

Article

Machine Learning-Based Multilevel Intrusion Detection Approach

Jiasheng Ling, Lei Zhang *, Chenyang Liu, Guoxin Xia and Zhenxiong Zhang

School of Artificial Intelligence and Data Science, Hebei University of Technology, Tianjin 300401, China

* Correspondence: zhanglei@hebut.edu.cn

Abstract: In this paper, we propose a multilevel-based intrusion detection model. Firstly, we design an integrated shared feature technique, which filters the features to create a general dataset, retaining fewer but more significant features to enhance the detection accuracy of the model and reduce computational costs. The first stage employs OC-SVM to achieve the efficient classification of normal and abnormal traffic based on a general dataset. Additionally, the first stage is deployed close to the monitored system to enable low-latency prediction and privacy-preserving operations, thus enhancing flexibility and improving global classification performance. The second stage proposes a novel Edge Attention Network (EGAT) with a Multi-Head Dynamic Mechanism (MHD) framework, which introduces the graph attention mechanism and considers edge information as the only element, assigning greater weights to nodes and edges exhibiting high similarity, emphasizing their relationships and thereby improving the model's accuracy and expressiveness. The MHDEGAT model facilitates additional weight learning by integrating the multi-head attention mechanism with edge features, while the weighted aggregation process enhances the data utilization across different network traffic. Finally, the model is trained and tested using the method of on-network data from a gas industrial control system, with an accuracy of 96.99%, a precision of 97.11%, a recall of 96.99%, and an F1 score of 96.93%, all of which outperform the comparison method.

Keywords: industrial control systems; multilevel-based model; graph attention; multi-head attention



Academic Editor: Ping-Feng Pai

Received: 29 November 2024

Revised: 8 January 2025

Accepted: 14 January 2025

Published: 15 January 2025

Citation: Ling, J.; Zhang, L.; Liu, C.; Xia, G.; Zhang, Z. Machine Learning-Based Multilevel Intrusion Detection Approach. *Electronics* **2025**, *14*, 323. <https://doi.org/10.3390/electronics14020323>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Industrial control systems (ICSs), as the core of critical infrastructure, are widely used in power, water supply, and manufacturing industries. However, with the continuous development of network technology, the cyber-security of ICSs faces serious challenges. Cyber-attacks not only lead to information leakage but may also cause damage to physical equipment, casualties, and environmental damage. Therefore, it is crucial to ensure the security of ICSs [1]. Although Intrusion Detection Systems (IDSs) and firewalls have been widely used, these traditional technologies still fall short in detecting unknown and sophisticated cyber-attacks, and there is an urgent need for more advanced detection methods.

Based on these challenges, many researchers have sought to improve the effectiveness of intrusion detection in ICS networks by leveraging advanced techniques such as deep learning and hybrid models that combine multiple detection techniques. Deep learning approaches, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), have demonstrated significant potential in detecting complex attack patterns by learning hierarchical representations of network traffic. These models, however, typically

rely on a single detection model, which can lead to high resource consumption and increased false alarm rates compared to multilevel or hybrid models. For instance, CNNs effectively capture spatial features from traffic data, but they often require large amounts of labeled data for training, which can be scarce in ICS environments. Similarly, RNNs excel at modeling temporal dependencies but struggle with long-range dependencies and may suffer from slow convergence rates. Li et al. [2,3] proposed a cascading approach to classify network traffic in stages, using a combination of binary classifiers and K-Nearest Neighbor (KNN) models. While this approach helps in simplifying the classification process, it faces limitations when dealing with high-dimensional data and complex attack patterns due to its reliance on traditional classifiers. Pajouh et al. [4] introduced a two-stage feature simplification strategy [5], combining Naive Bayes and KNN algorithms for attack detection. While this method is effective for detecting certain types of attacks, it is less effective in handling diverse attack vectors, particularly those that require adaptive and context-sensitive detection models. Early research conducted by Staniford Chen et al. [6] proposed the application of graph theory [7] to network behavior modeling, which led to the exploration of graph neural networks (GNNs) in intrusion detection.

In recent years, graph neural networks (GNNs) [8] have gained attention for their ability to model the complex relationships between entities in ICS networks. Zhou et al. demonstrated their potential by converting network traffic into graph structures, allowing GNNs to recognize intricate attack patterns that are difficult to detect using traditional methods. While GNNs offer promising capabilities in capturing complex attack scenarios, existing studies still face challenges in fully exploiting graph edge information and achieving efficient multilevel detection. However, most graph-based detection models are node-centric, heavily relying on node features while failing to fully utilize edge features. Additionally, GNNs often struggle with scalability in large-scale ICS environments and may encounter issues handling dynamic and evolving attack patterns in real-time systems. To cope with the high resource consumption of a single model as well as the high false alarm rate, existing graph attention networks applied to ICSs only focus on the information of graph nodes while ignoring the important information of edges, and to cope with the lack of flexibility in the allocation of weights, this paper proposes a multilevel-based intrusion detection model named GRL-HEGAT. The first layer integrates feature extraction through the GRL module and eliminates redundant features by using cumulative variance to overcome the complexity characteristics brought by high-dimensional data and realize lightweight and efficient intrusion monitoring. The second layer designs the MHDEGAT multi-classification intrusion detection model, which enhances the focus on graph structure and edge information to capture attack patterns in complex networks more flexibly and realize efficient layered intrusion detection.

The contributions of this paper are mainly categorized into the following three points:

1. This paper proposes a hierarchical detection model named GRL-HEGAT, which is primarily used for identifying intrusion traffic monitoring in ICSs.
2. We designed the GRL module to implement integrated feature extraction. We used the cumulative variance of the feature values returned by each technique to eliminate less contributing features, thereby achieving lightweight and efficient monitoring results.
3. We propose a second-layer multi-classification intrusion detection model for MHDEGAT, which improves the focus on graph edge information, enabling the network to capture complex graph structure features more flexibly, and effectively improves the accuracy of the model.

2. Materials and Methods

2.1. Feature Engineering

Industrial control system datasets usually contain a lot of feature information; however, not all feature information is useful for intrusion detection classification, and a good feature selection [9,10] step can remove redundant and useless feature information and thus improve the performance of intrusion detection models. The goal of feature selection [11] is to obtain a set of important features from the entire dataset, and these selected features are important for training the classification model [12]. Here, an integrated common feature technique for lightweight intrusion detection in industrial control systems is proposed, which combines three feature selection techniques, namely, the Gini index, recursive feature elimination, and L1 regularization, to improve the performance and computational efficiency of the model by filtering out the shared key features [13] and then constructing a general feature dataset with fewer but more representative features; this achieves high-efficiency classification with fewer features while improving the performance and computational efficiency of the model by integrating multiple feature selection methods to provide a higher classification rate [14].

(1) Gini index:

The Gini index was introduced in 1912 by the Italian statistician and sociologist Corrado Gini and was originally used to measure the degree of statistical dispersion in the distribution of income across different sectors [15] of the population. It is a way of performing feature selection similar to information entropy, usually used in classification and regression tree algorithms, and can be used to indicate the impurity of data. Nowadays, the Gini index is widely used in text mining in data mining research. Given a dataset D , which contains samples from l categories, $C_1, \dots, C_l$, let f be a feature with k different values in the dataset. Depending on the value of f , the dataset D can be partitioned into k non-overlapping subsets $D_1, \dots, D_k$, and the scores of the feature f can be computed using the Gini index, which is defined as follows:

$$G(f) = \sum_{i=1}^k \frac{|D_i|}{|D|} \left(1 - \sum_{j=1}^l p_{i,j}^2 \right) \quad (1)$$

where $p_{i,j}$ is the probability that an instance in D_i belongs to category C_j .

