Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.050223

ARTICLE

# A New Industrial Intrusion Detection Method Based on CNN-BiLSTM

**Jun Wang, Changfu Si, Zhen Wang and Qiang Fu***

College of Computer Science and Technology, Shenyang University of Chemical Technology, Shenyang, 110142, China

*Corresponding Author: Qiang Fu. Email: qiang.fu@outlook.com

Received: 31 January 2024 Accepted: 09 April 2024 Published: 20 June 2024

**ABSTRACT**

Nowadays, with the rapid development of industrial Internet technology, on the one hand, advanced industrial control systems (ICS) have improved industrial production efficiency. However, there are more and more cyber-attacks targeting industrial control systems. To ensure the security of industrial networks, intrusion detection systems have been widely used in industrial control systems, and deep neural networks have always been an effective method for identifying cyber attacks. Current intrusion detection methods still suffer from low accuracy and a high false alarm rate. Therefore, it is important to build a more efficient intrusion detection model. This paper proposes a hybrid deep learning intrusion detection method based on convolutional neural networks and bidirectional long short-term memory neural networks (CNN-BiLSTM). To address the issue of imbalanced data within the dataset and improve the model's detection capabilities, the Synthetic Minority Over-sampling Technique-Edited Nearest Neighbors (SMOTE-ENN) algorithm is applied in the preprocessing phase. This algorithm is employed to generate synthetic instances for the minority class, simultaneously mitigating the impact of noise in the majority class. This approach aims to create a more equitable distribution of classes, thereby enhancing the model's ability to effectively identify patterns in both minority and majority classes. In the experimental phase, the detection performance of the method is verified using two data sets. Experimental results show that the accuracy rate on the CICIDS-2017 data set reaches 97.7%. On the natural gas pipeline dataset collected by Lan Turnipseed from Mississippi State University in the United States, the accuracy rate also reaches 85.5%.

**KEYWORDS**

Intrusion detection; convolutional neural network; bidirectional long short-term memory neural network; multi-head self-attention mechanism

## 1 Introduction

The increasing use of standard protocols and devices in industrial systems that can connect to the Internet is causing these systems to become targets of more intrusions [1]. Traditional ICSs are based on physical isolation, leading people to pay more attention to functional safety issues while neglecting the issues related to information security. However, modern ICS is gradually moving towards informatization [2]. With the development of the Industrial Internet, ICS is becoming an open interconnected system, and various network attacks are emerging. The network security challenges of Industrial Control Systems are becoming increasingly severe [3], attracting more attention and research compared to traditional functional safety. Traditional security approaches for industrial networks

mainly rely on deploying security devices at the network edges, such as firewalls [4], which filter network traffic based on custom rules. However, this passive defense strategy can only intercept known network attacks and is unable to effectively identify and block new network threats. Therefore, there is a need for more proactive and intelligent intrusion detection methods.

The idea of the Intrusion Detection System (IDS) was first proposed by James Anderson of the National Security Agency in the 1980s [5]. It consists of a set of tools designed to assist administrators in reviewing and auditing tracking activities. As a rapidly advancing artificial intelligence technique, machine learning can uncover hidden patterns from a large amount of known data and use the learned patterns to predict and classify new data [6]. With the widespread adoption of machine learning, it exerts a fundamental influence on various disciplines, including the field of intrusion detection. In 1987, Narayan et al. [7] introduced an IDS that utilized artificial intelligence (AI) methods to detect abnormal traffic and potential intrusions. This method laid the foundation for intrusion detection based on machine learning algorithms. These IDS methods related to artificial intelligence can learn from known network attacks through training, thereby identifying new abnormal network traffic that has not been encountered before, significantly improving detection accuracy. Over the past 30 years, researchers have proposed numerous IDS based on an assortment of machine learning algorithms [8–10], and more. Gao et al. [11] used different classifiers as basic classifiers and proposed an adaptive ensemble learning voting algorithm to heighten accuracy. In 2015, Gaikwad et al. [12] proposed an intrusion detection technology based on machine learning ensemble method. Despite the use of various machine learning methods for IDS research to automatically identify normal and abnormal traffic in systems and networks, the issues of high false positives and low detection rates have not been completely resolved. To address these challenges, one approach is to improve existing network parameters through optimization algorithms, such as Differential evolution algorithm [13] and genetic algorithm [14], another approach is to enhance the learning capability of models by employing more complex deep models, such as CNN [15], Recurrent Neural Networks (RNN) [16], Deep Belief Networks (DBN) [17], and autoencoders [18].

Cao et al. [19] proposed a network intrusion detection model using CNN and Gated Recurrent Unit (GRU). The model achieved excellent detection results on the CICIDS-2017, UNSW-NB15, and KDD-NSL datasets. In the work by Su et al. [20], they introduced a traffic anomaly detection model named BAT, which integrates Bi-LSTM with an attention mechanism. The attention mechanism is employed to filter the network flow vectors produced by the Bi-LSTM model, enhancing the classification features for effective anomaly detection. The NSL-KDD dataset is utilized for testing, revealing accuracy rates of 84.25% in the KDDTest+ test set and 69.42% in KDDTest-21 testing sets. Sun et al. [21] devised a Deep Learning-based Intrusion Detection System. This system incorporates a hybrid network, combining CNN and LSTM architectures to extract both spatial and temporal features from network traffic. The effectiveness of this system in detecting intrusions was demonstrated through successful testing on the CICIDS-2017 dataset. Altunay et al. [22] proposed a hybrid model of CNN+LSTM and used the UNSW-NB15 and X-IIoTID data sets for verification in the experiment. The experimental process was very rigorous and comprehensive. By comparison, it has been demonstrated that deep learning methods exhibit superiority in detecting anomalous events within large and complex datasets.

Table 1 summarizes the methods mentioned in the literature review. Through the comparison of the data in the table, it becomes evident that the performance of deep learning models surpasses that of traditional machine learning methods. Therefore, it is imperative to conduct further exploratory research on the model structures of intrusion detection. When designing experiments, utilizing multiple datasets for verification can more effectively demonstrate the model's effectiveness.

**Table 1:** The description of existing work

| Authors | Model | Datasets | Accuracy |
|---|---|---|---|
| Gao et al. [11] | Ensemble voting | KDDTest+ | 85.2% |
| | Multi tree | | 84.23% |
| | DNN | | 81.61% |
| Gaikwad et al. [12] | Machine learning ensemble method | NSL_KDD | 99.6761% |
| Cao et al. [19] | CNN and gated recurrent unit (GRU) | CICIDS-2017 | 99.65% |
| | | UNSW-NB15 | 85.25% |
| | | NSL_KDD | 99.69% |
| Su et al. [20] | Bi-LSTM with an attention mechanism | NSL_KDD | 84.25% |
| Sun et al. [21] | CNN and LSTM | CICIDS-2017 | 98.67% |
| Altunay et al. [22] | CNN and LSTM | UNSW-NB15 | 93.21% |
| | | X-IIoTID | 99.84% |

As can be seen from the above review, there are still some problems to be solved in the field of deep learning intrusion detection. First, as attack methods become more and more complex, larger intrusion detection systems must be developed. Secondly, there should be innovations in the verification process, and more comprehensive verification methods should be used to prove the effectiveness of the method. Besides, it is observed that research on Intrusion Detection Systems (IDS) is primarily focused on datasets collected in laboratory settings. The effectiveness of detection on real intrusion data from industrial systems cannot be guaranteed. The method proposed in this paper utilizes various deep neural network models (CNN, Bi-LSTM) to enhance the accuracy of detecting network attacks. It has been tested on both the CICIDS-2017 dataset and a dataset collected from a real industrial environment involving natural gas pipeline data. The results demonstrate the efficiency and high accuracy of this approach in real-world industrial scenarios.

The proposed method contributes in the following ways:

1. This method represents an exploratory study based on a hybrid model. By adjusting the structure and parameters of the model, a new intrusion detection model is developed.
2. When analyzing the experimental results, they are thoroughly compared with other methods to demonstrate the superiority of the proposed approach.
3. To systematically assess the impact of each component of the model on the overall performance, components are selectively removed or disabled, confirming the rationale behind the proposed architecture.
4. The method was validated using real industrial datasets, demonstrating its applicability to real-world industrial environments. This challenges the prevalent practice of conducting research solely on laboratory datasets.

