Industrial Internet of Things ARP Virus Attack Detection Method Based on Improved CNN BiLSTM

Jianhua Wang

Northwest Minzu University, Lanzhou, Gansu 730030, China
E-mail: 13809312825@163.com

Received 28 April 2024; Accepted 16 June 2024

Abstract

In order to improve the performance of industrial Internet of Things ARP virus attack detection methods, this paper proposes an improved CNN BiLSTM based industrial Internet of Things ARP virus attack detection method. Firstly, analyze the data flow of normal data, construct an industrial Internet of Things ARP virus intrusion dataset, and obtain the sample distribution of the ETI dataset. Secondly, based on the domain knowledge of ETCN, a preliminary manual selection was performed on all extracted head features, and a feature correlation discrimination algorithm was designed to further screen the features. Then, the Pearson correlation coefficient is used to calculate its linear correlation, the distance correlation coefficient is used to calculate its nonlinear correlation, and a comprehensive calculation formula is designed based on the principle of "maximum correlation and minimum redundancy" to establish a comprehensive measurement coefficient. The value of the features selected in the first stage is ranked using this coefficient, and different feature subsets are constructed through sequential search. Effective features are selected based on the performance of the intrusion detection models trained on different feature subsets. Implement industrial Internet of Things (IoT) ARP feature extraction through feature extraction, data cleaning, feature transformation, and feature selection. Finally, an improved CNN BiLSTM structure is constructed by using CNN to filter out a large number of packets that are not related to the attack and have weak correlation in the data. Significant features are extracted from the data, and the feature data extracted by CNN is timestamped through timeDistribution. After flattening into one-dimensional data through the flat layer, it is used as input to the BILSTM layer. We used a bidirectional long short-term memory network (BILSTM) to train industrial IoT ARP virus attacks and output the final ARP virus attack detection results. The experimental results show that in the first 10 rounds of training, the training accuracy and validation accuracy of the model rapidly increase, indicating that the model learns a large amount of information in this stage of iteration. We achieved high F1 score (94.42%), high accuracy (94.58%), and low false alarm rate (5.33%) on the ETI dataset. The model consumed very little training time (8.0746s) and testing time (0.1664s). Verified the effectiveness of the design model.

Keywords: CNN, BILSTM, pearson correlation coefficient, sequential search, TimeDistribute, bidirectional long short term memory network.

1 Introduction

With the concept of integration of industrialization and industrialization, intelligent manufacturing and other concepts put forward, more and more industrial control systems that originally did not access the network and had low network security protection began to access the Internet, allowing hackers to directly attack industrial control systems through the Internet [1]. Due to the widespread application of industrial control systems in various fields, including infrastructure, people's livelihoods, industrial production, and military industry, once invaded, they face not only problems such as information leakage and unusable information equipment, but also substantial impacts on the real world where people are located, such as equipment failures, environmental pollution, casualties, and even endangering national security. The consequences are unpredictable, so protecting industrial control systems from hacker attacks is crucial [2, 3]. The use of intrusion detection technology is an effective method to promptly detect the intrusion of industrial control systems. Because ARP attacks are initiated by exploiting the vulnerabilities of ARP itself, with high success rates and strong concealment, many defense and detection methods for ARP attacks are achieved by modifying ARP or using static ARP tables [4, 5].

Wu Lian et al. proposed a computer virus detection method based on lightweight deep networks [6], collecting datasets including virus samples and normal file samples and labeling them. The quality and quantity of the dataset have a significant impact on the performance of deep learning models. Using lightweight deep networks to extract features from files, with the aim of extracting abstract features that can represent the content of the file. Design a lightweight deep learning model using convolutional neural networks (CNNs), calculate complexity using pruning techniques, train the designed deep network model using a dataset, and continuously optimize model parameters using loss functions and optimization algorithms to improve the model's generalization ability and accuracy. The characteristics of computer viruses are diverse, and some viruses even have variability. How to design effective feature extraction methods to extract key features that can identify viruses is a challenge.

Yang Shixin et al. proposed a ransomware detection method for industrial devices based on the integration of Android motion and static [7]. By monitoring the dynamic behavior of Android applications running on industrial devices, including file operations, network communication, system calls, etc., abnormal behavior can be detected, which may involve sandbox environments or virtualization technologies to simulate device environments. Perform static feature extraction on Android applications on industrial devices, including permission requests, code structures, API call sequences, etc., to identify potential malicious behavior features. Integrate the results of dynamic behavior analysis and static feature extraction, comprehensively consider the characteristics of the application during runtime and static state, and improve detection accuracy and robustness. Based on fusion analysis of features, design and establish detection models, which can use machine learning methods such as support vector machines (SVM), decision trees, or deep learning methods such as convolutional neural networks (CNN), recurrent neural networks (RNN), etc. Train the established detection model using labeled datasets and optimize the model through cross validation and other methods to improve its generalization ability and detection accuracy. Deploy the trained model onto industrial equipment for practical detection, and verify the performance and usability of the model through real-world experiments. Based on feedback information from practical applications, continuously improve and update the detection model to adapt to new ransomware variants and attack techniques. When conducting dynamic behavior analysis, it is necessary to consider the sensitive data and privacy information that may be involved in industrial equipment, to ensure that detection methods do not infringe on user privacy rights or increase system security risks. Ying Xianer et al. proposed a virus propagation network intrusion detection research based on graph neural networks [8], which transforms nodes, edges, and their relationships in the network into corresponding vector representations, and transforms network data into input forms suitable for graph neural networks. Design graph neural network structures suitable for virus transmission networks, such as Graph Convolutional Neural Network (GCN), Graph Attention Network (GAT), etc., as well as appropriate layers and parameter settings for graph neural networks. By modeling and simulating the propagation process of viruses in the network, a training dataset with labeled information is generated for supervised learning. Use graph neural networks for supervised learning of virus propagation networks to learn the characteristics of normal and abnormal behavior in network structures, and predict potential network intrusion situations. Evaluate the trained model using methods such as cross validation, adjust the network structure and hyperparameters to optimize detection performance. Apply the trained model to actual networks, monitor network conditions in real time, and detect and prevent virus transmission and other network intrusion behaviors in a timely manner. The virus transmission network data may be sparse, and some nodes and edges may have missing information. How to handle sparse graph data is a challenge. Obtaining accurate label information is crucial for model training, and for virus transmission networks, obtaining label information may face difficulties. Yuan Huihua proposed a research on anomaly threat detection based on network full traffic analysis technology [9], which collects all data traffic in the network, including packet, session and other information, to ensure the acquisition of complete network communication information. Preprocess the collected network traffic data, such as removing noise, normalizing, etc., and then extract effective features such as source IP address, target IP address, port number, protocol type, etc. Choose a machine learning model to model and learn traffic data. Train the model using the collected traffic data to determine whether the traffic is abnormal based on feature information, and then identify potential threat behaviors. Apply the trained model to actual traffic data, detect abnormal data streams in real-time traffic, and identify potential threat behaviors. When the model detects abnormal traffic, the system should promptly issue an alarm and take corresponding security measures, such as isolating devices, tracking attackers, etc. Obtaining accurate label information (normal/abnormal traffic) for supervised learning is a challenge that may require expert knowledge and manual annotation, with high costs.