The value of $G(f)$ is between 0 and 1, where the lower the value of $G(f)$, the more important the feature f is. For any subset, if all the samples in the subset belong to the same class, then $p_{i,j}^2$ is equal to 1, whereas it is 0 when feature f has the strongest differentiation.

(2) Recursive feature elimination:

Recursive feature elimination [16] is an effective feature selection method commonly used in machine learning to improve the predictive accuracy of a model or to increase the training efficiency of a model. The basic idea of the RFE method is to iteratively construct the model and remove the least important features at each step until the desired number of features is reached.

(3) L1 regularization:

The Least Absolute Shrinkage and Selection Operator is a widely used method in statistics and machine learning, especially for enhancing the interpretability of models and dealing with feature selection problems in high-dimensional datasets. L1 regularization effectively drives many feature weights to exactly zero by incorporating a penalty term, proportional to the absolute value of the feature weights, into the model's loss function. This approach not only reduces many feature weights to zero but also enhances the model's

sparsity and interpretability. This property allows L1 regularization to perform not only feature compression but also automatic feature selection, resulting in a sparse model. L1 Regularized Loss Function:

$$\min_w \left[\sum_{i=1}^N \left(w^T x_i - y_i \right)^2 + \lambda \|w\|_1 \right] \quad (2)$$

where w denotes the weight vector, x_i is the feature vector of the i th sample, and y is the target vector i of the sample. λ is the regularization strength, different values of λ will bring different effects, and a smaller value of λ will bring a weaker penalty effect, making the model more complex. Meanwhile, a larger value of λ will bring a larger penalty, making the model sparser and feature selection more significant. $\|w\|_1$ is the L1 paradigm of the weight vector.

2.2. One-Class Support Vector Machine

For intrusion detection problems, where the size of abnormal samples is disproportionately large, using traditional SVM classification methods, the performance is often unsatisfactory, with a large number of false alarms or omissions. OC-SVM is particularly suitable for anomaly detection because it learns the distribution of normal samples and establishes a decision boundary that separates normal samples from abnormal samples.

The ordinary SVM algorithm [17] aims to find a hyperplane by solving an optimization problem that separates normal and abnormal data points, maximizing the geometric margin. It maps input data points to a higher-dimensional feature space using kernel functions, which may have infinite dimensions. This transformation allows originally non-linearly separable patterns to become linearly separable in the new space. Similarly, the OC-SVM algorithm seeks to determine the decision boundary with the largest geometric margin. As an unsupervised learning approach, OC-SVM does not require labeled training data and is particularly effective for anomaly detection. It assumes that anomalous data points are near the origin while normal data points are farther away. Unlike traditional SVM, OC-SVM uses spherical decision boundaries in the high-dimensional feature space, with the objective of finding the smallest sphere that encompasses the largest number of normal data points. First, a kernel function is used to map the data into the high-dimensional feature space to better separate normal from abnormal samples. Commonly used kernel functions include linear kernels, Gaussian kernels, and polynomial kernels. In this paper, we use a Gaussian kernel. The Gaussian kernel is particularly effective in handling non-linear relationships between features, which is often the case in real-world intrusion detection scenarios where the data are not linearly separable. As for the selection of parameters, such as the kernel parameter (σ) and the regularization parameter (ν), these are determined through a combination of cross-validation and grid search. The size of the slack variables is tuned through cross-validation and grid search methods. The ν parameter controls the model's tolerance to anomalous samples, with larger values of ν allowing for more misclassified samples, thus increasing the size of the slack variables and ensuring that the model can adapt to more complex network traffic anomalies. It is ensured that the choice of slack variables improves the robustness of the model without leading to overfitting, thus improving the performance of the model in real intrusion detection tasks. We tested multiple values for the parameters, evaluating the model's performance in terms of detection accuracy, false positive rate, and computational efficiency. The final choice of parameters represents the optimal trade-off between detection precision and recall while also ensuring that the model does not overfit the training data. Then, OC-SVM finds a hyperplane by optimizing an objective function that contains most normal samples within

its boundaries. A slack variable is used in the optimization process to allow a small number of normal samples to be misclassified as abnormal, thus improving the robustness of the model. After the final training is complete, the model can be used to detect whether a new data point is an abnormal sample. A new data point is considered an abnormal sample if its position in the high-dimensional feature space falls outside the hyperplane.

2.3. Edge Attention Graph Neural Networks

Edge Attention Graph Neural Networks (EGATs) [18] are a specialized method for processing graph data with edge features, which enhances the focus on edge information on the basis of the original attention. Unlike the previous focus on graph nodes only, EGATs take the edge information of the graph as the main element for calculating the attention weight. A set of node features and a set of edge features are input to EGATs, and the degree of association between nodes is determined by calculating the similarity between node features and edge features. This approach better captures the complex relationships in the graph structure and takes into account the proximity between nodes in the feature learning process (Figure 1).

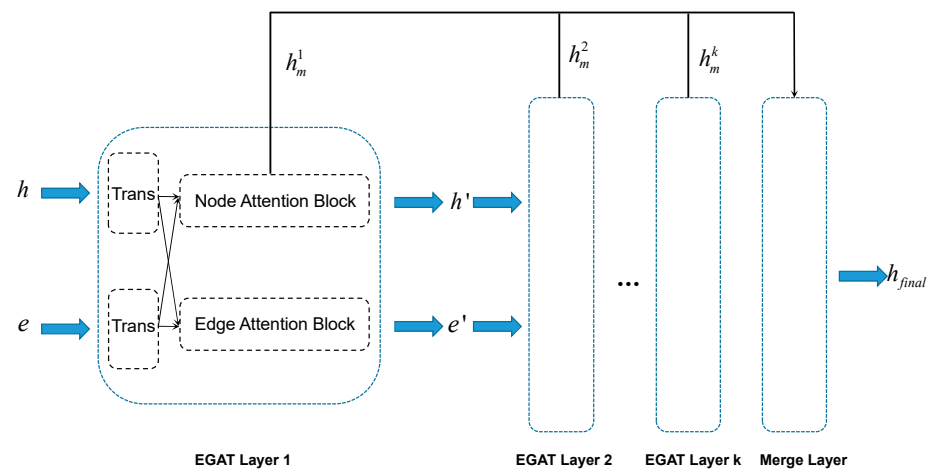


Figure 1. EGAT structure.

The EGAT model consists of two main parts, the node attention module and the edge attention module. In the node attention module, the similarity between each node in the graph and its neighboring nodes is calculated to measure the strength of the relationship between the nodes. The attention coefficients of neighboring nodes are calculated by combining the attention mechanism of edge features as follows:

$$\alpha_{i,j} = \frac{\exp(\text{LeakyReLU}(\vec{a}^T [\vec{h}_i \parallel \vec{h}_j \parallel \vec{e}_{ik}]))}{\sum_{k \in N_i} \exp(\text{LeakyReLU}(\vec{a}^T [\vec{h}_i \parallel \vec{h}_k \parallel \vec{e}_{ik}]))} \quad (3)$$

where $\alpha_{i,j}$ is the attention coefficient between nodes i and j ; $\vec{h}_i$ and $\vec{h}_j$ are the features of nodes i and j , respectively; $\vec{e}_{ij}$ are the edge features between nodes i and j ; $\vec{a}$ is the weight vector used to compute the attention coefficient; and $\vec{e}_{ik}$ denotes the edge features between node i and its neighboring node k .