## 2 Background Knowledge

### 2.1 Convolutional Neural Network

The CNN is inspired by biological processes, gaining widespread adoption in recent years within the realm of deep learning [23]. Comprising a variety of layers with different structures, the CNN enables the concurrent utilization of multiple convolutional kernels for extracting diverse features through convolutional operations. The formula for convolutional operation is shown in Eq. (1):

$$x_j^l = f\left(\sum_{i \in M_j} x_i^{l-1} * k_{ij}^l + b_j^l\right) \tag{1}$$

Here, $x_j^l$ is the $j$-th feature vector in the $l$-th layer; $f$ is the activation function; $M_j$ is the set used to calculate the $j$-th feature vector in the $l$-th layer; $x_i^{l-1}$ is the $i$-th feature vector in the $l-1$ layer; $k_{ij}^l$ is the convolution kernel, and $b_j^l$ is the offset of the $j$-th feature vector in the $l$-th layer.

The pooling layer's main function is to compress the features obtained after convolution [24]. The pooling layer's calculation process is like the convolutional layer, where the original feature vectors are computed with a pooling window. The difference lies in that, while the convolutional layer alters the feature values using weighted parameters in the convolutional kernel, the pooling layer directly calculates the original data within the pooling window. Pooling operations commonly employed in neural networks encompass both max pooling and average pooling techniques, which reduce parameters, lower data dimensions, and to some extent, prevent overfitting. The model proposed in this paper utilizes max pooling layers, extracting the maximum feature values from each specified window. This significantly reduces the length of the feature vectors, alleviating sensitivity to the position of features that need to be identified.

Following a series of convolutional and pooling operations, the data is ultimately fed into a fully connected layer, commonly referred to as Dense. In this layer, the output of the preceding layer (typically a convolutional or pooling layer) is flattened, creating a one-dimensional vector. The primary objective is to convert the high-dimensional feature representation into a more manageable, one-dimensional form. Upon connection to the Dense layer, each neuron establishes connections with all features from the preceding layer. This allows the Dense layer to capture complex relationships between all features in the input data.

### 2.2 Bidirectional Long Short-Term Memory Neural Network

The LSTM architecture represents a neural network innovation designed to tackle the challenge of managing long-term dependencies within RNNs. It introduces specialized storage units and cell states to specifically address the issue of prolonged dependencies, a limitation inherent in traditional RNNs. This enhancement in architecture aims to overcome the long-term dependency problem, providing a more effective framework for capturing and retaining information over extended sequences in data. The long-term dependency problem in RNNs [25] refers to the challenges faced by RNNs in computing relationships between distant nodes, often due to the issues of gradient explosion or gradient vanishing. In comparison to RNNs, LSTM can preserve historical information through the cell state, and the storage unit can decide whether to store information in in the cell state through gate mechanisms [26]. A standard RNN consists of multiple individual structures, such as a tanh activation layer. In contrast, an LSTM is composed of different gates, providing more flexibility in data processing. Since RNN simply stacks these modules together, it cannot selectively forget historical information during

data transmission. This may lead to problems of long-term dependencies in the data, resulting in poor training performance or model failure.

Bi-LSTM is an extension of LSTM [27]. While LSTM addresses the issue of not being able to retain feature information, Bi-LSTM structure is created by merging forward LSTM layers with backward LSTM layers [24]. This configuration enables the model to gather information from both preceding and succeeding contexts, allowing it to capture both past and future information simultaneously. Each recurrent unit in Bi-LSTM has feedback connections, providing memory and further exploration capabilities. By considering both past and future features of the sequence, Bi-LSTM enhances the model's performance in handling sequential problems.

The computation process of a Bi-LSTM involves various gates and states. Here are the key equations that describe the passes of a Bi-LSTM:

At time step $t$, the forward pass equations are as follows:

Input gat $i_t$ and Input Activation $\tilde{C}_t$

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \tag{2}$$

$$\tilde{C}_t = \tanh\left(W_C \cdot [h_{t-1}, x_t] + b_C\right) \tag{3}$$

Forget Gate $f_t$

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \tag{4}$$

Cell State $C_t$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{5}$$

Output Gate $o_t$ and Hidden State $h_t$

$$o_t = \sigma\left(W_o \cdot [h_{t-1}, x_t] + b_o\right) \tag{6}$$

$$h_t = o_t * \tanh\left(C_t\right) \tag{7}$$

The backward pass equations are like the forward. The final hidden state $h_t$ for time step t is obtained by concatenating the forward and backward hidden states:

$$h_t = \left[h_t^f; h_t^b\right] \tag{8}$$

Finally, we can get the hidden layer state sequence:

$$\{h_0, h_1 \ldots, h_t\} \tag{9}$$

The diagram in Fig. 1 depicts the neuronal architecture of a single LSTM neuron.

Bi-LSTM neural network architecture is depicted in Fig. 2.

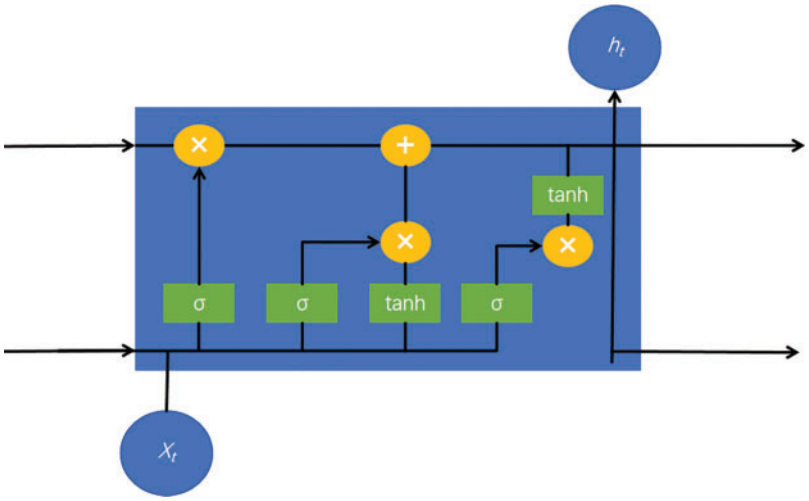



**Figure 1:** The neuron structure of the LSTM neural network

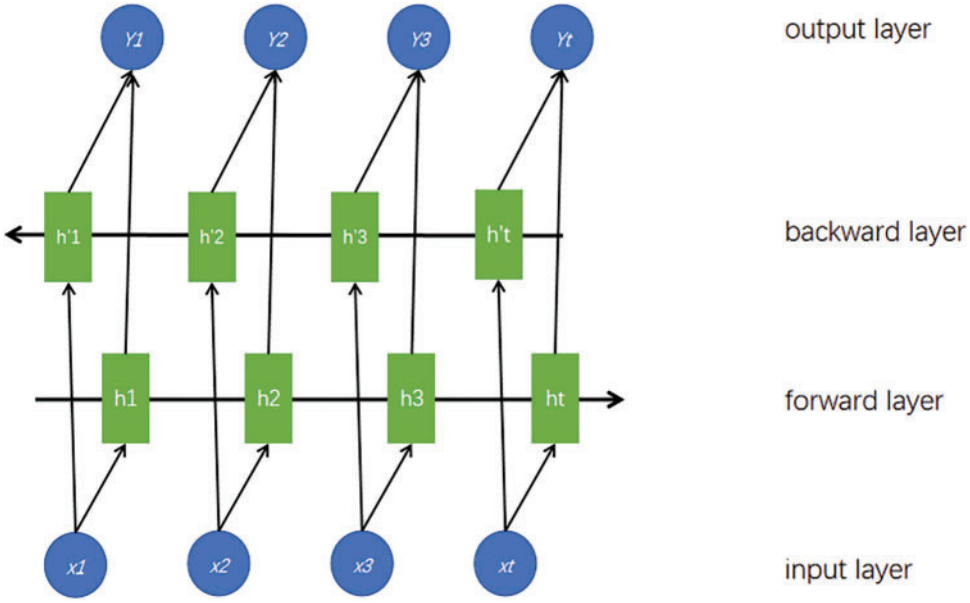



**Figure 2:** The structure of the Bi-LSTM neural network

## 3 Datasets and Data Preprocessing

### *3.1 CICIDS-2017 Dataset*

Since its development, the CICIDS-2017 dataset has garnered significant attention from numerous researchers and students. Based on the analysis and research of this dataset, many new intrusion detection models have been proposed. The dataset originates from the Canadian Institute for Cybersecurity and comprises 8 different files, containing both normal and attack traffic data spanning 5 days.

This dataset is extensive, and it simultaneously presents a significant class imbalance issue. This imbalance implies that most of the training time is spent learning how to detect the more numerous

classes, while the model struggles to effectively learn and identify the minority classes. Such a situation may lead to reduced precision and increased rates of false positives in the model's detection results [28].