In response to the above issues, this article proposes an industrial Internet of Things ARP virus attack detection method based on improved CNN-BiLSTM, constructs an improved CNN-BiLSTM structure, uses CNN to filter out packets that are not related to the attack and have weak correlation in the data, extracts significant features in the data, labels them with time labels through timeDistribution, flattens them into one-dimensional data through the flat layer, and uses them as input to the BILSTM layer. We used a bidirectional long short-term memory network (BILSTM) to train industrial IoT ARP virus attacks and output the final ARP virus attack detection results, effectively improving the detection efficiency of industrial IoT ARP virus attacks.

2 ARP Feature Extraction Method for Industrial Internet of Things

2.1 Intrusion Dataset Construction

For ease of description, the ETCN intrusion dataset constructed in this section is referred to as the ETI (Ethernet Train In intrusion) dataset. The process of building an ETI dataset includes data generation and collection, where the data generation part can be further divided into the generation of normal data and the generation of attack data.

In the normal data generation section, it is necessary to configure a normal network communication process in the semi physical experimental platform to generate corresponding normal communication data. This section refers to the network communication configuration information of a certain type of actual train, and adds some customized configuration content within the standard range (such as different communication cycles, different data lengths, etc.). Normal communication data is configured between the 12 terminal devices on the experimental platform. Specifically, configuration is mainly carried out from the aspects of data flow, data protocol type, communication cycle, communication identifier, data content, etc. [10].

In terms of data flow, refer to the data flow of the actual vehicle for configuration, as shown in Figure 1. ED1 simulates the data flow of the central control unit, which communicates with other terminal devices. ED2 and ED3 simulate the data flow of the traction and braking terminal, which communicates with ED1 and ED11 and ED12, which simulate the data flow of the event recorder. Other terminal nodes simulate the data flow of other terminal devices in the train. In terms of data protocol types, TRDP push

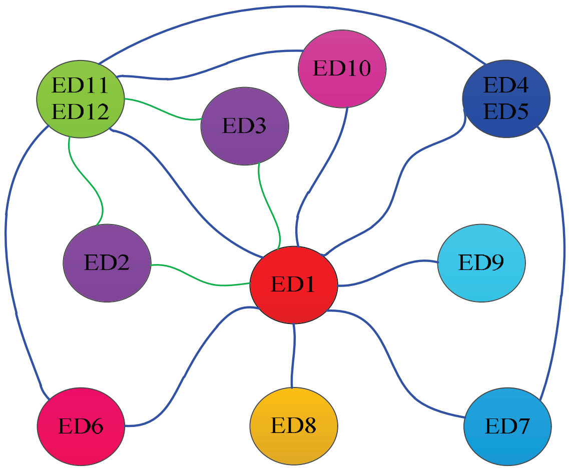


Figure 1 Data flow of normal data.

mode process data, TRDP pull mode process data, UDP data, TCP data, ARP data, ICMP data, etc. have been configured. Among them, TRDP push mode process data is the most important data protocol type for ETCN, so it is configured as the main communication data in the network, with the maximum communication volume [11]. In terms of communication cycle, the data transmitted periodically in the network is mainly TRDP push mode process data. The process data cycle between ED2, ED3 and ED1, ED11, and ED12 is configured as 32 ms or 48 ms (simulating the short cycle situation of traction and braking control part), and the process data cycle between other terminal devices is randomly configured as a time value that can be divided by 5 between 100 ms and 200 ms; In terms of communication identifiers (process identification parameters of TRDP), configure the communication identifiers of different processes with values between 1001 and 1099 (the protocol stipulates that communication identifiers between 1-999 are reserved for some exclusive purposes, so values greater than 999 are taken); In terms of data content, generate random content with different numerical values ranging from 64 to 500 as data content [12].

In the attack data generation section, a high-performance embedded development board and two portable computers are used as three attack simulation terminals to inject all thirteen typical attacks analyzed earlier into the network. Both the development board and computer are loaded with Nessus software and ETCN penetration testing software. Nessus software is used to simulate vulnerability scanning attacks, while ETCN penetration testing software is used to simulate twelve other typical attacks. This software was jointly developed by the authors and members of the research group. It is designed and implemented based on the Scapy library and the attack mechanism of ETCN typical attacks obtained through analysis. It can simulate other ETCN typical attacks besides vulnerability scanning analyzed in this chapter (vulnerability scanning requires the support of a large vulnerability database, which is achieved through mature Nessus software) [13]. In the attack simulation experiment, we chose three specific attack types for simulation and detection, instead of simulating more attack types, mainly for the following considerations:

First of all, these three attack types (Probe, DoS, MITM) are highly representative and potentially harmful in the industrial iot environment. Probe class attacks, such as address scanning and port scanning, are commonly used to probe network topology and potential security vulnerabilities, and are a necessary step before hacking. A DoS attack, such as a distributed denial-of-service attack, overloads the target system by sending a large number of requests or packets, rendering it unable to provide service to normal users, and for business-critical systems, this attack can lead to severe service disruptions. Mitm-class attacks (such as man-in-the-middle attacks) can steal sensitive information or destroy the integrity of communication by intercepting and tampering with communication data, which is crucial for protecting data security and communication privacy.

Second, while there are many other types of cyberattacks, we focused on these three types in this experiment because they cover the main dimensions and characteristics of cyberattacks. Probe attacks focus on information collection, DoS attacks focus on resource consumption, and MITM attacks focus on data tampering and theft. By simulating these three types of attacks, we can evaluate and improve our detection model more comprehensively, while avoiding experimental complexity and data redundancy caused by too many attack types.