After the acquired attention weights [19] of each node are normalized, the model applies these weights to the features of the neighboring nodes, performing a weighted summation operation. The result generated by this process is the output of the node attention module, which is expressed as the updated features of the node. These updated

node features can be expressed in the following way: the new features of a node are a weighted combination of the features of its neighboring nodes, and the weights are determined by the similarity between the nodes processed by the softmax function, which ensures the effective aggregation of information and maintains the diversity of features. The updated node features can be expressed as

$$\vec{h}'_i = \sigma\left(\sum_{j \in N_i} \alpha_{i,j} \vec{h}_j\right) \quad (4)$$

In the edge attention [20] module, the model calculates the similarity between each edge and its neighboring edges to measure the association between edges. By integrating the node features and edge features [21], the attention coefficients of neighboring edges are calculated as follows:

$$\beta_{p,q} = \frac{\exp(\text{LeakyReLU}(\mathbf{b} [\vec{e}_p \| \vec{e}_q \| \vec{h}_{pq}]))}{\sum_{k \in \mathcal{N}_p} \exp(\text{LeakyReLU}(\mathbf{b} [\vec{e}_p \| \vec{e}_k \| \vec{h}_{pk}]))} \quad (5)$$

where $\beta_{p,q}$ is the attention coefficient between edge p and edge q ; $\vec{e}_p$ and $\vec{e}_q$ are the features of edge p and edge q , respectively; and $\vec{h}_{pq}$ is the feature connecting edge p and edge q .

The new edge features are represented as follows:

$$\vec{e}'_p = \sigma\left(\sum_{q \in \mathcal{N}_p} \beta_{p,q} \vec{e}_q\right) \quad (6)$$

2.4. Multi-Head Dynamics

The multi-head dynamic attention mechanism allows the model to process information in parallel across multiple “heads”, enabling the model to learn features of the input data from different representation subspaces concurrently. Each head independently computes a set of attention coefficients, and the dynamic weight layer adjusts the contribution of each head, providing flexibility in capturing diverse information. The introduction of the dynamic weight layer dynamically adjusts the contribution rates of each head based on the input features, enhancing the model’s adaptability and flexibility. The specific implementation is as follows:

$$D_i = \text{sigmoid}(W_d \cdot h_i + b_d) \quad (7)$$

where W_d is the weight matrix, b_d is the bias term, and D_i is the dynamic weight vector computed from the node features, with each element corresponding to the weight of a head. The features computed independently for each head are first stacked and then weighted and averaged using the dynamic weights to combine information from all heads. The specific implementation is as follows:

$$h'_i = \frac{\sum_{k=1}^K D_{i,k} \cdot h_{i,k}}{\sum_{k=1}^K D_{i,k}} \quad (8)$$

where $h_{i,k}$ denotes the feature of the node computed by the k th head, and $D_{i,k}$ is the k th element in D_i , which denotes the weight of the k th head on node i .

The multi-head dynamic attention mechanism provides a powerful framework for enhancing the capabilities of graph neural networks by enabling the parallel processing of information from multiple perspectives while dynamically adjusting the contribution of

each attention head. This mechanism improves the model's ability to capture complex and diverse graph structural features with greater flexibility. Furthermore, the introduction of the dynamic weight layer facilitates the adaptive aggregation of the outputs from different attention heads, optimizing the fusion of information and strengthening the model's feature representation.

2.5. Multilayered Deployment

To address the challenges of high computational costs, bandwidth demands, and privacy risks in existing intrusion detection systems that rely on a single machine learning model, we propose a hierarchical deployment approach. This method divides the detection process into multiple stages, allowing independent or combined deployment at each layer. By implementing this multilayered architecture, the system reduces bandwidth usage and latency while supporting diverse anomaly detectors trained on local data to model normal behavior within specific sub-networks. This enhances both flexibility and scalability.

In the first stage, located at the edge of the monitored industrial control system network, only suspicious events are forwarded to higher layers, allowing normal traffic to remain local and preserving privacy. The hierarchical design optimizes network resource utilization, lowers data transmission latency, and strengthens data security. Additionally, it supports flexible deployment adjustments for various network environments. In high-traffic or time-sensitive contexts, more computational tasks are handled by edge devices, reducing bandwidth and latency, while the centralized layer processes complex analyses and model training. This strategy not only boosts system efficiency but also strengthens intrusion detection capabilities in large-scale industrial networks, enabling low-latency, privacy-preserving operations across diverse threats.

3. The Proposed Methodology

3.1. Algorithm Overview

In this paper, we propose a hierarchical intrusion detection method combining the GRL feature extraction method, OC-SVM, and an enhanced graph neural network model. Building on the previous EGAT model, we introduce improvements by incorporating a multi-head dynamic attention mechanism, which strengthens the focus on edge information and improves the representation of relationships between network traffic nodes. The multi-head dynamic attention mechanism assigns different weights to various parts of the graph, enabling the model to focus more effectively on the most relevant features, thus leading to the development of the MHDEGAT model. The GRL feature extraction method can eliminate features with a small contribution rate and generate feature-less shared feature datasets, and lightweight malicious sample filtering is achieved using OC-SVM. Finally, the MHDEGAT module performs multi-class classification based on graph neural networks, accurately identifying various types of attacks.

3.2. Model Structure

In this paper, we propose a multilevel intrusion detection model. As shown in Figure 2, the model roughly consists of the GRL module, the OC-SVM module, and the MHEGAT module. The GRL module generates a common feature dataset by integrating three feature selection methods (Gini index, RFE, and L1 regularization) to filter out the features that impact the model most. Subsequently, the OC-SVM module identifies potential anomalies using the Gaussian kernel function, classifies the data points by anomaly scores, and passes the data with high anomaly scores to the next stage. Finally, the MHEGAT module performs graph neural network-based multi-class classification of these potential anomalies to identify various types of attacks accurately.

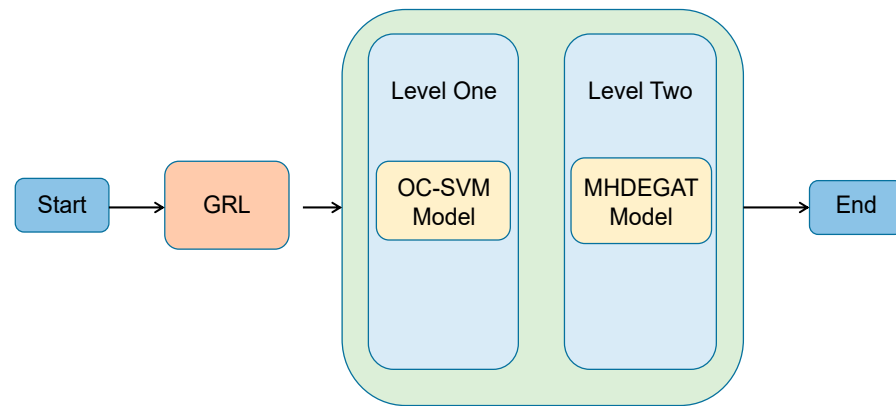


Figure 2. Model structure diagram.

The structure of the proposed GRL-HEAGT multilayer intrusion detection model is shown in Figure 3, and the operation flow of the model is described in detail next.