To address these issues, methods proposed by Goryunov et al. [29], Stiawane et al. [30], Salo et al. [31], and others were consulted. These methods effectively alleviate class imbalance by merging minority classes to form a new attack category and removing some data from the majority classes. The preprocessed dataset features are referred to as CICIDS2017_sample, providing details about the characteristics and contents of new dataset in Table 2.

**Table 2:** The description of CICIDS-2017 dataset

| Serial number | Labels | Number of instances |
|---|---|---|
| 0 | BENIGN | 22767 |
| 1 | DoS | 19035 |
| 2 | PortScan | 7946 |
| 3 | BruteForce | 2767 |
| 4 | WebAttack | 2180 |
| 5 | Bot | 1966 |

### *3.2 Gas Pipeline Dataset*

In addition to utilizing the CICIDS-2017 dataset, to ensure the effectiveness of the proposed method in a real industrial internet environment, this study also employed a dataset collected from an actual industrial setting. The dataset was obtained from network traffic data logs of natural gas pipelines by a research team at Mississippi State University [32]. The natural gas pipeline data logs were captured in a laboratory setting. Data in this dataset comprising regular information and diverse forms of network attack data. They are categorized into eight classes, as illustrated in the following figure. The specific distribution of the data is outlined in Table 3.

**Table 3:** The description of natural gas pipeline dataset

| Serial number | Abbreviation |
|---|---|
| 0 | Normal(0) |
| 1 | NMRI(1) |
| 2 | CMRI(2) |
| 3 | MSCI(3) |
| 4 | MPCI(4) |
| 5 | MFCI(5) |
| 6 | DOS(6) |
| 7 | Recon(7) |

### *3.3 Data Preprocessing*

#### *3.3.1 Normalization*

Both parameter-based models and distance-based models require feature normalization. There are two commonly used methods for normalizing continuous features: Min-Max scaling (linear normalization) and Standard Scaling (standardization). In this paper, the Min-Max scaling technique is employed, facilitating the transformation of data into a range spanning from 0 to 1. The Min-Max scaling formula applied is outlined as follows:

$$x_{\_scale} = \frac{x - x_{\min}}{x - x_{\max}} \tag{10}$$

Here, $x$ represents the initial value in the data set, $x_{\_scale}$ is the value after normalization, $x_{\max}$ and $x_{\min}$ represent the range of values for $x$.

#### *3.3.2 Oversampling*

In the CICIDS-2017_sample dataset, although the problem of data imbalance has been alleviated to some extent, it persists. If the original data is directly used for training, it would lead the detection model to favor the majority class. As a result, the detection performance would be poor, with an increase in false positive rate and a decrease in accuracy. To address this issue, the SMOTE-ENN algorithm is employed to further tackle the problem of data imbalance.

The SMOTE algorithm is one of the commonly used oversampling methods. In essence, the algorithm involves randomly generating new instances of minority class samples, thereby increasing the quantity of these minority class instances. Specifically, for each instance of minority class samples, the algorithm identifies the closest neighbors belonging to the identical class. Subsequently, interpolation is performed between them to create new instances. This process is repeated to generate enough new instances, effectively increasing the quantity of instances representing the less prevalent class and achieving a balance in the dataset across different classes [33].

Here is a pseudocode description of the SMOTE algorithm:

**Algorithm:** SMOTE

**Input:** Instances that belong to the less represented class (min_class), Number of synthetic examples to generate (N), SMOTE ratio (k_neighbors)
**Output:** Synthetic minority class examples (synthetic_examples)
for each minority example in min_class:
  Find k_neighbors nearest neighbors of the current example in min_class using a distance metric (e.g., Euclidean distance).
  for i = 1 to N:
    Randomly select one of the k_neighbors, let's call it nn.
    Compute the difference vector diff = nn – current_example.
    Generate a random number between 0 and 1, let's call it rand.
    Create a synthetic example as follows: Synthetic_example = current_example + rand ∗ diff.
    Add synthetic_example to the synthetic_examples.
return synthetic_examples

The ENN algorithm considers the nearest neighbor relationship of samples and removes samples from the predominant class that are obviously inconsistent with the decision boundaries of adjacent

instances from the most prevalent class to improve the balance of the dataset [34]. The application of ENN helps eliminate some noise in the majority class and abnormal samples near the decision boundary, thereby improving model performance.

Here is a pseudocode description of the ENN algorithm:

**Algorithm:** ENN

**Input:** Dataset (X, y), Number of nearest neighbors to consider (k_neighbors)
**Output:** Cleaned dataset (X_cleaned, y_cleaned)
for each example in X:
  Find k_neighbors nearest neighbors of the current example using a distance metric (e.g., Euclidean distance).
  if example is misclassified:
    Remove the example from the dataset.
X_cleaned = remaining examples in X
y_cleaned = corresponding labels for X_cleaned
return X_cleaned, y_cleaned

The core idea of SMOTE-ENN is to synthesize new artificial data through SMOTE, and then remove the noise in the majority class samples through ENN, thereby realizing the processing of imbalanced datasets [35].

### *3.3.3 K-Fold Cross-Validation*

When dealing with data imbalance, the utilization of K-fold cross-validation becomes pivotal to ensure a thorough and robust evaluation of machine learning model performance. This approach entails partitioning the dataset into K subsets, and the model undergoes evaluation through K distinct training-validation iterations. In each cycle, one subset serves as the validation set, while the others are used for training. This methodology enhances the reliability and comprehensiveness of assessing the model's performance under varying data scenarios. This process yields estimates of model performance for K different models, and their average performance is calculated. This approach provides a more realistic and comprehensive assessment of model performance. In experiments, different values of K are chosen for testing, and the parameter values yielding the best performance are further evaluated.

After the processing, Table 4 represents the distribution of the train set data of CICIDS-2017.

**Table 4:** The distribution of the train set data of CICIDS-2017

| Serial number | Before oversampling | After oversampling |
|---|---|---|
| 0 | 18973 | 18404 |
| 1 | 1639 | 18899 |
| 2 | 2306 | 18928 |
| 3 | 15863 | 18870 |
| 4 | 6621 | 18950 |
| 5 | 1816 | 18895 |

## 4 IDS Architecture

This chapter provides an overview of the deep learning IDS structure, with specific parameter settings illustrated through experiments conducted on the CICIDS-2017 dataset. There are slight adjustments to the parameters for another dataset, but the structure remains consistent.

### *4.1 Convolution Layer and Pooling Layer*

After preprocessing, the dataset undergoes convolution and pooling operations. To make efficient use of the parallel computing capabilities of the GPU, improve training efficiency, and consider memory consumption, an appropriate batch size needs to be chosen before training. For this task, a batch size of 32 was selected. The construction of the neural network model then begins.

Firstly, the input undergoes convolutional operations. The convolutional layer, by using shared weights, reduces the number of model parameters and enhances the model's generalization capability. This is particularly useful for intrusion detection tasks as it allows better adaptation to different network traffic patterns. The feature vector dimension of the input is (77, 1). A convolutional layer with a kernel_size of 64 is used, and the activation function is chosen as ReLU. To maintain the input and output dimensions, prevent information loss, and avoid feature map shrinking, padding is applied with the 'same' setting. Convolutional operations effectively extract local features from input data, saving computational time and improving the efficiency of intrusion detection through parameter sharing. Theoretically, using smaller convolutional kernels can improve parameter sharing and reduce computational complexity. However, empirical evidence shows that selecting larger convolutional kernels leads to faster training. Specifically, with a kernel_size = 2, one training epoch takes around 170 s, while with kernel_size = 64, one training epoch takes less than 100 s. This might be related to the overall model architecture and data format.

After convolutional layer processing, the feature vector's dimension becomes (77, 64), which is not favorable for subsequent processing. Therefore, a max-pooling layer is added after the convolutional calculations, with a pool_size set to 8. This effectively reduces data dimensionality, lessening the computational burden while retaining crucial information. For intrusion detection tasks, this helps preserve key features in network traffic and reduces model complexity to prevent overfitting. Following this, a batch normalization layer is added, providing a regularization effect and reducing the risk of overfitting, thereby improving the model's robustness.

### *4.2 Bi-LSTM Layer*

As mentioned above, convolutional layers and pooling layers can extract features from data, preserving the spatial hierarchy of the input data and helping the model learn crucial spatial patterns. However, with the evolution of industrial network attack patterns, there is an increasing occurrence of progressive, multi-step complex attacks. To identify these attacks, it is often necessary to capture long-distance dependencies in the data sequence.