In the specific simulation process, in order to increase the diversity of attack data within the class, we take different strategies. For Probe attacks, we adopt traversal address and port scanning to simulate the behavior of an attacker in detecting each terminal node in the network. For class DoS attacks, we configure the stepped flooding intensity and use randomly generated source IP addresses to simulate the characteristics of distributed denial of service attacks. For MITM attacks, we configured different header tampering and payload tampering schemes, and randomly configured replay lag time in replay attacks to simulate the attacker's behavior of destroying

Table 1 Sample distribution of ETI dataset

	Category	Number of Samples	Proportion/%
Normal	Normal	551734	51.45
Probe type attacks	Address scanning	37492	3.50
	Port scanning	39218	3.66
	Vulnerability scanning	38507	3.59
DoS type attacks	TRDP Push Flooding	42135	3.93
	TRDP Full Flooding	41960	3.91
	UDP flooding	41755	3.89
	TCP flooding	42349	3.95
	MAC flooding	40280	3.76
	Smurf flooding	41084	3.83
MITM type attacks	ARP spoofing	38161	3.56
	First tampering	39624	3.69
	Payload tampering	40125	3.74
	replay	37972	3.54

communication data [14]. By simulating these three representative and potentially harmful attack types and increasing the intra-class diversity of attack data, it can evaluate the performance of the detection model more comprehensively and accurately, and provide strong support for its application in the real industrial IoT environment.

In the process of attack simulation, network data is mirrored and aggregated to the data acquisition terminal through the port image of the switch. Wireshark software is used to grab network data packets at the data acquisition terminal, and continuously allocate sample labels to them to obtain an ETI data set with a total sample size of 1072396, including three types of attack samples, a total of 13 attack samples. The sample distribution of the dataset is shown in Table 1. After completing the method research in the following chapters, the ETI data set can be used for intrusion detection experiments to evaluate its performance and verify its effectiveness.

It should be noted that in order to make the constructed dataset have a balanced sample distribution (in the evaluation experiment of the intrusion detection model based on machine learning, the uneven sample distribution will affect the accuracy of the evaluation), the frequency and duration of different attacks are adjusted in the attack simulation process, so that the number of attack samples in the ETI dataset is close to the number of normal samples (about 1:1), and the number of samples for different types of attacks is close. In addition, unlike ETCN network failures and other issues, the

ETCN network attacks analyzed and summarized earlier mainly exploit the vulnerability issues of different network protocols, such as stealing relevant information, disrupting communication processes, and tampering with communication content. When a device is attacked, its attack effect occurs at the logical level of network communication, specifically manifested as changes in the data characteristics of network packets in the network layer, transmission layer, and TRDP layer, as well as temporal changes between packets, which are not affected by the operating system and hardware composition of the device [15]. Therefore, the differences between the operating system and hardware structure of the ETCN semi physical experimental platform built and the actual train terminal equipment will hardly affect the effectiveness of the network data collected in the attack simulation experiment.

2.2 Industrial Internet of Things ARP Feature Processing Design

The essence of machine learning is to learn the patterns and knowledge contained in data. For intrusion detection tasks based on machine learning, the network behavior patterns reflected by network packets are inherent, and the corresponding detection algorithms are the process of approximating and searching for this behavior pattern in the data space. This process is achieved by designing and training detection models. In the intrusion detection task of this article, in order to present the information embedded in the data in a more standardized and effective way, it is necessary to first preprocess the original ETCN network data before training the detection model, that is, carry out feature engineering.

Feature engineering is a process that utilizes different methods and combines relevant knowledge in the field where the data is located to improve the quality of the original data, enabling it to meet the formal requirements of detection algorithms, and selecting and extracting data features from the original data that better represent business logic. This chapter conducts feature engineering method design from four parts, namely feature extraction, data cleaning, feature transformation, and feature selection.

(1) Feature extraction

The features reflect the information carried by the data samples, and extracting and selecting reasonable and effective features is the key to successfully carrying out a learning task. Feature extraction refers to the process of extracting sample features from raw data [16]. The ETCN data packet mainly consists of a header and a payload. As the header features contain the vast

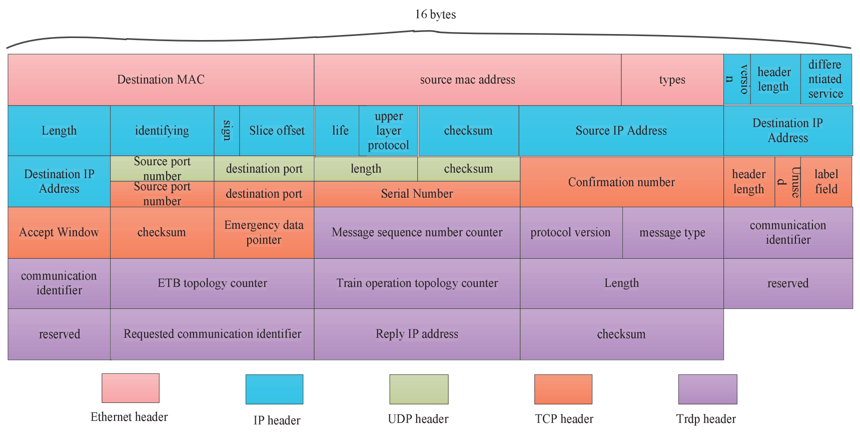


Figure 2 Head features of TRDP process data packets.

majority of key information that can be used for network behavior analysis in the network data packet, all the features extracted from the header of the data packet are used as the preliminary sample features extracted in this section of feature engineering; The data content in the payload has a wide range of changes and extremely high uncertainty. In intrusion detection tasks, the noise it carries far exceeds its effective information. The features extracted from it often lead to overfitting in intrusion detection, suppressing the generalization and universal detection ability of intrusion detection. Therefore, features are not extracted from it.

Taking the core TRDP process packets in ETCN as an example, the header features that can be extracted include: Ethernet header features, IP header features, UDP/TCP header features, and TRDP header features, as shown in Figure 2 (UDP and TCP are peer-to-peer protocols at the transport layer, and their features do not appear in the same packet at the same time, so the UDP header and TCP header are depicted in parallel above and below in the figure). Except for TRDP process packets, ICMP packets, ARP packets, etc. all have their own unique header features, which will not be elaborated here.

(2) Data cleaning

Data cleaning is an important component of feature engineering, which is mainly used to improve the data quality of raw data and does not involve information extraction from raw data. The main content of data cleaning includes handling duplicate data, handling outlier data, and handling missing values [17]. Due to the fact that network packet capture is a data collection method with a certain degree of certainty, there is generally no duplicate or outlier data in the captured data packets, and only missing value processing is required.

The mutual exclusion between some protocol types can lead to missing values in the sample features.

