing redundant features by approximately 80% compared to the original features. Additionally, we compare our model with other popular ML models, and our approach exhibits good performance in terms of AUC and F1-Score.

In future research, we plan to extend the application of our model to newer intrusion detection datasets, such as CIC-DDoS2019, to enhance its performance in more specific attack domains. Furthermore, we aim to explore feature transformation techniques to optimize the interpretability and effectiveness of machine learning in IDS, thereby improving overall system performance [57].

Acknowledgements
Not applicable.

Author contributions
KR conceived the initial idea and completed the analysis and calculations. KR wrote the manuscript. BS, YZ revised the manuscript, and YZ did the post-proofreading. Y.Z. supervised the work and provided funding support. All authors reviewed the manuscript.

Funding
This research was supported by the 100 Top Talents Program, SYSU (No. 190158), the Project Supported by National Key Laboratory (No. XM2020XT1009), the Project Supported by National Key Laboratory (No. 6142101190201), the Project Supported by Advanced Research (No. XM2020XT2136).

Availability of data and materials
The dataset investigated for this work is the public dataset CSE-CIC-IDS2018 and NSL-KDD. The former can be downloaded from IDS 2018 | Datasets | Research | Canadian Institute for Cybersecurity | UNB at https://www.unb.ca/cic/datasets/ids-2018.html, and the latter can be downloaded from NSL-KDD | Datasets | Research | Canadian Institute for Cybersecurity | UNB at https://www.unb.ca/cic/datasets/nsl.html. The computer algorithms originated during the current study can be made available from the corresponding author Y.Z. on a reasonable request.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare no competing interests.

Received: 11 March 2023 Accepted: 18 August 2023
Published online: 05 September 2023

References

1. Masdari M, Khezri H. A survey and taxonomy of the fuzzy signature-based intrusion detection systems. Appl Soft Comput. 2020;92: 106301.
2. Singh G, Khare N. A survey of intrusion detection from the perspective of intrusion datasets and machine learning techniques. Int J Comput Appl. 2022;44:659–69.
3. Nugroho EP, Djatna T, Sitanggang IS, Buono A, Hermadi I. A Review of intrusion detection system in IoT with machine learning approach: current and future research. New York: IEEE; 2020.
4. Thakkar A, Lohiya R. A review of the advancement in intrusion detection datasets. Proc Comput Sci. 2020;167:636–45.
5. Denning DE. An intrusion-detection model. IEEE Trans Softw Eng. 1987;SE-13:222–32.
6. Radoglou-Grammatikis P, Rompolos K, Sarigiannidis P, Argyriou V, Lagkas T, Sarigiannidis A, et al. Modeling, detecting, and mitigating threats against industrial healthcare systems: a combined software defined networking and reinforcement learning approach. IEEE Trans Ind Inform. 2022;18:2041–52.
7. Zuech R, Hancock J, Khoshgoftaar TM. Detecting SQL injection web attacks using ensemble learners and data sampling. In: Zuech R, editor. 2021 IEEE international conference on cyber security and resilience. Rhodes: IEEE; 2021. p. 27–34.
8. Liang J, Ma M, Tan X. GaDQN-IDS: a novel self-adaptive IDS for VANETs based on bayesian game theory and deep reinforcement learning. IEEE Trans Intell Transp Syst. 2022;23:12724–37.
9. Dong S, Xia Y, Peng T. Network abnormal traffic detection model based on semi-supervised deep reinforcement learning. IEEE Trans Netw Serv Manag. 2021;18:4197–212.
10. Dey A. Deep IDS: A deep learning approach for intrusion detection based on IDS. In: Dey A, editor. 2020 2nd Int Conf Sustain Technol Ind 40 STI. Dhaka: IEEE; 2018. p. 1–5.
11. Akhtar MS, Feng T. Deep learning-based framework for the detection of cyberattack using feature engineering. Secur Commun Netw. 2021;2021:1.
12. Arulkumaran K, Deisenroth MP, Brundage M, Bharath AA. Deep reinforcement learning: a brief survey. IEEE Signal Process Mag. 2017;34:26–38.
13. Hosseini S, Zade BMH. New hybrid method for attack detection using combination of evolutionary algorithms, SVM, and ANN. Comput Netw. 2020;173: 107168.
14. Kocher G, Kumar G. Machine learning and deep learning methods for intrusion detection systems: recent developments and challenges. Soft Comput. 2021;25:9731–63.
15. Yin Y, Jang-Jaccard J, Xu W, Singh A, Zhu J, Sabrina F, et al. IGRF-RFE: a hybrid feature selection method for MLP-based network intrusion detection on UNSW-NB15 dataset. J Big Data. 2023;10:15.
16. Emmons S, Eysenbach B, Kostrikov I, Levine S. RvS: what is essential for offline RL via supervised learning? Mach Learn. 2022. https://doi.org/10.48550/arXiv.2112.10751.
17. Wan J, Chen H, Yuan Z, Li T, Yang X, Sang B. A novel hybrid feature selection method considering feature interaction in neighborhood rough set. Knowl Based Syst. 2021;227: 107167.
18. Wan J, Chen H, Li T, Sang B, Yuan Z. Feature grouping and selection with graph theory in robust fuzzy rough approximation space. IEEE Trans Fuzzy Syst. 2022. https://doi.org/10.1109/TFUZZ.2023.3250639.
19. Mahmood RAR, Abdi A, Hussin M. Performance evaluation of intrusion detection system using selected features and machine learning classifiers. Baghdad Sci J. 2021;18:884–98.
20. Zhou Y, Cheng G, Jiang S, Dai M. Building an efficient intrusion detection system based on feature selection and ensemble classifier. Comput Netw. 2020;174: 107247.