The Bi-LSTM model consists of two independent LSTM networks. The input sequences are put into the two LSTM models in both the forward and reverse directions for feature extraction. The final feature representation is formed by concatenating the output vectors from both LSTM models. The design philosophy of the Bi-LSTM model is to enable the feature data obtained at time t to simultaneously possess information from both the past and the future. Due to its ability to capture contextual information, Bi-LSTM is commonly used for feature extraction in text content, facilitating a more comprehensive understanding of attack behavior and underlying patterns. This mode can also be applied to intrusion detection tasks. For progressive multi-step attacks, the attacker's strategy may

change over time, and the dynamic nature of Bi-LSTM neural networks allows them to adapt better to such variations.

The first Bi-LSTM layer in the model I proposed consists of 32 units, followed by normalization and pooling processes. Subsequently, a second Bi-LSTM layer is added, comprising 64 units. The advantage of this structure lies in the gradual increase in the number of units in the two Bi-LSTM layers, representing a learning strategy from coarse to fine. This approach allows the model to capture broader patterns and features in the early layers and then refine these representations in subsequent layers. This is beneficial for learning hierarchical structures and complex representations in the data.

### *4.3 Fully Connected Layer and Output*

After the processing, a fully connected layer is ultimately used to change the data dimension, allowing for normal output. Before the output, a dropout layer is applied to randomly set a fraction of the data outputs to zero. This serves as a powerful regularization method to effectively prevent overfitting, especially given the combination of CNN and RNN in the model, which tends to increase the risk of overfitting and could lead to suboptimal performance when applied to the test set. The dropout parameter is set to 0.6. The SoftMax activation function is applied to transform the raw outputs of the model into a form representing the probability distribution over classes. This serves as the output layer of the model.

To summarize the entire contents of this chapter, the complete framework diagram of the intrusion detection system proposed in this article can now be given in Fig. 3.

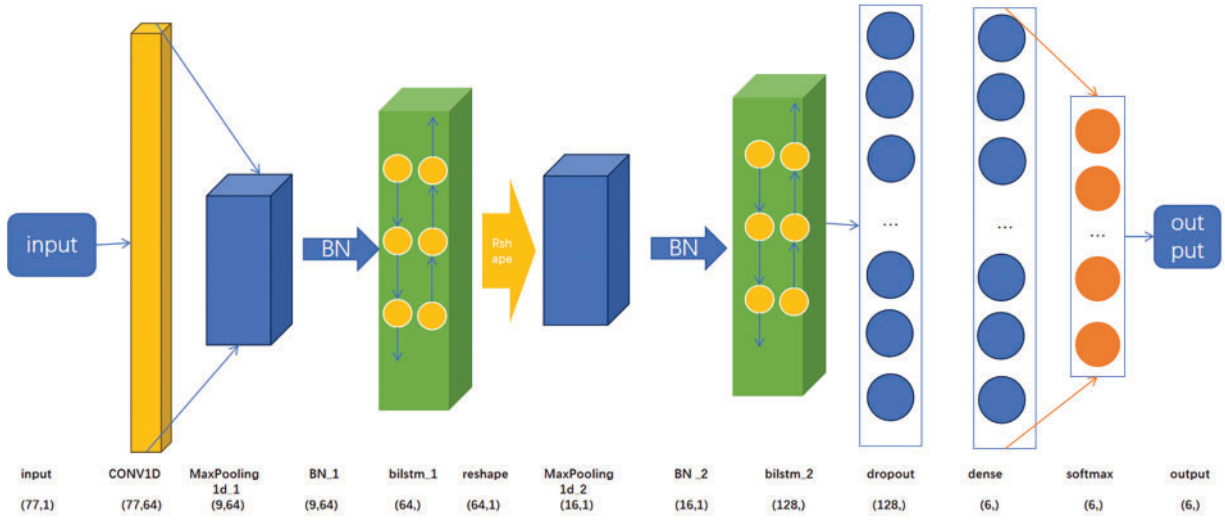



**Figure 3:** Process of my model

## 5 Comparative Analysis of Experimental Performance

This experiment utilizes two datasets to assess the effectiveness of my deep learning hybrid model detection method: The CICIDS-2017 dataset and the natural gas pipeline dataset. Firstly, experiments conducted on the CICIDS-2017 dataset are presented, showcasing various metrics to demonstrate that the model is competent for intrusion detection tasks. A comparison is made with detection methods proposed in other literature, highlighting the superior effectiveness of this approach over other deep

learning methods. Subsequently, my model is applied to the natural gas pipeline dataset, providing evidence that the method can work on industrial data.

### *5.1 Evaluation Metric*

The assessment of model performance incorporates several key metrics, including accuracy, precision, recall, F1-score, and the ROC-AUC curve. Accuracy gauges the model's proficiency in predicting across all classes. Precision specifically signifies the model's precision in predicting positive examples. Recall means the degree of coverage of positive examples by the model. Precision and recall may not fully describe a model's performance alone, and F1-score provides a more realistic measure of performance. The definitions for the indicators are presented in Eqs. (11)–(14).

In this experiment, a confusion matrix is employed to illustrate the practical detection performance of the model [36]. The confusion matrix effectively presents the distinctions between observed and predicted classes, offering a comprehensive depiction of the model's detection outcomes. In the matrix, it can be clearly seen that how much data is correctly classified and how much data is not correctly classified.

In my tasks, the matrix utilized is of size N $*$ N, with N denoting the total number of data categories. Table 5 is an example.

**Table 5:** The confusion matrix for multi classification tasks

| | $CLASS_1$ | $CLASS_2$ | ... | $CLASS_N$ |
|---|---|---|---|---|
| $CLASS_1$ | $A_{11}$ | $A_{12}$ | | $A_{1N}$ |
| $CLASS_2$ | $A_{21}$ | $A_{22}$ | | $A_{2N}$ |
| ... | | | | |
| $CLASS_N$ | $A_{N1}$ | $A_{N2}$ | | $A_{NN}$ |

According to the confusion matrix, parameters evaluating models are calculated:

$$Accuracy\,(A_1) = \frac{\sum_{i=1}^{n} A_{ii}}{\sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij}} \tag{11}$$

$$Percision\,(A_1) = \frac{A_{11}}{\sum_{i=1}^{n} A_{i1}} \tag{12}$$

$$Recall\,(A_1) = \frac{A_{11}}{\sum_{i=1}^{n} A_{1i}} \tag{13}$$

$$F1 - score = \frac{2 * Percision * Recall}{Percision + Recall} \tag{14}$$

### *5.2 Performance Analysis on CICIDS-2017 Dataset*

First, experiments were conducted on the CICIDS-2017 data set to determine the data preprocessing method and model parameters. The figure below shows the impact of k-fold cross-validation and

oversampling techniques on detection accuracy. As you can see from the Fig. 4, the experiment should use 6-fold cross-validation and SMOTE-ENN oversampling technology to preprocess the data.

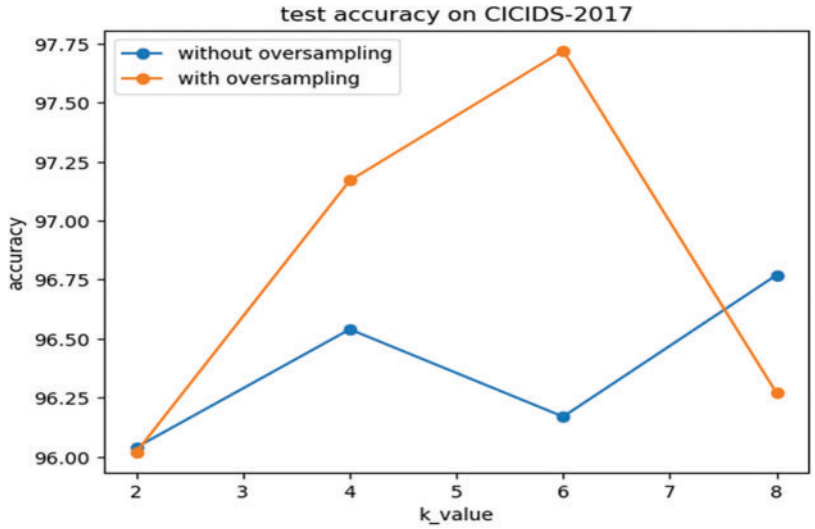



**Figure 4:** Accuracy of multiple categories on CICIDS-2017

Then adjusted the parameters and structure of my model. From the comparison in the Fig. 5, it can be learned that the model I proposed has better detection performance. If k-fold cross-validation was not performed, the detection accuracy would decrease. A smaller convolution kernel could reduce the parameters of the model, accelerating training, but the detection accuracy would decrease too. If a more complex network structure was introduced, such as adding a self-attention layer, in an attempt to enhance the feature extraction capabilities of the model, the detection performance became unstable, and the average result was still inferior to my method. Therefore, it can be said that my method is very effective and the selection of parameters is reasonable.