For example, since UDP and TCP are mutually exclusive, the header information of these two protocols will not appear in the same sample at the same time. Therefore, in samples with protocol type UDP, the value of the TCP feature is null, indicating the presence of missing values. The processing methods to be adopted vary depending on the reasons for missing values. The missing values described here belong to the non random missing type. When dealing with non random missing values, zero imputation technique is often used to process them, that is, "0" is used to fill in the missing numerical features [18]. However, due to the fact that some features in the ETCN network samples indirectly contain feature information with 0 values, that is, if the feature value of a certain sample is 0, it indicates that the sample covers the protocol header corresponding to that feature. Directly filling in missing values with 0 values can cause feature ambiguity and introduce noise into the sample. Therefore, this section combines the characteristics of ETCN intrusion samples to adaptively improve the zero padding technique, using "-1" to fill in the missing feature values in the samples. The advantage of this improvement is that it can fill in the missing values without introducing new noise to the samples.

(3) Feature transformation

Feature transformation is mainly used to transform sample features into a form that can be effectively processed by the detection model, including feature type transformation and feature standardization. The ETCN network samples contain three types of features: categorical features, such as protocol types, etc; String type features, such as source IP address, destination IP address, etc; Numeric features, such as UDP source port number, total packet length, etc.

For categorical features, the intrusion detection method proposed in this chapter can be directly processed without the need for feature transformation, which is also a manifestation of the ease of use of the intrusion detection method in this chapter. For string type features, they are converted into category type features by corresponding them one by one, that is, each set of strings with the same content corresponds to a feature category. For numerical features, due to the significant differences in numerical ranges between different numerical features in ETCN intrusion samples, in order to balance the differences between large and small range features in mathematical space and avoid unreasonable threshold shifts during model learning, a normalization method is adopted to standardize numerical features, which maps feature values to between 0 and 1. The calculation method is as follows:

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

Among them, x_{std} is the standardized feature value, x is the original feature value, and $x_{\max}$ and $x_{\min}$ are the maximum and minimum values of the feature, respectively.

(4) Two-stage feature selection method

Thanks to the automatic learning of intrinsic patterns from data by machine learning, especially deep learning, all extracted features can be directly used as input for intrusion detection models after data cleaning and feature transformation. However, the degree to which different features reflect network behavior patterns varies, and the information that different features can provide in intrusion detection tasks is also different. As a special network system, ETCN network samples have unique features of ETCN, namely the head features of TRDP protocol. At the same time, some of the head features of protocols in traditional information networks do not appear in ETCN packets. In this case, new features are added and old features are reduced, further selection of the extracted features is needed, that is, selecting features that can provide effective information in intrusion detection. At the same time, due to the higher number of features, i.e. the higher the feature dimension, the lower the training and detection speed of the intrusion detection model. Therefore, to improve detection efficiency, it is necessary to simplify the number of features as much as possible and select key features that can provide more information.

In summary, for ETCN network data, this section proposes a two-stage feature selection method. In the first stage, manual feature selection is based on ETCN domain knowledge, and in the second stage, a feature selection algorithm based on correlation discrimination is designed.

(1) Phase 1

At this stage, combining the domain knowledge of ETCN to perform preliminary manual selection on all extracted head features can quickly eliminate

Table 2 Features obtained through manual selection in the first stage

Serial Number	Feature Name	Serial Number	Serial Number
1	Protocol	21	ICMP identification
2	Total packet length	22	ICMP serial number
3	Destination MAC	23	UDP source port number
4	source mac address	24	UDP destination port number
5	Network layer protocol type	25	UDP LENGTH
6	IP header length	26	TCP source port number
7	Differentiated service domain	27	TCP destination port number
8	IP length	28	TCP serial number
9	IP identification	29	TCP confirmation number
10	IP flag	30	TCP header length
11	life	31	TCP flag field
12	Source IP Address	32	TCP receive window
13	Destination IP Address	33	Trdp serial number
14	ARP operation type	34	Trdp message type
15	Sender MAC address	35	Trdp communication identifier
16	Sender IP address	36	ETB topology counter
17	destination mac address	37	Train operation topology counter
18	Destination IP address	38	Trdp length
19	ICMP type	39	Requested communication identifier
20	ICMP code	40	Reply IP address

features that have little impact on detection performance, no impact on detection performance, and may have negative impacts. Starting from all the extracted features, remove features that have little impact on network communication, no differences between different data packets, or contain high noise, such as IP protocol version number (currently ETCN still uses IPv4 version, so the information entropy of this feature is small and the amount of information contained is small), and the sum of each head (as long as there is a numerical difference, different sums of tests may be obtained, so the information noise of this feature is too large), and preliminarily obtain the manually selected features shown in Table 2.

Among the extracted features, the 1st to 32nd dimensional features are common features of TCP/IP protocol clusters, and the 33rd to 40th dimensional features are TRDP header features unique to ETCN. For network attack packets, these features may undergo relatively significant changes. For example, in TRDP Push flooding attacks, the TRDP communication identifier (ComID) may change; In ARP spoofing attacks, the combination relationship between the sender's IP address and the sender's MAC address may change, and so on.

(2) Phase 2

At this stage, a feature correlation discrimination algorithm was designed to further screen the features in Table 2. This algorithm decomposes the correlation between features and features, as well as between features and categories, in mathematical space. Linear correlation is calculated using Pearson correlation coefficient, and nonlinear correlation is calculated using distance correlation coefficient. A comprehensive calculation formula is designed based on the principle of "maximum correlation and minimum redundancy" to establish a comprehensive measurement coefficient. The value of the features selected in the first stage is ranked based on this coefficient, and different feature subsets are constructed through sequential search. Effective features are selected based on the performance of the intrusion detection models trained on different feature subsets. The core concept of this algorithm is: firstly, based on the comprehensive consideration of linear and nonlinear correlations in mathematical space, features with high correlation between incentives and category labels, and features with high redundancy between penalties and other features are designed, and a comprehensive measurement coefficient is designed to integrate incentives and punishments; Secondly, based on the ranking of the comprehensive measurement coefficients, a sequential search is performed on the feature subsets. By comparing the performance of intrusion detection models trained on different levels of feature subsets, the selection of effective features is completed. The detailed steps of the algorithm are:

Step 1. Due to the need for numerical mathematical operations, before starting the calculation, use the feature transformation method to digitize and normalize all sample features; In addition, for category labels, map them sequentially from 0 to decimal values.