21. Ren J, Guo J, Qian W, Yuan H, Hao X, Jingjing H. Building an effective intrusion detection system by using hybrid data optimization based on machine learning algorithms. Secur Commun Netw. 2019;2019:1.
22. Ren K, Zeng Y, Cao Z, Zhang Y. ID-RDRL: a deep reinforcement learning-based feature selection intrusion detection model. Sci Rep. 2022;12:15370.
23. Tian Y, Chen G, Song Y, Wan X. Dependency-driven Relation Selection with Attentive Graph Convolutional Networks. In: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, Vol 1 Long Pap. Online: Association for Computational Linguistics. 2021. p. 4458–71.
24. Wang H, Pan S, Ju X, Feng Y. Intrusion detection system based on Global-feature Contribution Network. In: Wang H, editor. IEEE international conference on computer science, electronic information engineering and intelligent control technology (CEI). Fuzhou: IEEE; 2021. p. 258–63.
25. Liu F, Hu Z, Zhang A, Du R, Qin D, Xu J. Multiple classification algorithm based on graph convolutional neural network for intrusion detection. Res Sq. 2021. https://doi.org/10.21203/rs.3.rs-515900/v1.
26. Zhang Y, Yang C, Huang K, Li Y. Intrusion detection of industrial internet-of-things based on reconstructed graph neural networks. IEEE Trans Netw Sci Eng. 2022. https://doi.org/10.1109/TNSE.2022.3184975.
27. Bougueroua N, Mazouzi S, Belaoued M, Seddari N, Derhab A, Bouras A. A survey on multi-agent based collaborative intrusion detection systems. J Artif Intell Soft Comput Res. 2021;11:111–42.
28. Liu K, Fu Y, Wu L, Li X, Aggarwal C, Xiong H. Automated feature selection: a reinforcement learning perspective. IEEE Trans Knowl Data Eng. 2021;35:1–1.
29. Wang Y, Zhao X, Xu T, Wu X. Autofield: automating feature selection in deep recommender systems. Proc ACM Web Conf. 2022;2022:1977–86.
30. Chandrashekar G, Sahin F. A survey on feature selection methods. Comput Electr Eng. 2014;40:16–28.
31. Kipf TN, Welling M. Semi-Supervised Classification with Graph Convolutional Networks. 2022. https://openreview.net/forum?id=SJU4ayYgl. Accessed 14 Jun 2023.
32. Yang Y, Pedersen JO. A comparative study on feature selection in text categorization. Icml. 1997;97:35.
33. Forman G. An extensive empirical study of feature selection metrics for text classification. J Mach Learn Res. 2003;3:1289–305.
34. Yu L, Liu H. Feature selection for high-dimensional data: A fast correlation-based filter solution. In: Proc 20th Int Conf Mach Learn ICML-03. 2003. p. 856–63.
35. Kohavi R, John GH. Wrappers for feature subset selection. Artif Intell. 1997;97:273–324.
36. Narendra PM, Fukunaga K. A branch and bound algorithm for feature subset selection. IEEE Trans Comput. 1977;26:917–22.
37. Schaul T, Quan J, Antonoglou I, Silver D. Prioritized experience replay. ArXiv. 2015. https://doi.org/10.48550/arXiv.1511.05952.
38. Yang J, Honavar V. Feature subset selection using a genetic algorithm. IEEE Intell Syst Their Appl. 1998;13:44–9.
39. Kim Y, Street WN, Menczer F. Feature selection in unsupervised learning via evolutionary search. Proc Sixth ACM SIGKDD Int Conf Knowl Discov Data Min. 2000. p. 365–9.