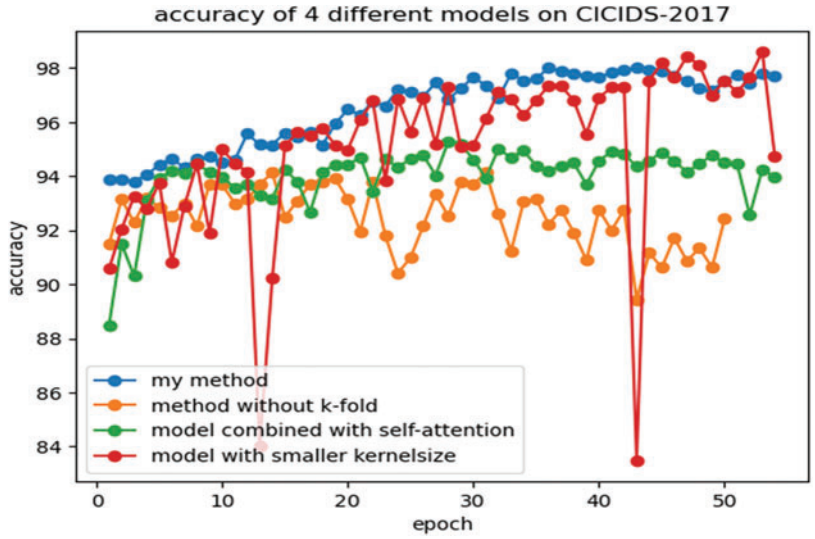
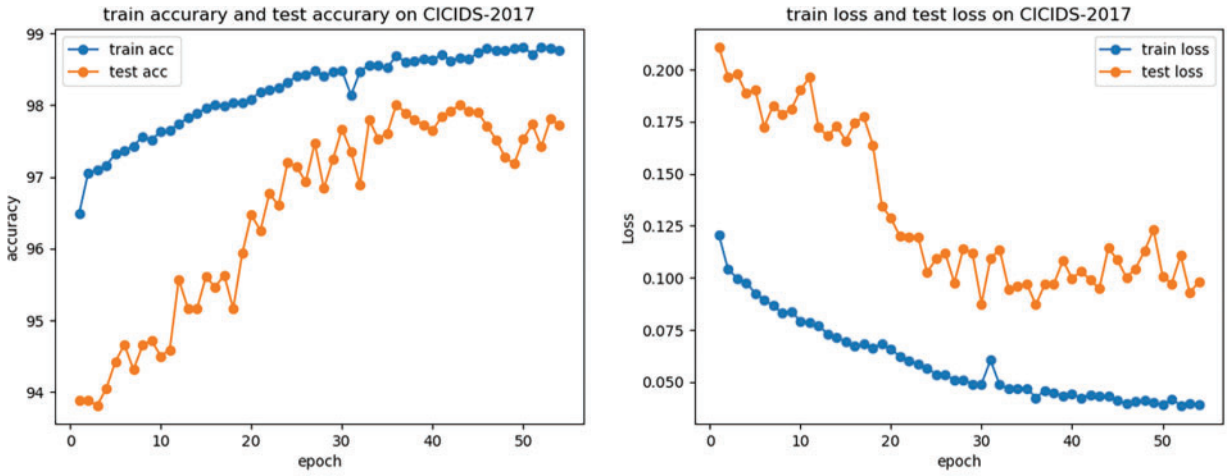



**Figure 5:** Population selection

The Figs. 6–8 give more detailed detection results. My method not only has good overall detection results but can also accurately classify each type of data.




(a) train accuracy and test accuracy (b) train loss and test loss

**Figure 6:** Accuracy and loss of multiple categories on CICIDS-2017

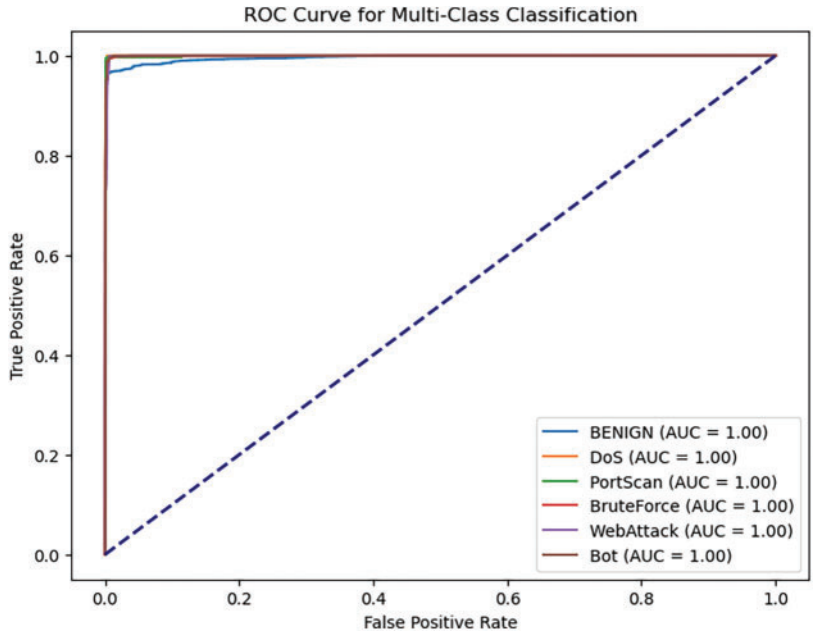



**Figure 7:** ROC curve for multi-class classification on CICIDS-2017 dataset

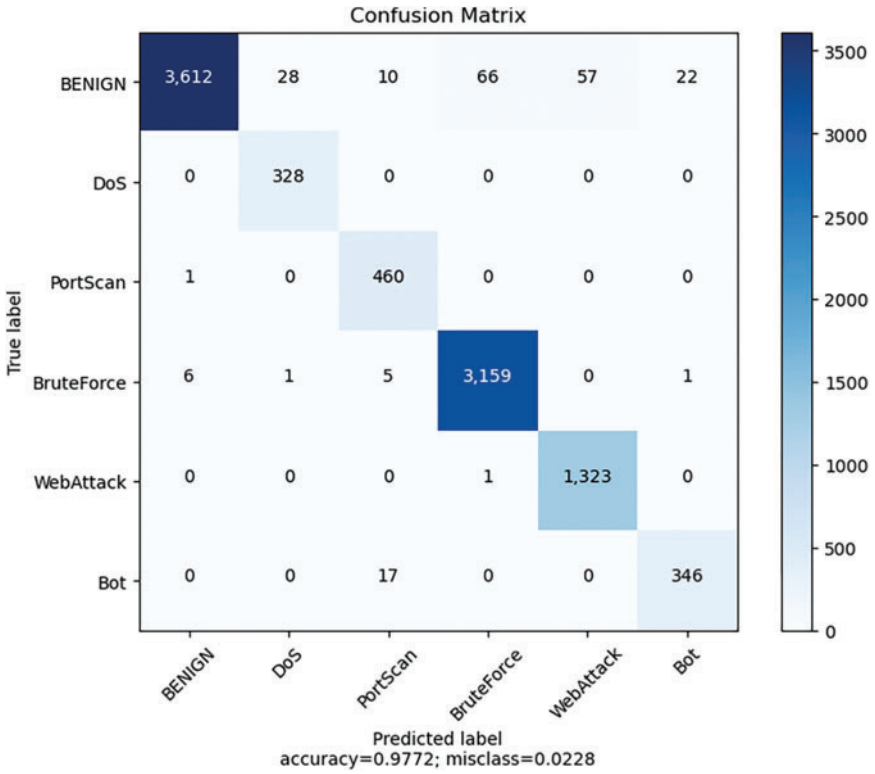



**Figure 8:** Confusion matrix for multi-class classification on CICIDS-2017 dataset

### *5.3 Comparison between CNN-Bi-LSTM and Other Models*

Table 6 evaluates the detection performance by comparing the method proposed in my paper with other existing methods introduced in other studies. The referenced papers utilized methods such as MLP, LSTM, Deep-GFL, etc. In comparison to Paper 1 and Paper 2, my method demonstrates superior precision, recall, and F1-score. In Paper 3, authors proposed an LSTM-based approach with higher precision; however, this model is only suitable for binary classification, distinguishing between normal and attack data. It cannot handle multi-class classification for various attack types, unlike my method, which can classify multiple different categories within the dataset.