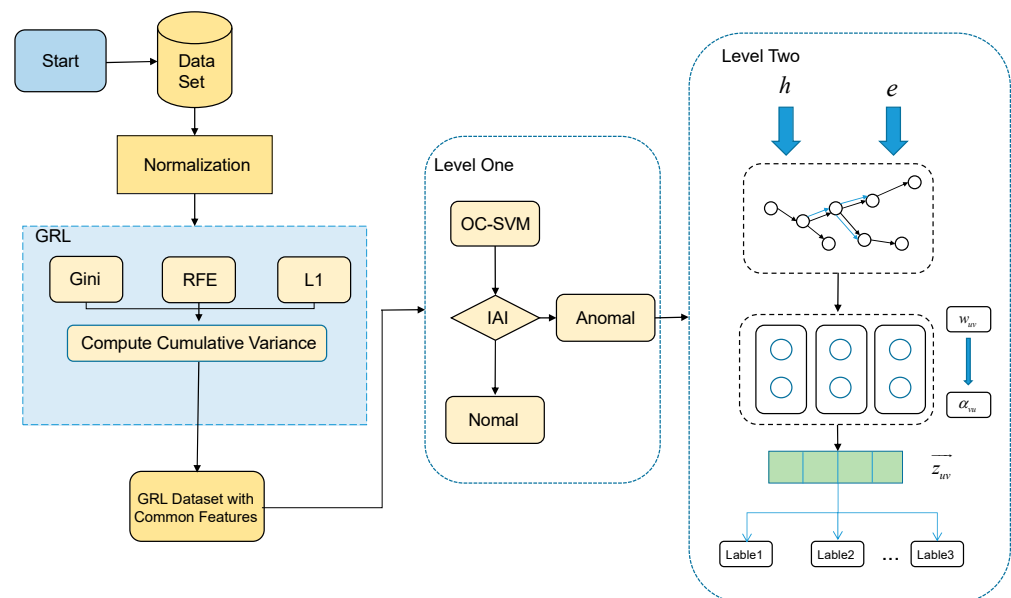


Figure 3. Algorithm flowchart.

(1) GRL Feature Selection: The GRL module integrates three feature selection methods to implement an integrated common feature technique [22] by eliminating features that contribute less and thus generating a common feature dataset. Among them, Gini uses the Gini index to assess the importance of features [23] by calculating the impurity decrease that each feature brings in the classification. Initially, the Gini index filters out features with a significant impact on classification performance. RFE ranks the features according to the absolute value of the coefficients in the logistic regression model. Features with larger coefficients have a greater impact on the model's output. Unimportant features are then eliminated. In contrast, the L1 linear regression method, which imposes a penalty by summing the absolute values of the regression coefficients, retains the most influential features and removes those that do not contribute much to the model's predictions. After applying each technique to the dataset, feature importance is assessed and ranked. The features are then evaluated to determine which are the most important. For the ranked list of features, the variance contribution of each feature is accrued starting from the most crucial feature until the cumulative variance [24] reaches a preset threshold, which is typically set to 95% of the total variance. This threshold was chosen because it ensures that the

model retains most of the explanatory power while eliminating less contributing features. At 95%, the features included contribute the majority of the variance, while redundant or less significant features, which do not significantly impact the model's performance, are removed. After the cumulative variance increases to a certain point, adding new features no longer significantly increases the amount of variance explained, at which point the process of adding features can be stopped. These added features are considered important, while features that are not added are considered redundant or less contributing and can be removed. The 95% cumulative variance threshold was selected because it strikes a balance between reducing the dimensionality of the feature set and maintaining the explanatory power of the model, which is generally sufficient for most practical applications. Finally, by comparing the features selected by the three methods, the features that are considered important in all three methods are identified, and the intersection of the three feature lists is taken to form the common feature dataset. The specific feature engineering process is shown in Algorithm 1.

Algorithm 1 For Feature Engineering.

```

1: Import the dataset.
2: For the dataset  $T$ :
3:   Sort and rank the features  $(v_i, v_{i+1}, \dots, v_n)$ .
4:   Arrange  $i \leftarrow a_i, a_{i+1}, \dots, a_n$  in descending order of their magnitudes.
5:   Calculate  $Cum_{var} \leftarrow Var(a_{i+1}, \dots, a_n)$ .
6:   Exit if  $Cum_{var} + a_i \neq Cum_{var}$ .
7:   Save  $A_i \leftarrow v_i$  as the set of selected features based on  $Cum_{var}$ .
8: Repeat for  $k \leftarrow 1$  to  $N$ :
  • Perform steps 2 to 7 with a new feature selector  $F_{selector}$ .
9: If  $x \in (A_i \cap A_{ii} \cap A_{iii})$ 
10:   Select  $x_i, \dots, x_n$ .
11:   Save  $Y \leftarrow x_i$ .
12:   Repeat steps 9 to 11.
13:   End if.
14:   Return  $Y$ .
```

In this study, the original dataset contains 68 features, which encompass multidimensional information on network traffic. However, some features are redundant, and the model suffers from low computational efficiency and high memory usage. After applying the GRL method, a common feature set is obtained, with the feature dimensionality reduced to 17. This final feature set retains only the most critical features, ensuring that it consists of those with the highest information content for intrusion detection. The retained information after filtering includes features such as the source IP address, destination IP address, source port number, destination port number, packet count, byte count, and protocol type. These features are directly associated with the core behavioral patterns of network traffic and serve as crucial indicators for distinguishing between normal and abnormal traffic. This ensures that the model achieves high accuracy and precision when detecting anomalous behavior, effectively supporting the detection of network intrusions in practical applications.

(2) Level One: Next, the utility feature data are sent to the first-level OC-SVM module of the model. In our proposed hierarchical intrusion detection system, the first stage is the critical initial screening session, which aims to identify potential anomalous activities quickly and efficiently. To enhance the performance of this phase, we employ the OC-SVM algorithm instead of the traditional one-class classification method. In the specific

implementation, we randomly select samples and features from a generalized dataset and train the OC-SVM model. OC-SVM identifies data points that are significantly different from normal data by learning the distribution of the normal data, and maps the input data to a high-dimensional space by using a Gaussian kernel function. In the high-dimensional space, the OC-SVM constructs a boundary to include most of the normal data points while excluding the abnormal data points. The anomaly score of a data point is calculated based on its distance from the hyperplane. Data points that are farther away from the hyperplane have higher anomaly scores, indicating a higher probability of abnormality. The threshold for anomaly score determination is critical for effective anomaly detection. In our model, the threshold is set based on the distribution of anomaly scores in the training set. During the training phase, we compute the anomaly scores for all the normal data points and analyze the distribution of these scores. Typically, the mean and standard deviation of the anomaly scores for the normal data are computed. The threshold is then set as a value greater than the mean by a multiple of the standard deviation, such as $\text{mean} + 3 \times \text{standard deviation}$. This ensures that the majority of normal data points fall below the threshold, while data points that are far from the hyperplane (i.e., the most anomalous) exceed the threshold and are classified as potential anomalies. The threshold value can also be fine-tuned through cross-validation, depending on the trade-off between precision and recall, to optimize the system's ability to detect anomalies without generating too many false positives. A data point is only marked as a potential anomaly and passed to the next stage for in-depth analysis if its anomaly score exceeds this threshold.