40. Fortin F-A, De Rainville F-M, Gardner M-AG, Parizeau M, Gagné C. DEAP: evolutionary algorithms made easy. J Mach Learn Res. 2012;13:2171–5.
41. Sugumaran V, Muralidharan V, Ramachandran KI. Feature selection using decision tree and classification through proximal support vector machine for fault diagnostics of roller bearing. Mech Syst Signal Process. 2007;21:930–42.
42. Tibshirani R. Regression shrinkage and selection via the lasso. J R Stat Soc Ser B Methodol. 1996;58:267–88.
43. Deng X, Zhu J, Pei X, Zhang L, Ling Z, Xue K. Flow topology-based graph convolutional network for intrusion detection in label-limited IoT networks. IEEE Trans Netw Serv Manag. 2023;20:684–96.
44. Cheng Q, Wu C, Zhou S. Discovering attack scenarios via intrusion alert correlation using graph convolutional networks. IEEE Commun Lett. 2021;25:1564–7.
45. Zhou X, Liang W, Li W, Yan K, Shimizu S, Wang KI-K. Hierarchical adversarial attacks against graph-neural-network-based IoT network intrusion detection system. IEEE Internet Things J. 2022;9:9310–9.
46. Liu H, Simonyan K, Yang Y. Darts: differentiable architecture search. ArXiv. 2018. https://doi.org/10.48550/arXiv.1806.09055.
47. Lin K, Zhao R, Xu Z, Zhou J. Efficient large-scale fleet management via multi-agent deep reinforcement learning. In: Proc 24th ACM SIGKDD Int Conf Knowl Discov Data Min. 2018. p. 1774–83.
48. Zeynivand A, Javadpour A, Bolouki S, Sangaiah AK, Ja'fari F, Pinto P, et al. Traffic flow control using multi-agent reinforcement learning. J Netw Comput Appl. 2022;207:103497.
49. Fard SMH, Hamzeh A, Hashemi S. Using reinforcement learning to find an optimal set of features. Comput Math Appl. 2013;66:1892–904.
50. Fitni QRS, Ramli K. Implementation of Ensemble Learning and Feature Selection for Performance Improvements in Anomaly-Based Intrusion Detection Systems. In: 2020 IEEE Int Conf Ind 40 Artif Intell Commun Technol IAICT. 2020. p. 118–24.
51. Ma X, Shi W. AESMOTE: adversarial reinforcement learning with SMOTE for anomaly detection. IEEE Trans Netw Sci Eng. 2021;8:943–56.
52. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: machine learning in python. J Mach Learn Res. 2011;12:2825–30.
53. Leevy JL, Hancock J, Zuech R, Khoshgoftaar TM. Detecting cybersecurity attacks across different network features and learners. J Big Data. 2021;8:38.
54. Mayuranathan M, Saravanan SK, Muthusenthil B, Samydurai A. An efficient optimal security system for intrusion detection in cloud computing environment using hybrid deep learning technique. Adv Eng Softw. 2022;173:103236.
55. Li H, Wang Z, Meng H, Zhou Z. Solving the data imbalance problem in network intrusion detection: A MP-CVAE based method. In: 2022 10th Int Workshop Signal Des Its Appl Commun IWSDA. 2022. p. 1–5.

56. Zhang Y, Liu Q. On IoT intrusion detection based on data augmentation for enhancing learning on unbalanced samples. Future Gener Comput Syst. 2022;133:213–27.
57. Xiao M, Wang D, Wu M, Qiao Z, Wang P, Liu K, et al. Traceable Automatic Feature Transformation via Cascading Actor-Critic Agents. Proc 2023 SIAM Int Conf Data Min SDM. SIAM. 2023. p. 775–83

Publisher's Note