**Table 6:** The comparison with other method

| Study | Method | Precision | Recall | F1-score |
|---|---|---|---|---|
| This paper | CNN-Bi-LSTM | 0.978 | 0.977 | 0.977 |
| Paper1 [37] | MLP | 0.871 | 0.995 | 0.873 |
| Paper2 [38] | Deep-GFL | 0.948 | 448 | 0.531 |
| Paper3 [39] | MLP | 0.884 | 0.862 | 0.872 |
| Paper3 [39] | LSTM | 0.984 | 0.898 | 0.895 |

### *5.4 Ablation Study and Theoretical Analysis*

To demonstrate the rationality of the proposed model structure, this section employs an ablation study method, conducting experiments to validate the respective roles of the CNN module and Bi-LSTM in the model. Furthermore, the principles of the model are analyzed and discussed.

The Fig. 9 illustrates the detection performance of the model after removing certain modules. The accuracy of the detection demonstrates that using either the CNN model or the Bi-LSTM model alone does not achieve detection effectiveness comparable to the hybrid model.

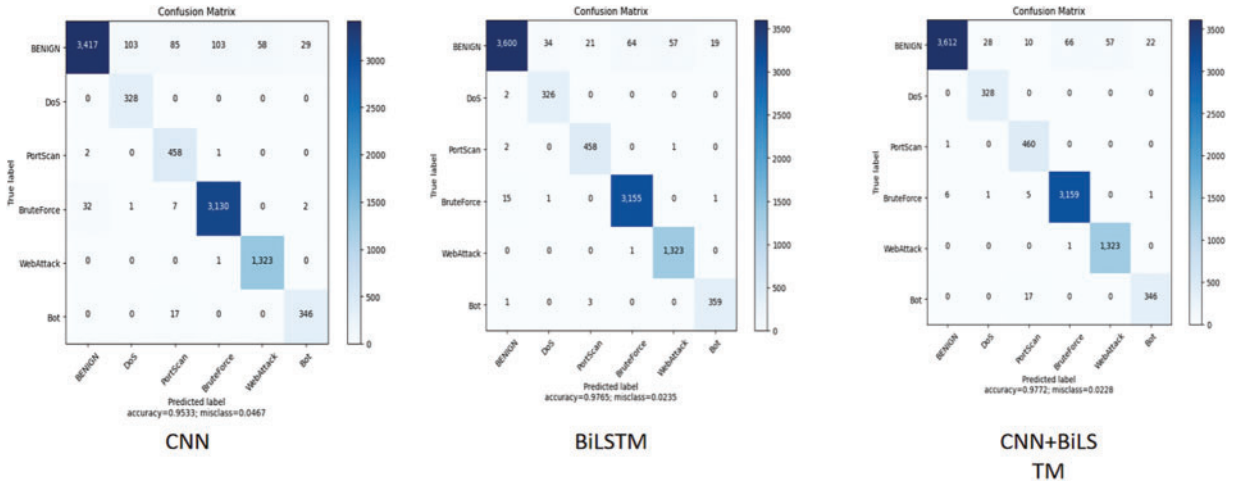



**Figure 9:** The results of ablation study

Observing the Tables 7 and 8, it is evident that the Bi-LSTM model exhibits a significant advantage in detecting DOS and PSCAN attacks, with higher accuracy compared to the CNN model. However, the CNN model demonstrates lower accuracy but faster detection speed. This paragraph aims to analyze and explain these results.

**Table 7:** The precision, recall, F1-score of CNN model

| Number | Pre | Rec | F1-score |
|---|---|---|---|
| 0 | 0.990 | 0.900 | 0.943 |
| 1 | 0.759 | 1.00 | 0.863 |
| 2 | 0.808 | 0.993 | 0.891 |
| 3 | 0.968 | 0.987 | 0.977 |
| 4 | 0.958 | 0.999 | 0.978 |
| 5 | 0.918 | 0.953 | 0.935 |

The superior performance of the Bi-LSTM model in detecting DOS and PSCAN attacks can be attributed to its ability to capture long-term dependencies and contextual information inherent in sequential data. DOS and PSCAN attacks often involve intricate patterns and subtle variations over time, which are better captured by the sequential nature of Bi-LSTM. Additionally, the bidirectional modeling of Bi-LSTM enables it to effectively incorporate past and future information, enhancing its ability to discern abnormal patterns in network traffic data.

**Table 8:** The precision, recall, F1-score of Bi-LSTM model

| Number | Pre | Rec | F1-score |
|---|---|---|---|
| 0 | 0.994 | 0.949 | 0.971 |
| 1 | 0.903 | 0.994 | 0.946 |
| 2 | 0.950 | 0.993 | 0.971 |
| 3 | 0.980 | 0.995 | 0.987 |
| 4 | 0.958 | 0.999 | 0.978 |
| 5 | 0.947 | 0.989 | 0.968 |

On the other hand, the CNN model's lower accuracy may stem from its limitation in capturing long-range dependencies and contextual information present in sequential data. CNNs excel in extracting local features through convolutional operations but may struggle to capture temporal dynamics inherent in sequential data. However, the CNN model's faster detection speed can be advantageous in scenarios where real-time detection is paramount, as it processes data in parallel across different regions. the Bi-LSTM model's superior performance in detecting DOS and PSCAN attacks underscores its capability to capture long-term dependencies and contextual information in sequential data. Meanwhile, the CNN model's faster detection speed makes it suitable for scenarios requiring real-time detection, albeit at the expense of slightly lower accuracy.

In summary, the CNN-Bi-LSTM deep learning model can effectively extract features, capture sequence information, and understand context in intrusion detection tasks, and has good robustness and generalization capabilities, so it has great potential and superiority in practical applications.

### *5.5 Performance Analysis on Natural Gas Pipeline Dataset*

The experimental data in the previous section was mainly obtained on the CICIDS-2017 data set, which initially proved the detection performance of this method. In order to further prove that this method is still effective in real industrial Internet intrusion detection tasks, this detection model was applied to a natural gas pipeline data set collected in a real industrial system.

Raw natural gas pipeline data only has 17 features, but the data volume is larger and there are more categories. Before training, the size of the input and output layers need to be adjusted, as well as the convolution kernel, so that the model can process the data smoothly, and other parameters do not need to be changed. It can be seen from Table 9, Fig. 10 that the model's ability to identify certain attack data is insufficient.

To solve this problem, the classification strategy must be changed. For an industrial system, it only needs to receive normal industrial data and does not need other data. If it can prevent attacks from entering the system, it does not care about the type of intrusion data. Therefore, the natural gas pipeline data set can be classified into two categories. As can be seen from Fig. 11, this model can correctly identify normal data and attack data. It can therefore be concluded that this model can work in industrial systems.

**Table 9:** The precision, recall, F1-score of every kind of data in natural gas pipeline dataset

| Categories | Pre | Rec | F1-score |
|---|---|---|---|
| 0 | 0.851 | 0.997 | 0.919 |
| 1 | 0.000 | 0.000 | 0.000 |
| 2 | 0.880 | 0.020 | 0.040 |
| 3 | 0.910 | 0.402 | 0.557 |
| 4 | 0.968 | 0.460 | 0.623 |
| 5 | 0.927 | 1.00 | 0.962 |
| 6 | 0.982 | 0.442 | 0.610 |
| 7 | 1.000 | 0.895 | 0.944 |