(3) Level Two: In our hierarchical intrusion detection system, the purpose of the second stage is to perform further detailed analysis of potential anomalous events passed from the first stage. This stage employs a multi-class classification model based on the graph neural network, which aims to identify and classify various known attack types accurately. In the second layer, edge information plays a pivotal role. Specifically, edge features represent the relationships or interactions between the nodes in the graph, capturing the flow of information or communication between different entities. These edges are not simply connectors between nodes but carry crucial contextual data that defines how nodes interact with each other in the system. For example, in the context of network traffic, edges can represent communication paths, and the edge features could include metrics such as traffic volume, communication frequency, and protocol types. These edge features provide important insights into the patterns and behaviors of network communications, which are essential for accurately detecting complex attack scenarios. In the context of this research, key information from each flow is extracted from large datasets, including source IP addresses, destination IP addresses, source port numbers, destination port numbers, packet counts, byte counts, and protocol types. Among these, the source IP address and destination IP address are treated as the nodes in the graph, while features such as TCP flags, packet counts, byte counts, and protocol types are treated as edge features. This distinction is critical because node features represent the entities within the network, while edge features capture the interactions between these entities. MHDEGAT first applies the attention mechanism to edges in order to capture and utilize the importance of the edges more accurately. During propagation, the edge data are considered the key element, and the node data are used as a staging area for its neighboring edge data, thus avoiding the node feature aggregation, which brings about the computational cost increase. The input to MHDEGAT is graph information, which includes both node information (such as source IP address and destination IP address) and edge information (such as packet counts and protocol types). Once the graph is constructed, the next step is to train the model with the provided dataset. Node and edge features are initialized separately, with node features representing the entities (e.g., devices or systems) and edge features

representing the interactions between these entities (e.g., communication paths). These features are then transformed into higher-dimensional spaces through learnable weight matrices, allowing the model to better capture complex relationships between nodes and edges. The node features in the graph are updated through a node aggregation process where nodes aggregate information from their neighboring nodes to generate new edge embeddings. These updated edge embeddings are utilized for final edge classification. The dynamic weight layer adjusts the contribution rates of each attention head during training. The contribution of each head is dynamically computed based on the importance of the corresponding edges, with more attention being given to edges that show higher relevance for detecting specific attack patterns. This dynamic adjustment is made possible by learning the attention weights through backpropagation and using them to fine-tune the model's focus during different stages of training. Throughout training, the model minimizes the loss function [25] to reduce the gap between predicted and true values. The final edge embeddings are used for classification, where the model assigns probabilities to each category through a softmax layer, outputting the likelihood of each attack type.

4. Experiment

4.1. Experimental Settings

The experiments in this paper were conducted using Python 3.9 and implemented with PyTorch as the deep learning framework, with Windows 11 as the operating system, an NVIDIA GeForce RTX 3070 as the graphics card, and an Intel i5-12400F as the processor.

In the experiments, the parameters of the model were set as follows: we used 512 as BatchSize, LearningRate was set to 0.0001, WeightDecay was 0.00001, the number of attention heads in the GAT layer was 3, and the output feature dimensions of each head was 16. EmbeddingSize was set to 32, ReLU was used for the activation function, the k value was set to 3 in order to effectively sample neighboring nodes, the whole model was trained on 60,000 epochs, and Adam was used for the optimizer.

The total training time for the model was approximately 4 h, with a memory consumption during training of around 8 GB of GPU memory. The exact training time and memory usage may vary depending on the complexity of the dataset and system performance.

4.2. Evaluation Standard

Since the intrusion detection system is a multi-class classification task, we used generalized evaluation metrics for multi-class problems to assess the model's performance, including accuracy, precision, and recall. To provide a more comprehensive, objective, and accurate evaluation, we also introduced the F1 score as a classification metric. The accuracy rate indicates the ratio of the number of samples correctly predicted by the classifier to all samples:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (9)$$

Recall represents the proportion of true cases correctly identified as positive by the classifier relative to the total number of actual positive cases:

$$Recall = \frac{TP}{FN + TP} \quad (10)$$

Precision is the ratio of correctly predicted positive samples to the total number of samples classified as positive by the model:

$$Precision = \frac{TP}{TP + FP} \quad (11)$$

The *F1* score is the harmonic mean of precision and recall:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (12)$$

False positive rate (FPR) is the proportion of actual negative samples misclassified as positive:

$$FPR = \frac{FP}{FP + TN} \quad (13)$$

where *TP* indicates that the model predicts positive samples as positive samples, *TN* indicates that the model predicts negative samples as negative samples, *FP* indicates that the model predicts negative samples as positive samples, and *FN* indicates that the model predicts positive samples as negative samples.

4.3. Experimental Datasets and Preprocessing

The dataset in this study is from a factory control system's network traffic, captured using Wireshark. It includes 102,272 packets: 57,693 normal and 44,624 attack packets (brute-force, DDoS, XSS, and secondary detection attacks). Six categories, including normal traffic, are labeled from 0 to 5. The dataset contains a variety of attributes, such as source IP address, destination IP address, source port, destination port, protocol type, packet count, byte count, and attack type. The specific description of the dataset is shown in Table 1.

Table 1. Dataset description.

Name	Amount	Percentages	Label
Normal	57,693	62.52%	0
Brute Force	10,008	10.85%	1
Attack A	4472	4.85%	2
Attack B	512	0.55%	3
XSS	19,632	21.27%	4
DDoS	10,000	10.84%	5

The attack packets are classified into different categories:

- Brute-force attacks (labeled as 1) involve repeated login attempts to gain unauthorized access to the system.
- DDoS attacks (labeled as 5) are characterized by large volumes of traffic aimed at overwhelming the system.
- XSS attacks (labeled as 4) exploit vulnerabilities in web applications to inject malicious scripts.
- Attack A/B (labeled as 2 and 3) represent other types of attacks detected by the system.

Since the model in this paper uses multi-stage processing and the anomaly detection method based on the graph neural network is used in the second stage, the preprocessing for the data is different before the first stage and before the second stage.

In the first stage, data preprocessing involves label encoding, a method that transforms categorical variables into unique integer labels ranging from 0 to $n - 1$, and Z-score standardization, which normalizes data by adjusting values based on the mean and standard deviation to ensure consistency and comparability.

In the second phase, data preprocessing is useful for converting the filtered data from the previous phase into graphical data suitable for training and testing. In the second stage, data preprocessing helps transform the filtered data from the previous stage into graph data suitable for training and testing. First, key information is extracted from a large volume of NetFlow data for each flow, including source and destination IP addresses, source and

destination port numbers, TCP flags, total byte count, and protocol types. Among these, the source IP address and destination IP address are treated as nodes in the graph, while features such as TCP flags, packet count, byte count, and protocol types are treated as edge features. Node features represent the entities within the network, while edge features capture the interactions between these entities. This representation allows network traffic to be modeled as a graph, with node and edge features providing rich contextual information, which is then passed as input to the MHDEGAT model for further processing.

4.4. Experimental Results and Analysis

Our first layer was a lightweight classification model. In order to prove the effectiveness and efficiency of our first stage, we first used the first layer of the modeling method in the generalized feature dataset after GRL integrated feature [26] processing for testing, comparing with the use of the original dataset. Also, metrics such as the overall accuracy, computation time, and memory consumption were recorded for each experiment, and, at the same time, we compared the first layer of the model with some other standard machine learning classification models, such as iForest [27] and Auto-Encoder [28]. In order to ensure a fair comparison between all models, the parameters for iForest and Auto-Encoder were optimized using grid search with cross-validation. This approach allows for a systematic exploration of the hyperparameter space, ensuring that each model is tested with its optimal settings. Specifically, for iForest [29], we optimized parameters such as the number of trees and the sub-sampling size, and for Auto-Encoder, we optimized the number of hidden layers and the learning rate. These hyperparameters are critical to achieving the best performance, and the grid search ensures that the parameter configurations are optimized for the task at hand, avoiding any bias due to arbitrary choices. The results, as shown in Table 2, show that our model outperforms other machine learning methods, proving the superiority of the Isolated Forest model for the first stage application.