Step 2. Set the total sample size of the dataset to n and the feature dimension to m. For the linear correlation part, traverse and calculate the Pearson correlation coefficient between features $p_{XX'}$:

$$p_{XX'} = \frac{\sum_{i=1}^{n}(X_i - \overline{\mathrm{X}})(X'_i - \overline{\mathrm{X}'})}{\sqrt{\sum_{i=1}^{n}(X_i - \overline{\mathrm{X}})^2}\sqrt{\sum_{i=1}^{n}(X'_i - \overline{\mathrm{X}'})^2}} \tag{2}$$

Among them, X and X' are the two features involved in the calculation, X_i and X_i' respectively refer to the ith value of features X and X', and $\overline{X}$ and $\overline{X}'$ respectively refer to the mean of features X and X'.

After completing the traversal calculation, a Pearson correlation matrix R_p is obtained, which contains the Pearson correlation coefficient between any two features in the dataset (the diagonal of the matrix represents the correlation coefficient between a certain feature and itself, therefore it is always 1). It can be formalized as:

$$R_p = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1m} \\ p_{21} & p22 & \cdots & p_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ p_{m1} & p_{m2} & \cdots & p_{mm} \end{bmatrix}. \tag{3}$$

When two features are highly dependent, removing one of them has a relatively small impact on the detection ability of the intrusion detection algorithm, which means that these two features are considered redundant to each other. Therefore, this algorithm defines the average correlation between a certain feature and other features as redundancy, and calculates the linear redundancy degree $r_p(k)$ between the k and dimensional feature and other features:

$$r_p(k) = \omega \cdot \sum_{l=1}^{m-1} P_{kl} \tag{4}$$

Among them, ω is the redundancy degree weight. Since redundant features do not contribute to the detection ability completely, this weight can be used to adjust the penalty level of the feature selection algorithm for redundant features in this section.

Step 3. Calculate the Pearson correlation coefficient p_{XY} between all features and category labels, using Equation (2) as the calculation formula. Simply replace X in equation with sample label Y.

Step 4. For the nonlinear correlation part, traverse and calculate the distance correlation coefficient between features. Firstly, calculate the distance matrices a_{ij} and b_{ij} between the elements of X and X':

$$a_{ij} = \|X_i - X_j\| \tag{5}$$

$$b_{ij} = \|X_i' - X_j'\| \tag{6}$$

Among them, ij $= 1, 2 \ldots, n$ and n represent the total number of samples in the dataset, and $\|\cdot\|$ represents the calculation of Euclidean norm.

Secondly, calculate the center distance matrices A_{ij} and B_{ij}:

$$A_{ij} = a_{ij} - \overline{a}_{i\cdot} - \overline{a}_{\cdot j} + \overline{a}_{\cdot\cdot} \tag{7}$$

$$B_{ij} = b_{ij} - \overline{b}_{i\cdot} - \overline{b}_{\cdot j} + \overline{b}_{\cdot\cdot} \tag{8}$$

Among them, $\overline{a}_{i\cdot}$ is the mean of line i in a_{ij}, $\overline{a}_{\cdot j}$ is the mean of column j in a_{ij}, $\overline{a}_{\cdot\cdot}$ is the mean of all elements in a_{ij}, and so on up to $\overline{b}_{i\cdot}\overline{b}_{\cdot j}\overline{b}_{\cdot\cdot}$.

Again, calculate the distance covariance $dCov_n^2(X, X')$ between X and X':

$$dCov_n^2(X, X') = 1/n^2 \sum_{i,j=1}^{n} A_{ij}B_{ij} \tag{9}$$

On the basis of Equation (9), calculate the distance variances $dVar_n^2(X)$ and $dVar_n^2(X')$ for X and X', respectively:

$$dVar_n^2(X) = dCov_n^2(X, X) \tag{10}$$

$$dVar_n^2(X') = dCov_n^2(X', X') \tag{11}$$

Finally, calculate the distance correlation coefficient $d_{XX'}$:

$$d_{XX'} = \frac{dCov_n^2(X, X')}{\sqrt{dVar_n^2(X) \cdot dVar_{\mathrm{n}}^2(X')}} \tag{12}$$

According to Equation (12), traverse all features to obtain a distance correlation matrix R_d, which is expressed in the same form as Equation (3). Based on R_d, further calculate the non-linear feature redundancy $r_d(k)$ between the kth dimensional feature and other features. The calculation process follows Equation (4), and simply replace p_{kl} with d_{kl} in Equation (4) to obtain $r_d(k)$.

Step 5. Calculate the distance correlation coefficient d_{XY} between all features and category labels. Using the calculation method in Step 4, simply replace X' with sample label Y.

Step 6. Following the principle of “maximum correlation and minimum redundancy”, which aims to maximize the correlation between features and category labels in the ETCN intrusion dataset and minimize the redundancy between their features, a comprehensive formula is designed to weight and synthesize the results of the linear and nonlinear parts mentioned above. The formula is:

$$v(k) = h \cdot (d_{kY} - r_d(k)) + (1 - h) \cdot (p_{kY} - r_p(k)) \quad k = 1, 2, \ldots, m \tag{13}$$

Among them, $v(k)$ is the comprehensive measurement coefficient of the kth dimensional feature finally obtained, $r_p(k)$ and $r_d(k)$ are the linear redundancy and nonlinear redundancy of the kth dimensional feature obtained in Step 2 and Step 4, p_{kY} and p_{kY} are the Pearson correlation coefficient and distance correlation coefficient of the kth dimensional feature and category label obtained in Step 3 and Step 5, and k is the nonlinear weight, which is used to adjust the proportion of linear part and nonlinear part in the comprehensive measurement.

Step 7. Because TRDP data packet is the key data packet in etcn, most of the control instructions in the vehicle need to be transmitted in the form of TRDP process data, and its characteristics are of high importance. Therefore, when calculating the $v(k)$ of TRDP related features, a higher weight coefficient is specially assigned to it, i.e. $v(k)$ times 1.2 of TRDP features as its final $v(k)$.

Step 8. Rank the comprehensive measurement coefficient $v(k)$ of all features of etcn network data from large to small. The larger the value of $v(k)$, the more important the krd Witt sign is. In the first stage, 40 dimensional features have been obtained by manual selection. The first 20 dimensional features sorted by $v(k)$ in this stage are used as the starting point to build a feature subset, and the remaining features are added to the subset in order to obtain a new feature subset (21 feature subsets can be obtained). The obtained feature subset is used as the training set to train the intrusion detection model in this chapter, and select the feature subset that can make the model achieve the best comprehensive performance. The features contained in the subset are the effective features obtained by screening.

As described in Step 8, the feature selection results of the second stage need to be obtained through experimental comparison.