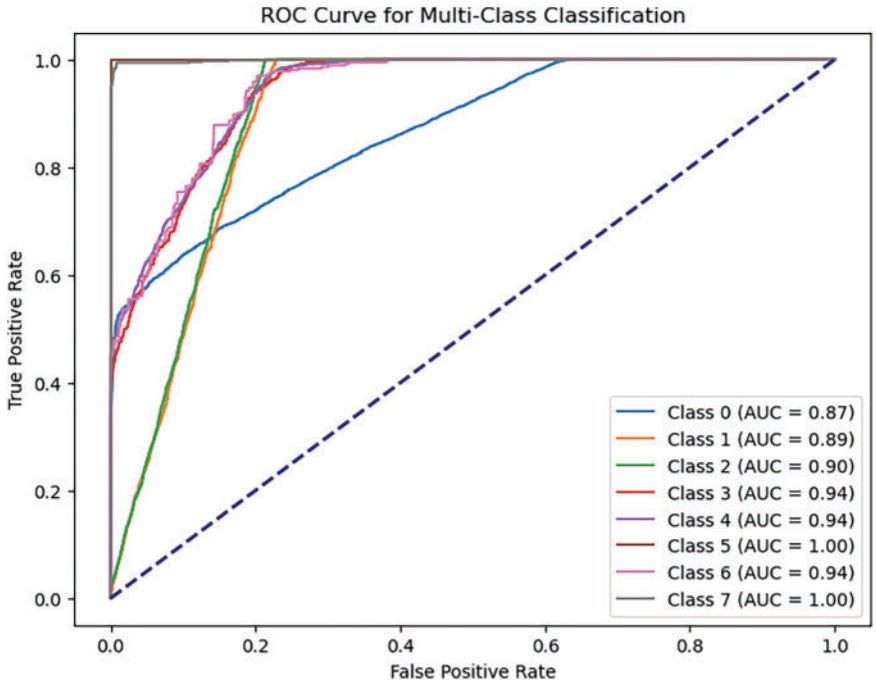



**Figure 10:** ROC curve for multi-class classification on natural gas pipeline dataset

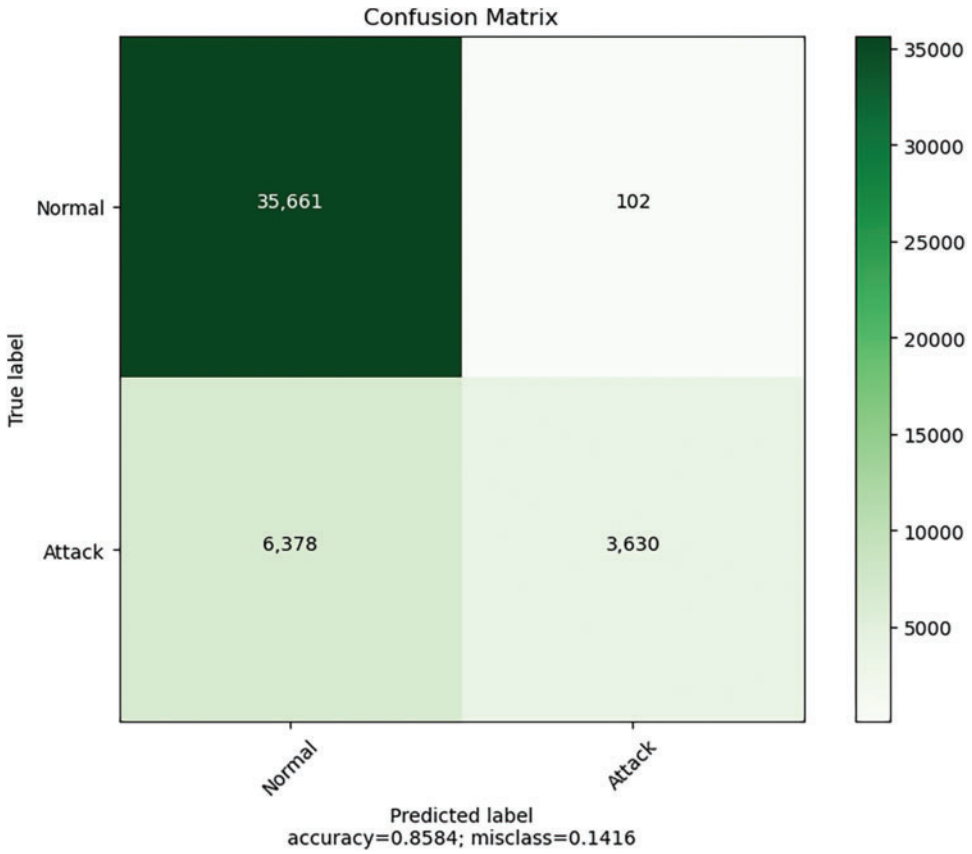



**Figure 11:** Confusion matrix for binary classification on natural gas pipeline dataset

### *5.6 Final Discussion*

This section summarizes all the processes and results of my experiments in Chapter 5. To assess the model's capabilities, clear criteria need to be established. In Section 5.1, metrics are provided to validate the model, along with explanations of their significance and calculation methods. Starting from the second section, the experimental steps are progressively elucidated. The experiments in Section 5.2 showcase the process of adjusting model parameters, including cross-validation during data preprocessing and improvements to the deep learning model based on training results. Compared to existing works in Tables 1 and 6, my model not only demonstrates superior detection performance but also can be utilized on a natural gas pipeline dataset, which is closer to the real industrial systems. The results in Section 5.5 ensure the model's practical feasibility and promising application prospects. To further validate the method's scientific validity, Section 5.4 conducts an ablation study and theoretical analysis of my method to better understand the contribution of each component to the model's performance. From a theoretical perspective, explain the role and superiority of the CNN-Bi-LSTM deep learning model in intrusion detection tasks.

## 6 Conclusion

In the realm of industrial network security, the increasing openness of ICS has led to a rise in the complexity of attacks faced by industrial networks. To tackle this challenge, this paper introduces a

deep learning-based intrusion detection method. The preprocessing of data involves the application of the SMOTE-ENN algorithm to address data imbalance issues. The model is constructed using a combination of CNN and Bi-LSTM. This amalgamation allows the model to capture both spatial and temporal features. The network structure is fine-tuned by adjusting model parameters to achieve optimal detection performance. By testing on the CICIDS-2017 dataset and comparing with other methods, it demonstrates the model's exceptional detection effectiveness. Subsequent testing on a natural gas pipeline dataset reveals the model's ability to effectively discern normal data from attacks in real industrial networks, albeit with challenges in achieving high accuracy for certain attack categories.

To improve my model, I can attempt to utilize better sample balancing techniques to address data imbalance issues in future research or apply data augmentation techniques to increase the diversity of the training dataset, thus improving the model's generalization capability.

**Acknowledgement:** The authors sincerely appreciate the support from the Liaoning Province Nature Fund Project; and Scientific Research Project of Liaoning Province Education Department.

**Funding Statement:** The authors sincerely appreciate the support from the Liaoning Province Nature Fund Project (No. 2022-MS-291); the Scientific Research Project of Liaoning Province Education Department (LJKMZ20220781, LJKMZ20220783, LJKQZ20222457, JYTMS20231488).

**Author Contributions:** The authors confirm contribution to the paper as follows: Study conception and design: J. W, C. S, Q. F; data collection: J. W, Q. F, Z. W; experiment analysis and interpretation of results: J. W, C. S, Z. W; writing—original draft preparation: J. W, C. S; writing—review and editing: J. W, C. S, Q. F. All authors reviewed the results and approved the final version of the manuscript.

**Availability of Data and Materials:** The data that support the findings of this study are available from the corresponding author upon reasonable request.

**Conflicts of Interest:** The authors declare that they have no conflicts of interest to report regarding the present study.

## References

[1] H. C. Altunay, Z. Albayrak, A. N. Özalp, and M. Çakmak, "Analysis of anomaly detection approaches performed through deep learning methods in SCADA systems," in *2021 3rd Int. Cong. Human-Comput. Intact., Optim. Robotic Appl. (HORA)*, Ankara, Turkey, IEEE, 2021, pp. 1–6.

[2] D. Bhamare, M. Zolanvari, A. Erbad, R. Jain, K. Khan and N. Meskin, "Cybersecurity for industrial control systems: A survey," *Comput. Secur.*, vol. 89, pp. 101677, 2020. doi: 10.1016/j.cose.2019.101677.

[3] K. Tange, M. de Donno, X. Fafoutis, and N. Dragoni, "A systematic survey of industrial internet of things security: Requirements and fog computing opportunities," *IEEE Commun. Surv. Tut.*, vol. 22, no. 4, pp. 2489–2520, 2020. doi: 10.1109/COMST.2020.3011208.

[4] J. Liang and Y. Kim, "Evolution of firewalls: Toward securer network using next generation firewall," in *Proc. 12th CCWC*, Las Vegas, NV, USA, 2022, pp. 752–759.

[5] L. Chen, X. Kuang, A. Xu, S. Sou, and Y. Yang, "A novel network intrusion detection system based on CNN," in *Proc. 8th CBD*, Taiyuan, China, 2020, pp. 243–247.

[6] S. Badillo *et al.*, "An introduction to machine learning," *Clin. Pharmacol. Ther.*, vol. 107, no. 4, pp. 871–885, 2020. doi: 10.1002/cpt.1796.

[7] K. G. Narayan, S. Mookherji, V. Odelu, R. Prasath, A. C. Turlapaty and A. K. Das, "IIDS: Design of intelligent intrusion detection system for internet-of-things applications," arXiv:2308.00943, 2023.