Table 2. The experimental results for each method.

Method	Accuracy	FPR	Time (s)	Memory
iForest	0.9300	0.1059	4.9270	313.5312
AE	0.8926	0.0294	12.397	374.84
OC-SVM	0.9490	0.0811	6.6070	319.34
OC-SVM (GRL)	0.9785	0.0342	2.4402	181.93

As shown in Table 2, it can be revealed that when the first stage classification was performed directly without the GRL feature selection method, the accuracy of the model was 94.90%, and the computation time was 6.6070 s. The memory consumption was 319.34 MB, while with the addition of the integrated feature selection method GRL in this paper, the accuracy of the model was 97.85%, the computation time was 2.4402 s, and the memory consumption was 181.93 MB. In order to further prove the advantage of the GRL method, we used OC-SVM and OC-SVM (GRL) to conduct ten consecutive sets of tests, respectively. We compared the computation time and memory consumption of these two models in the form of line graphs to obtain more obvious differences in time and memory. As shown in Figure 4, the OC-SVM (GRL) model achieved better results in comparison with other commonly used machine learning models for anomaly detection.

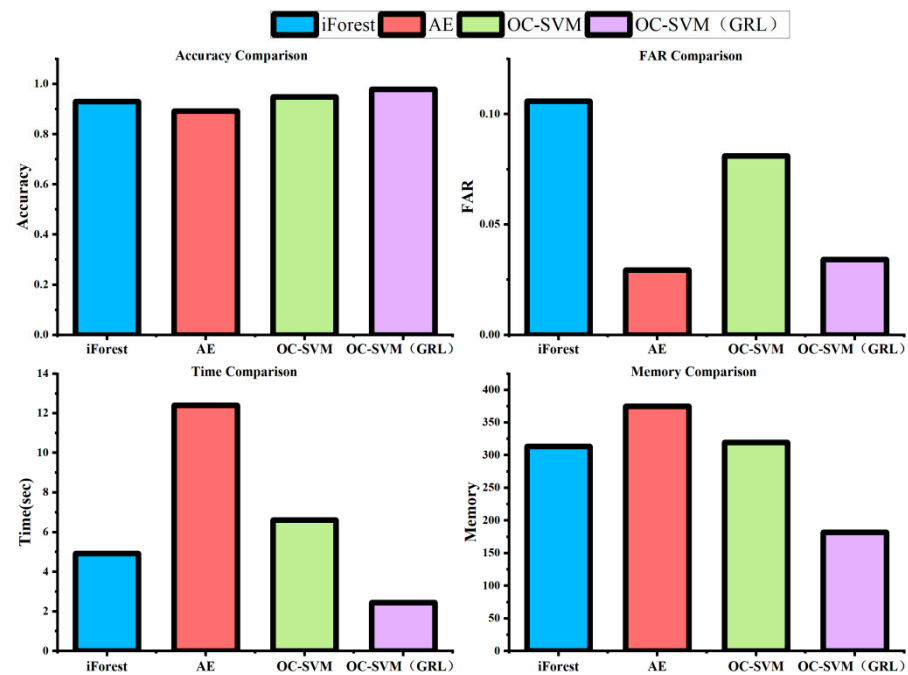


Figure 4. Comparison experiment chart.

The comparison results are shown in Figure 5: in terms of computation time, the time consumption of the OC-SVM (GRL) model was significantly lower than that of the traditional OC-SVM model, while the time consumption of the OC-SVM (GRL) was stabilized at about 3 s, and the computational efficiency of the model was significantly improved. In terms of memory consumption, the memory usage of the OC-SVM (GRL) model was always lower than 200 MB and remained stable across groups of experiments. Memory consumption was significantly reduced. After adding the GRL feature selection method, the model achieved less computation time while ensuring a similar high level of classification and lower memory consumption, indicating that the method was less computationally expensive while ensuring high accuracy of the classification model.

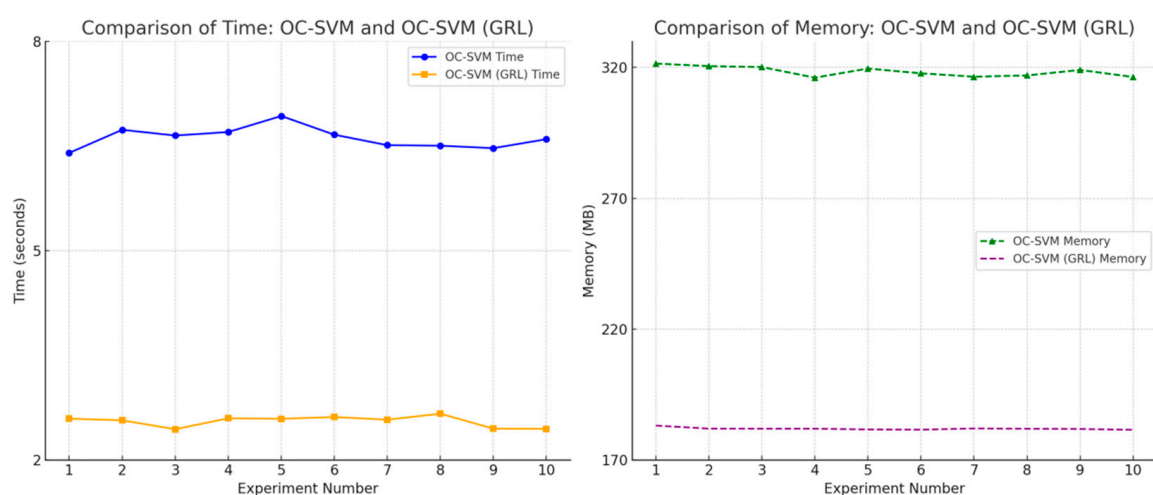


Figure 5. Time memory comparison chart.

In summary, by introducing the GRL method, the OC-SVM model achieves significant optimization in terms of computation time and memory occupation, further proving the effectiveness and applicability of the GRL method. Specifically, the first-layer model

proposed in this paper achieves a substantial reduction in computation and memory occupation compared to traditional methods by realizing lightweight, efficient, and fast characteristics, making the model very suitable for deployment in the near-end of the actual industrial control system network being monitored, especially in field environments requiring real-time response, which greatly enhances the flexibility and scalability of the system.

In order to further validate the detection capability of this paper's multilayer model, standard intrusion detection models were selected as comparison methods, which includes existing graph neural algorithms commonly used for intrusion detection, such as E-GraphSAGE and GCN algorithms. Similarly, the parameters of these models were optimized by grid search and cross-validation to ensure fairness in performance comparisons. The grid search process involves testing a set of predefined parameter values such as the learning rate, number of layers, and batch size for both E-GraphSAGE and GCN. By systematically tuning these hyperparameters, we can ensure that each model maximizes its potential, allowing for meaningful comparisons of the performance of different approaches. The evaluation metrics selected were accuracy, recall, precision, F1-score, FNR, and FPR. The false positive rate and false negative rate provide a more comprehensive understanding of the model's performance, particularly when balancing the detection of true intrusions and minimizing false alarms. FPR helps assess the model's ability to avoid incorrectly labeling normal traffic as attacks, while FNR measures the model's sensitivity in detecting actual attacks. Together, these two metrics ensure that the model not only achieves high detection accuracy but also maintains a low rate of false positives and false negatives, which is critical for real-world deployment. The results of the comparison test are shown in Table 3. It is evident that the model proposed in this paper demonstrates certain advantages over the comparison model.