3 ARP Virus Attack Detection Method for Industrial Internet of Things Based on Improved CNN BILSTM

According to the ARP attack principle, if hackers want to launch ARP attacks, they need to continuously send false ARP response messages to the attacked equipment, which requires hackers to use software to attack. The attack software will send the forged ARP response message at a fixed interval each time, which makes the ARP attack message show the characteristics of periodic change in time. Because the message data in the network is generated by all the devices in the network, the number of messages in the data is huge and various, and it also has the characteristics of concurrency. A large number of data suddenly appear in a very short time. It is very difficult to find attacks in the network according to the huge number and variety of network packet data. In addition, in order to fully explain whether the network is currently under attack, it is inaccurate to judge only based on the past and current network status, because the packet flow in the network is uncertain. To improve the detection rate, it is also necessary to rely on the packet changes in the network in the future.

According to the characteristics of packet data in the above network, in order to better detect ARP attacks, this paper needs to solve the following three problems:

(a) The amount of message data in the network is huge and various. How to filter out a large number of packets that are unrelated to attacks and have no strong correlation between data without destroying the characteristics of packet sequence?
(b) How to quickly and accurately detect ARP attacks in the network by taking advantage of the periodic change of forged ARP messages sent by attackers?
(c) How to accurately determine whether ARP attacks have occurred in the system according to the current, past and future network status?

To solve the above problems, this paper proposes an improved CNN-BILSTM hybrid model. The model combines the advantages of CNN in extracting the local characteristics of the message and BiLSTM in processing timing data, capture the ability of the periodic change regularity, can not destroy the premise of the message sequence characteristics, effectively filtering irrelevant message, and using the fake ARP message periodic changes, combined with the current, past and future network state information, comprehensively and accurately determine whether the ARP attack occurred in the system. The model structure mainly consists of four parts, which is shown in Figure 3.

Aiming at the three problems, this paper proposes an improved cnn-bilstm hybrid model to detect ARP attacks in industrial control systems. The structure of the model is mainly composed of four parts, and its structure is shown in Figure 3.

From Figure 3, it can be seen that the relevant data is inputted and preprocessed through the Convolutional Neural Network using AvergePooling2D. After implementing Time Distribute (plate), two parallel LSTM layers are introduced through BI LSTM, one from front to back (forward) and one from back to front (backward). The forward and backward contextual information

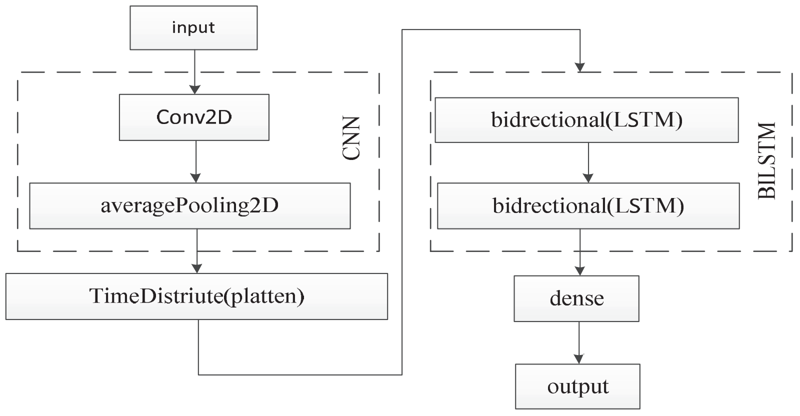


Figure 3 CNN-BILSTM hybrid model.

of the sequence is captured simultaneously to achieve dense processing of ARP virus attack data and output detection results.

(a) Convolutional neural network (CNN)
CNN is a feedforward neural network, which is mainly composed of convolution layer and pooling layer. The convolution layer mostly uses multiple convolution cores for convolution operation at the same time, and generates multiple feature maps to extract different features of the data. Next, the pooling layer will further extract the features of the feature map generated by the convolution layer and reduce the data dimension. Therefore, the model first uses CNN to filter out a large number of packets in the data that are unrelated to the attack and have weak correlation, and extract the significant features in the data. In order to keep the time-series characteristics of the data from being destroyed, the pooling method used in this paper is average pooling, and the convolutional kernel and pooling kernel are both 2×2 convolutional neural networks.

(b) Connection layer
The feature data extracted by CNN is time tagged through timedistribution, and then flattened into one-dimensional data through flatten layer, which is used as the input of bilstm layer.

(c) The core part of the model uses the bidirectional long-term and short-term memory network (bilstm).
LSTM neural network is a kind of cyclic neural network, which is mainly composed of input gate, output gate and forgetting gate. It can process data

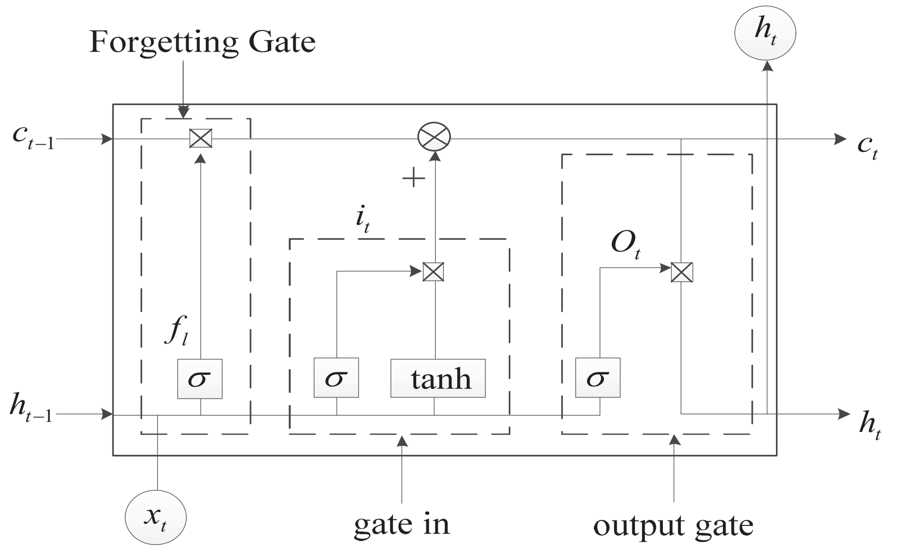


Figure 4 Basic units of LSTM neural network.

with temporal characteristics, such as language and character recognition. Compared with the traditional recurrent neural network, LSTM solves the problems of gradient disappearance and gradient explosion. The basic units of LSTM neural network are shown in Figure 4.