[8] R. Panigrahi *et al.*, "A consolidated decision tree-based intrusion detection system for binary and multiclass imbalanced datasets," *Mathematics*, vol. 9, no. 7, pp. 751, 2021. doi: 10.3390/math9070751.
[9] A. K. Balyan *et al.*, "A hybrid intrusion detection model using EGA-PSO and improved random forest method," *Sensors*, vol. 22, no. 16, pp. 5986, 2022.
[10] M. Aljanabi, M. A. Ismail, and A. H. Ali, "Intrusion detection systems, issues, challenges, and needs," *Int. J. Comput. Int. Sys.*, vol. 14, no. 1, pp. 560–571, 2021. doi: 10.2991/ijcis.d.210105.001.
[11] X. Gao, C. Shan, C. Hu, Z. Niu, and Z. Liu, "An adaptive ensemble machine learning model for intrusion detection," *IEEE Access*, vol. 7, pp. 82512–82521, 2019. doi: 10.1109/ACCESS.2019.2923640.
[12] D. P. Gaikwad and R. C. Thool, "Intrusion detection system using bagging ensemble method of machine learning," in *Proc. ICCUBEA-2015*, Pune, India, 2015, pp. 291–295.
[13] J. C. Huang, G. Q. Zeng, G. G. Geng, J. Weng, K. D. Lu and Y. Zhang, "Differential evolution-based convolutional neural networks: An automatic architecture design method for intrusion detection in industrial control systems," *Comput. Secur.*, vol. 132, no. 2, pp. 103310, 2023. doi: 10.1016/j.cose.2023.103310.
[14] J. C. Huang, G. Q. Zeng, G. G. Geng, J. Weng, and K. D. Lu, "SOPA-GA-CNN: Synchronous optimisation of parameters and architectures by genetic algorithms with convolutional neural network blocks for securing industrial internet-of-things," *IET Cyber-Syst. Robot.*, vol. 5, no. 1, pp. e12085, 2023. doi: 10.1049/csy2.12085.
[15] J. Kim, J. Kim, H. Kim, M. Shim, and E. Choi, "CNN-based network intrusion detection against denial-of-service attacks," *Electronics*, vol. 9, no. 6, pp. 916, 2020. doi: 10.3390/electronics9060916.
[16] M. Almiani, A. AbuGhazleh, A. Al-Rahayfeh, S. Atiewi, and A. Razaque, "Deep recurrent neural network for IoT intrusion detection system," *Simul. Model. Pract. Theor.*, vol. 101, pp. 102031, 2020. doi: 10.1016/j.simpat.2019.102031.
[17] A. A. Süzen, "Developing a multi-level intrusion detection system using hybrid-DBN," *J. Ambient Intell. Humaniz Comput.*, vol. 12, no. 2, pp. 1913–1923, 2021. doi: 10.1007/s12652-020-02271-w.
[18] C. Tang, N. Luktarhan, and Y. Zhao, "An efficient intrusion detection method based on lightGBM and autoencoder," *Symmetry*, vol. 12, no. 9, pp. 1458, 2020. doi: 10.3390/sym12091458.
[19] B. Cao, C. Li, Y. Song, Y. Qin, and C. Chen, "Network intrusion detection model based on CNN and GRU," *Appl. Sci.*, vol. 12, no. 9, pp. 4184, 2022. doi: 10.3390/app12094184.
[20] T. Su, H. Sun, J. Zhu, S. Wang, and Y. Li, "BAT: Deep learning methods on network intrusion detection using NSL-KDD dataset," *IEEE Access*, vol. 8, pp. 29575–29585, 2020. doi: 10.1109/ACCESS.2020.2972627.
[21] P. Sun *et al.*, "DL-IDS: Extracting features using CNN-LSTM hybrid network for intrusion detection system," *Secur. Commun. Netw.*, vol. 2020, pp. 1–11, 2020.
[22] H. C. Altunay and Z. Albayrak, "A hybrid CNN+LSTM-based intrusion detection system for industrial IoT networks," *Eng. Sci. Technol.*, vol. 38, pp. 101322, 2023. doi: 10.1016/j.jestch.2022.101322.
[23] Y. Chen, H. Jiang, C. Li, X. Jia, and P. Ghamisi, "Deep feature extraction and classification of hyperspectral images based on convolutional neural networks," *IEEE Trans. Geosci. Remote Sens.*, vol. 54, no. 10, pp. 6232–6251, 2016. doi: 10.1109/TGRS.2016.2584107.
[24] H. Gholamalinezhad and H. Khosravi, "Pooling methods in deep neural networks, a review," arXiv:2009.07485, 2020.
[25] A. M. Schaefer, S. Udluft, and H. G. Zimmermann, "Learning long-term dependencies with recurrent neural networks," *Neurocomputing*, vol. 71, no. 13–15, pp. 2481–2488, 2008. doi: 10.1016/j.neucom.2007.12.036.
[26] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: LSTM cells and network architectures," *Neural Comput.*, vol. 31, no. 7, pp. 1235–1270, 2019. doi: 10.1162/neco_a_01199.
[27] F. Zou, H. Zhang, F. Sang, X. Li, W. He and X. Liu, "Bearing fault diagnosis based on combined multi-scale weighted entropy morphological filtering and Bi-LSTM," *Appl. Intell.*, vol. 51, no. 10, pp. 6647–6664, 2021.
[28] R. Panigrahi and S. Borah, "A detailed analysis of CICIDS2017 dataset for designing intrusion detection systems," *Int. J. Eng. Technol.*, vol. 7, no. 3.24, pp. 479–482, 2018.

[29] M. N. Goryunov, A. G. Matskevich, and D. A. Rybolovlev, "Synthesis of a machine learning model for detecting computer attacks based on the CICIDS2017 dataset," in *Proc. ISP of the RAS*, vol. 32, no. 5, pp. 81–94, 2020. doi: 10.15514/ISPRAS-2020-32(5)-6.

[30] D. Stiawan, M. Y. B. Idris, A. M. Bamhdi, and R. Budiarto, "CICIDS-2017 dataset feature analysis with information gain for anomaly detection," *IEEE Access*, vol. 8, pp. 132911–132921, 2020. doi: 10.1109/ACCESS.2020.3009843.

[31] F. Salo, M. Injadat, A. B. Nassif, A. Shami, and A. Essex, "Data mining techniques in intrusion detection systems: A systematic literature review," *IEEE Access*, vol. 6, pp. 56046–56058, 2018. doi: 10.1109/ACCESS.2018.2872784.

[32] I. P. Turnipseed, *A New Scada Dataset for Intrusion Detection Research.* Mississippi, USA: Mississippi State University ProQuest Dissertations Publishing, 2015, pp. 10–14.

[33] A. K. Rastogi, N. Narang, and Z. A. Siddiqui, "Imbalanced big data classification: A distributed implementation of smote," in *Proc. Workshop Program 19th Int. Conf. Distrib. Comput. and Netw.*, Varanasi, India, 2018, pp. 1–6.

[34] M. M. R. Khan, R. B. Arif, M. A. B. Siddique, and M. R. Oishe, "Study and observation of the variation of accuracies of KNN, SVM, LMNN, ENN algorithms on eleven different datasets from UCI machine learning repository," in *Proc. 4th iCEEiCT*, Dhaka, Bangladesh, 2018, pp. 124–129.

[35] A. Puri and M. Kumar Gupta, "Improved hybrid bag-boost ensemble with K-means-SMOTE-ENN technique for handling noisy class imbalanced data," *Comput. J.*, vol. 65, no. 1, pp. 124–138, 2022. doi: 10.1093/comjnl/bxab039.

[36] X. Deng, Q. Liu, Y. Deng, and S. Mahadevan, "An improved method to construct basic probability assignment based on the confusion matrix for classification problem," *Inf. Sci.*, vol. 340, no. 1, pp. 250–261, 2016. doi: 10.1016/j.ins.2016.01.033.

[37] O. Belarbi, A. Khan, P. Carnelli, and T. Spyridopoulos, "An intrusion detection system based on deep belief networks," in *Proc. SciSec 2022*, Cham, Springer International Publishing, 2022, vol. 13580, pp. 377–392.

[38] Y. Yao, L. Su, and Z. Lu, "DeepGFL: Deep feature learning via graph for attack detection on flow-based network traffic," in *Proc. MILCOM*, Los Angeles, CA, USA, 2018.

[39] M. Roopak, G. Yun Tian, and J. Chambers, "Deep learning models for cyber security in IoT networks," in *Proc. 9th CCWC*, Las Vegas, NV, USA, 2019, pp. 452–457.