Table 3. The experimental results of each method.

Method	Accuracy	Precision	Recall	F1	FNR	FPR
XGBoot	0.9089	0.9156	0.9015	0.9104	0.0871	0.0217
KNN	0.8971	0.9026	0.8914	0.8736	0.0713	0.0178
GCN	0.9239	0.9544	0.9352	0.9363	0.1053	0.0263
E-GraphSAGE	0.9531	0.9396	0.9531	0.9428	0.0408	0.0102
Ours	0.9699	0.9711	0.9699	0.9704	0.0263	0.0081

As can be seen from the table, the Accuracy, F1-score, Recall, and F1-score of the proposed model in this paper were 0.9699, 0.9711, 0.9699, and 0.9604. Compared with the other comparative models, these metrics were significantly improved, and compared with the E-GraphSAGE model, which performed better, the metrics of this paper's model also improved by about 1.6–3%, respectively. In addition, this paper's model also demonstrated significant improvement in the two metrics of false positive rate (FPR) and missed report rate (FNR), which were 0.0081 and 0.0263, respectively, and as can be seen in Figure 6, this paper's model was lower performing than the other comparative models. This indicates that the hierarchical intrusion detection model proposed in this paper is not only able to detect most of the attacks effectively but also has significant advantages in reducing false alarms and missed alarms. This indicates that the model proposed in this paper improves detection accuracy by effectively selecting the most critical features through GRL-integrated feature extraction. The inclusion of the multi-head dynamic attention mechanism further enhances EGAT by allowing it to flexibly adjust its focus on edge features, thereby improving the model's ability to capture complex attack patterns and achieving superior detection performance.

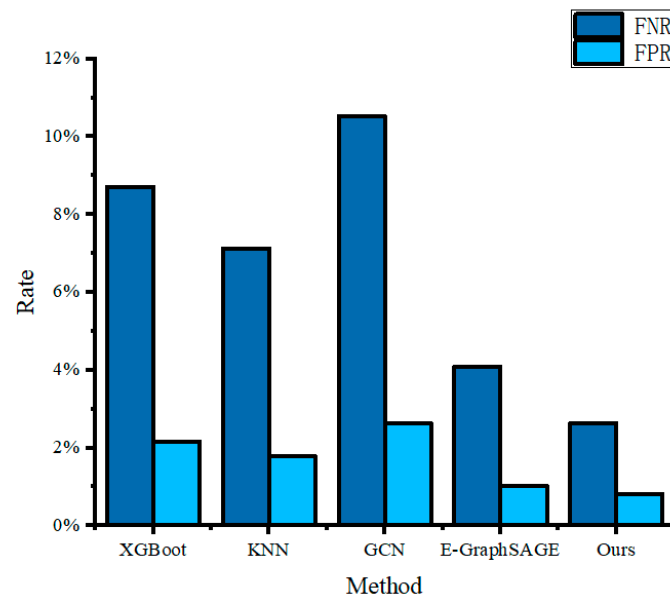


Figure 6. Comparison experiment chart.

The results show that the hierarchical intrusion detection model improves in terms of time efficiency and performs better in terms of detection effectiveness. To provide a more intuitive representation of the algorithms' effects, radar charts were used to visualize and compare the performance of the models across six dimensions, namely, accuracy, precision, recall, F1-score, leakage, and false alarms, as shown in Figure 7. It can be clearly seen that the model proposed in this paper outperforms the other compared models in all performance metrics, especially in accuracy, precision, and F1-score, and these results further demonstrate the effectiveness of this paper's model in improving intrusion detection performance. Specifically, the significant improvements in precision and F1-score suggest that the model is highly reliable [30] in detecting true positives while minimizing both false negatives and false positives, which is crucial for practical deployment in sensitive environments like industrial control systems.

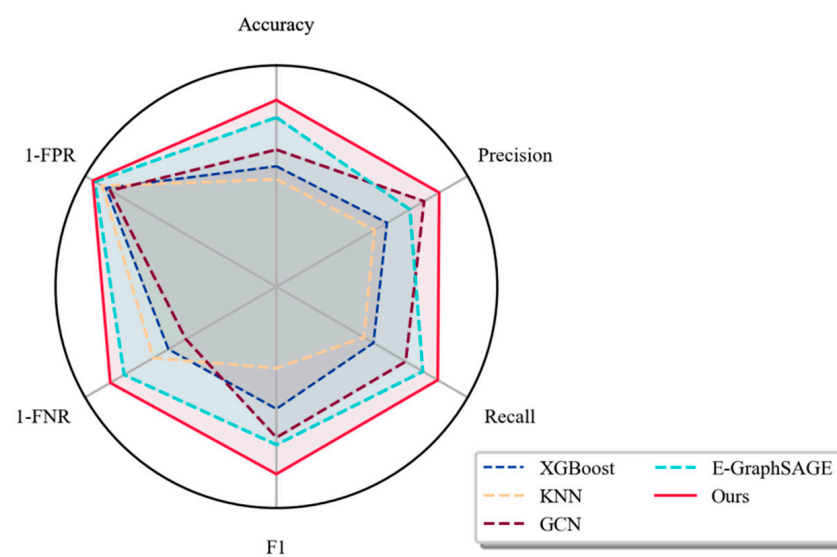


Figure 7. Comparison experiment radar chart.

5. Conclusions

In this research, a novel hierarchical multi-stage intrusion detection method was proposed, characterized by its privacy-preserving capabilities and high flexibility. Initially, the general framework and methodology were introduced, featuring the integration of common feature technology through GRL combined with OC-SVM. This integration aimed to achieve efficient anomaly detection in Stage 1 and to alleviate the computational load on the multi-classification model in Stage 2. Subsequently, in Stage 2, an enhanced graph-neural anomaly detection method, MHDEGAT, was introduced. This method augmented the focus on edge information beyond the traditional graph-neural approach and incorporated a multi-head dynamic attention mechanism to assign varying weights. This adjustment enabled a more comprehensive utilization of both node features and edge information, demonstrating the method's high adaptability and significantly improving global classification performance.

After describing the research methodology of this paper, an experimental evaluation was carried out with a factory PLC industrial control dataset. The experimental results show that our method is effective in detecting intrusions in industrial control systems. Furthermore, the dataset used in this study is sourced from the PLC network traffic of a real-world factory, making it highly representative of the operational characteristics of most industrial control systems. It accurately reflects the network behaviors in industrial environments. Therefore, the proposed method and the dataset utilized are not only applicable to the specific dataset used in this research but also have the potential to be generalized to other industrial control systems, demonstrating strong generalization ability.

The method in this paper achieves high-accuracy detection while improving flexibility and privacy protection. However, this approach has some limitations when facing unknown attacks, and our future work will focus on the problem of low-sample attacks and improving the detection of unknown attacks.

Author Contributions: Conceptualization, J.L. and L.Z.; methodology, J.L.; software, G.X.; validation, C.L. and Z.Z.; formal analysis, J.L.; writing—original draft preparation, J.L.; writing—review and editing, J.L.; visualization, L.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Our dataset has been described in detail in this paper.