In the context of deep learning and recurrent neural network (RNN), especially in the long-term and short-term memory (LSTM) network, input gate, output gate and forgetting gate play an important role. The input gate, as the controller for information flow, determines which input signals should be allowed to be updated to the storage unit of the LSTM unit. It makes decisions based on the current input and the hidden state of the previous time step, ensuring that only important information can affect the state of the storage unit. The output gate controls the effect of memory cell state on other neurons. It also determines when and how to transfer the information in the storage unit to the subsequent parts of the network based on the current input and the previous hidden state. Forgetting gate is a key part of LSTM, which determines how much previous information the storage unit should retain or forget. By adjusting the self cyclic connection weight of the storage unit, the forgetting gate allows the LSTM unit to remember or forget the historical information as needed, so as to effectively deal with the problem of long-term dependence.

The input gate, output gate and forgetting gate formulas of the basic unit of LSTM neural network are as follows:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{14}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{15}$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{16}$$

Where: i_t is the information forgetting value, o_t is the storage capacity of unit status, f_t is the received value in hidden state, Their inputs are the input x_t of the t time storage unit and the output h_{t-1} of the previous time, but their respective weights WU and offset values b are different. Where σ represents sigmod function.

Storage unit status at the previous time:

$$\widetilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{17}$$

When the input gate activation i_t, the forgetting gate activation f_t and the previous unit status value $\widetilde{c}_t$ are given, the new status of the storage unit at time t can be calculated:

$$c_t = i_t \odot \widetilde{c}_t + f_t \odot c_{t-1} \tag{18}$$

Current output unit:

$$h_t = o_t \odot \tanh(c_t) \tag{19}$$

Where: $\odot$ represents the bitwise multiplication of elements in the vector. From the formula, it can be seen that the storage unit c_t is a memory, which can obtain the information in the last group of time data and act on itself, that is, the output information at t time in the neural network contains not only the input information at the current time, but also the output information at $t - 1$ time.

BILSTM neural network is composed of two LSTM neural networks, one of which is trained in the forward direction and the other is trained in the reverse direction. The output result is determined by the LSTM neural networks in the forward and reverse directions. Compared with the standard LSTM neural network, the weight of bilstm neural network is determined not only by the input data, but also by the data to be input in the future.

Therefore, the model uses bilstm neural network to make full use of current, past and future network state information by training network weights. This feature enables the model to fully consider the timing characteristics of attack messages when detecting ARP attacks in industrial control systems. Different from traditional detection methods, Bilstm not only pays attention to the historical data, but also uses the future state as the prediction basis, which significantly improves the prediction accuracy. This comprehensive way of information utilization makes the model more advantageous in dealing with complex and changeable network attacks.

(d) Output layer. The full connected network and softmax function are used to display the output of data prediction results.

4 Experiment

4.1 Experimental Design

(1) Experimental data

In order to evaluate and verify the processing effect of this chapter's Feature Engineering on the original data and the actual performance of this chapter's intrusion detection methods, the etcn intrusion data set constructed in this paper, that is, the ETI data set, is used as the main verification data of this chapter. See Section 1 for information about this dataset. In addition, the normal network data collected on a certain type of actual train is introduced into the real vehicle normal data experiment, and the corresponding auxiliary data set is constructed to further verify the ability of intrusion detection methods in this chapter to establish the behavior boundary between normal and attack.

(2) Hardware and software environment

The software and hardware environment of the experiment in this chapter is: Windows 1064 bit operating system, python3.8 development language, AMD ryzen 51600x CPU@3.60GHz, NVIDIA geforce GTX 1060 GPU, 32GB ram, etc. All these experiments will be carried out in this environment and will not be repeated then.

(3) Superparameter setting

The superparameter refers to the model parameters that need to be set manually. The superparameter settings of the experiments in this chapter are shown in Table 3, and the corresponding setting basis is as follows

For the feature selection method, the redundancy degree weight 22 is used to adjust the punishment intensity of redundancy between features. The

Table 3 Super parameter setting

Method	Parameter	Numerical Value
Feature selection	Redundancy weight ω	0.4
method	Nonlinear weight h	0.7
Intrusion detection	Maximum Number Of Iterations M	100
methods	Maximum depth D	6
	Sampling proportion of large gradient samples a	0.5
	Sampling proportion of small gradient samples b	0.1
	Prior weight coefficient ω	0.1

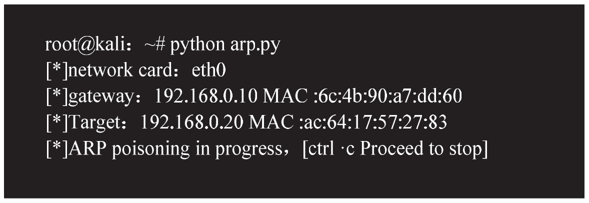
value range is (0,1]. The greater the value, the greater the punishment to the redundancy degree. Since the redundancy weight is not without contribution, it is set ω 0.4. The nonlinear weight h is used to adjust the proportion of the nonlinear part in the comprehensive measurement coefficient. The value range is (0,1]. The larger the value, the higher the proportion of the nonlinear part. Because the network data has the characteristics of high nonlinearity, h is set to 0.7.

For intrusion detection methods, the maximum number of iterations m refers to the total number of iterations of the detection model, which is equivalent to the total number of basic decision trees. The maximum depth D refers to the depth limit of each decision tree. Because the greater the number of iterations and the maximum depth, the stronger the fitting ability of the model, but it is also easier to trigger over fitting, so set these two parameters to 100 and 6. The sampling ratios a and b of large and small gradient samples in the unilateral gradient sampling strategy are set to empirical values, i.e. 0.5 and 0.1. The prior weight coefficient ω in the designed category feature processing method should be set to a smaller normal number, so it should be set to 0.1.

(4) Experimental attack process

The same as the experimental background of replay attack above, ARP spoofing attack is carried out when the system is close to stability. The attack background is the ARP attack script under Kali Linux. ARP poisoning is performed on the water level control system under normal operation, as shown in Figure 5.

Figure 5 shows the real-time liquid level diagram under ARP spoofing attack. Under the ARP spoofing attack, the third-party attacker cheated the PLC controller and the PC terminal host computer at the same time, so that the communication of the control system was interrupted.



```
root@kali：~# python arp.py
[*]network card：eth0
[*]gateway：192.168.0.10 MAC :6c:4b:90:a7:dd:60
[*]Target：192.168.0.20 MAC :ac:64:17:57:27:83
[*]ARP poisoning in progress，[ctrl ·c Proceed to stop]
```

Figure 5 ARP poisoning process.

Table 4 Confusion matrix

	Test category	
Real Category	Attack	Normal
Attack	TP	FN
Normal	FP	TN

4.2 Experimental Index

In the two category network intrusion detection task, all possible detection results of attack samples (positive cases) and normal samples (negative cases) can be divided into the following four situations:

True positive (TP): correctly detect attack samples as attack samples;
False positive (FP): false detection of normal samples as attack samples;
False negative (FN): the attack sample is incorrectly detected as a normal sample;
True negative (TN): correctly detect normal samples as normal samples.