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

References

1. Faramondi, L.; Flammini, F.; Guarino, S.; Setola, R. A hardware-in-the-loop water distribution testbed dataset for cyber-physical security testing. *IEEE Access* **2021**, *9*, 122385–122396. [\[CrossRef\]](#)
2. Gehrman, C.; Gunnarsson, M. A digital twin based industrial automation and control system security architecture. *IEEE Trans. Ind. Inform.* **2019**, *16*, 669–680. [\[CrossRef\]](#)
3. Hoang, N.X.; Hoang, N.V.; Du, N.H.; Huong, T.T.; Tran, K.P. Explainable anomaly detection for industrial control system cybersecurity. *IFAC-PapersOnLine* **2022**, *55*, 1183–1188.
4. Varghese, S.A.; Ghadim, A.D.; Balador, A.; Alimadadi, Z.; Papadimitratos, P. Digital twin-based intrusion detection for industrial control systems. In Proceedings of the 2022 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops), Pisa, Italy, 21–25 March 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 611–617.
5. Mubarak, S.; Habaebi, M.H.; Islam, M.R.; Rahman, F.D.; Tahir, M. Anomaly Detection in ICS Datasets with Machine Learning Algorithms. *Comput. Syst. Sci. Eng.* **2021**, *37*. [\[CrossRef\]](#)
6. Staniford-Chen, S.; Cheung, S.; Crawford, R.; Dilger, M.; Frank, J.; Hoagland, J.; Levitt, K.; Wee, C.; Yip, R.; Zerkle, D. GrIDS—a graph based intrusion detection system for large networks. In Proceedings of the National Information Systems Security Conference, (NISS), Baltimore, MD, USA, 22–25 October 1996; pp. 361–370.

7. Chikwendu, I.A.; Zhang, X.; Mensah, I.A.; Ukwuoma, C.C.; Ejiyi, C.J. A Comprehensive Survey on Deep Graph Representation Learning Methods. *J. Artif. Intell. Res.* **2023**, *78*, 287–356. [\[CrossRef\]](#)
8. Caville, E.; Lo, W.W.; Layeghy, S.; Portmann, M. Anomal-E: A self-supervised network intrusion detection system based on graph neural networks. *Knowl.-Based Syst.* **2022**, *258*, 110030. [\[CrossRef\]](#)
9. Lu, Y.; Xu, X.; Wang, L. Smart manufacturing process and system automation—a critical review of the standards and envisioned scenarios. *J. Manuf. Syst.* **2020**, *56*, 312–325. [\[CrossRef\]](#)
10. Hu, Y.; Yang, A.; Li, H.; Sun, Y.; Sun, L. A survey of intrusion detection on industrial control systems. *Int. J. Distrib. Sens. Netw.* **2018**, *14*, 1550147718794615. [\[CrossRef\]](#)
11. Zhang, H.; Li, J.L.; Liu, X.M.; Dong, C. Multi-dimensional feature fusion and stacking ensemble mechanism for network intrusion detection. *Future Gener. Comput. Syst.* **2021**, *122*, 130–143. [\[CrossRef\]](#)
12. Rodríguez, D.; Ruiz, R.; Cuadrado-Gallego, J.; Aguilar-Ruiz, J. Detecting fault modules applying feature selection to classifier. In Proceedings of the 2007 IEEE International Conference on Information Reuse and Integration, Las Vegas, NV, USA, 13–15 August 2007; IEEE: Piscataway, NJ, USA, 2007; pp. 667–672.
13. Aljawarneh, S.; Aldwairi, M.; Yassein, M.B. Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model. *J. Comput. Sci.* **2018**, *25*, 152–160. [\[CrossRef\]](#)
14. Ayo, F.E.; Folorunso, S.O.; Abayomi-Alli, A.A.; Adekunle, A.O.; Awotunde, J.B. Network intrusion detection based on deep learning model optimized with rule-based hybrid feature selection. *Inf. Secur. J. A Glob. Perspect.* **2020**, *29*, 267–283. [\[CrossRef\]](#)
15. 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. [\[CrossRef\]](#)
16. Otokwala, U.; Petrovski, A.; Kalutarage, H. Improving intrusion detection through training data augmentation. In Proceedings of the 2021 14th International Conference on Security of Information and Networks (SIN), Edinburgh, UK, 15–17 December 2021; IEEE: Piscataway, NJ, USA, 2021; Volume 1, pp. 1–8.
17. Daniya, T.; Geetha, M.; Kumar, K.S. Classification and regression trees with gini index. *Adv. Math. Sci. J.* **2020**, *9*, 8237–8247. [\[CrossRef\]](#)
18. Jeon, H.; Oh, S. Hybrid-recursive feature elimination for efficient feature selection. *Appl. Sci.* **2020**, *10*, 3211. [\[CrossRef\]](#)
19. Fong, S.; Narasimhan, S. An unsupervised Bayesian OC-SVM approach for early degradation detection, thresholding, and fault prediction in machinery monitoring. *IEEE Trans. Instrum. Meas.* **2021**, *71*, 3500811. [\[CrossRef\]](#)
20. Li, X.; Zhang, J.; Yuan, Y.; Zhou, C. Network Intrusion Detection with Edge-Directed Graph Multi-Head Attention Networks. *arXiv* **2023**, arXiv:2310.17348.
21. Zhou, L.; Zeng, Q.; Li, B. Hybrid anomaly detection via multihead dynamic graph attention networks for multivariate time series. *IEEE Access* **2022**, *10*, 40967–40978. [\[CrossRef\]](#)
22. Liu, Y.; Yang, C.; Huang, K.; Gui, W.; Hu, S. A systematic procurement supply chain optimization technique based on industrial internet of things and application. *IEEE Internet Things J.* **2022**, *10*, 7272–7292. [\[CrossRef\]](#)
23. Kim, J.; Lee, S.; Kim, Y.; Cho, S. A graph embedding-based identity inference attack on blockchain systems. In Proceedings of the 2022 International Conference on Electronics, Information, and Communication (ICEIC), Jeju, Republic of Korea, 6–9 February 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 1–3.
24. Li, X.; Zhu, M.; Yang, L.T.; Xu, M.; Ma, Z.; Zhong, C.; Li, H.; Xiang, Y. Sustainable ensemble learning driving intrusion detection model. *IEEE Trans. Dependable Secur. Comput.* **2021**, *18*, 1591–1604. [\[CrossRef\]](#)
25. Di Mauro, M.; Galatro, G.; Fortino, G.; Liotta, A. Supervised feature selection techniques in network intrusion detection: A critical review. *Eng. Appl. Artif. Intell.* **2021**, *101*, 104216. [\[CrossRef\]](#)
26. Otokwala, U.J.; Petrovski, A. Ensemble common features technique for lightweight intrusion detection in industrial control system. In Proceedings of the 2023 IEEE 6th International Conference on Industrial Cyber-Physical Systems (ICPS), Wuhan, China, 8–11 May 2023; IEEE: Piscataway, NJ, USA, 2023; pp. 1–6.
27. Liu, F.T.; Ting, K.M.; Zhou, Z.H. 2008 eighth ieee international conference on data mining. In Proceedings of the Eighth IEEE International Conference on Data Mining, Pisa, Italy, 15–19 December 2008; pp. 413–422.
28. Hnamte, V.; Hussain, J. DCNNBiLSTM: An efficient hybrid deep learning-based intrusion detection system. *Telemat. Inform. Rep.* **2023**, *10*, 100053. [\[CrossRef\]](#)
29. Shen, P.; Chen, L.J. Traffic anomaly detection method based on iForest score extension. *Electr. Meas. Technol.* **2024**, *47*, 157–163.
30. Catillo, M.; Pecchia, A.; Villano, U. Successful intrusion detection with a single deep autoencoder: Theory and practice. *Softw. Qual. J.* **2024**, *32*, 95–123. [\[CrossRef\]](#)

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.