The relationship between the above four cases can be represented by the confusion matrix, as shown in Table 4.

Confusion matrix is the original detection result of intrusion detection. Based on it, there are many performance metrics derived from it. According to statistics, the four most commonly used indicators in machine learning intrusion detection tasks are accuracy, recall, score and accuracy. First of all, the evaluation indicators in this chapter continue to use the above indicators. Because F1 score is a comprehensive measure of the accuracy rate and recall rate, it contains the information that these two indicators can reflect. Therefore, in order to avoid redundancy, F1 score is used to replace the accuracy rate and recall rate in the specific evaluation and analysis; Secondly, considering the control network characteristics of etcn, we should pay attention to the misjudgment of its normal data, so the false alarm rate index is added to the evaluation index; Finally, since this chapter mainly pursues the detection efficiency of etcn intrusion detection, the training time and test time indicators reflecting the model speed are added to the evaluation indicators.

The above indicators are divided into indicators reflecting detection accuracy and model speed. The practical significance, calculation method and specific selection reasons of their expression are as follows:

(1) Index reflecting detection accuracy

(a) F1 points

Before introducing the F1 score, first give the calculation method of accuracy rate P and recall rate R:

$$\begin{cases} \mathrm{P} = \dfrac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}} \\ \mathrm{R} = \dfrac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} \end{cases} \tag{20}$$

Among them, the accuracy rate P represents the proportion of correct detection in all samples detected as attacks; The recall rate of R indicates the proportion of all samples that are really attacks that are correctly detected as attacks.

The F1 score is the harmonic average of P and R, which can comprehensively reflect its comprehensive performance. Its calculation method is:

$$F_1 = \frac{2 \times \mathrm{P} \times R}{\mathrm{P} + R} \tag{21}$$

In the intrusion detection task, P reflects the detection accuracy of attack samples, R reflects the detection integrity of attack samples, and F_1 scores cover the information of P and R, which can comprehensively reflect the detection ability of the model to attack samples.

(b) Acc(Accuracy)

Accuracy refers to the proportion of correctly detected samples in all samples (including detecting normal samples as normal and detecting attack samples as attack). It can reflect the overall detection accuracy of the model for attack samples and normal samples. Its calculation method is:

$$Acc = \frac{TP + TN}{TP + FP + TN + FN} \tag{22}$$

(c) FPR(False Positive Rate)

The false alarm rate is also known as the false alarm rate. Intuitively, false alarm refers to the generation of “false kill”, that is, the false detection of normal samples as attack samples. The false alarm rate fpr refers to the proportion of normal samples that are incorrectly detected as attack in all normal samples, reflecting the "false kill" degree of the model. The lower the better. In etcn, normal samples refer to legal network communication data. Since these data may contain key control instructions, false positives of normal samples as attacks may lead to subsequent defense systems intercepting normal communication data, thus affecting the normal operation of trains.

Therefore, extra attention should be paid to the false alarm rate when evaluating the performance of etcn intrusion detection. Its calculation method is:

$$FPR = \frac{FR}{\mathrm{FP} + \mathrm{TN}} \tag{23}$$

It should be noted that the traditional intrusion detection evaluation index also includes the missing alarm rate index, also known as the missing alarm rate. When evaluating intrusion detection based on machine learning, "missing alarm rate = 1-recall rate", that is, the information contained in this index can be completely reflected by the recall rate and F1 score. Therefore, in order to avoid redundancy, it is not necessary to include this index in the evaluation.

(2) Indicators reflecting model speed: training time and testing time
Training time refers to the time consumed by completing the training of the intrusion detection model with the training data, and the test time refers to the time actually consumed when applying the trained model to the test data. After completing the model configuration, turn on the timer before the training or test, and turn off the timer after the training or test, so as to complete the index calculation. With the same detection accuracy reached, the shorter the training time and test time, the higher the detection efficiency of the model.

4.3 Feature Selection Experiment

This section carries out feature selection experiments on ETI data sets to select effective etcn sample features. In the two-stage feature selection algorithm designed in this chapter, 40 dimensional features as shown in Table 5 have been obtained through manual selection in the first stage; The

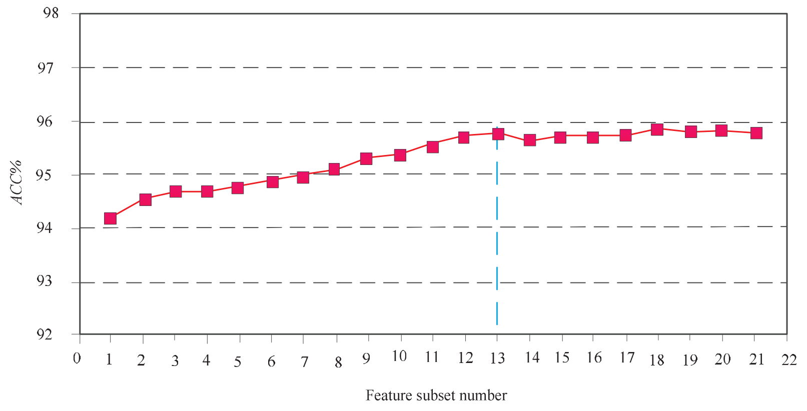


Figure 6 Detection accuracy of intrusion detection models trained on different feature subsets.

algorithm in the second stage is used to further screen these 40 dimensional features, and the final comprehensive measurement coefficient is gradually calculated according to the derived calculation formula v (k), according to v (k) According to the selection strategy of the designed feature selection algorithm, starting from the top 20 dimensional features, the subsequent features are added one by one in order to construct different feature subsets (that is, the nth subset contains the top 20 + (n − 1) dimensional features in the order). For 40 dimensional features, a total of 21 feature subsets can be constructed on the basis of ETI data set. These feature subsets are divided into training set and verification set according to the ratio of 8:2. Different improved CNN bilstm models are trained and the accuracy of the model on the verification set is recorded. The results are shown in Figure 6.

From Figure 6, it can be seen that as features are continuously added in order, the detection accuracy of the model gradually improves, and the comprehensive measurement coefficient v (k) The front features have a more significant promoting effect on accuracy, while the later features have the opposite effect. Starting from the 13th feature subset, adding new features can no longer significantly improve accuracy, and may even lead to a decrease. This may be due to the introduction of noise by the newly added features. Due to the fact that the fewer features there are, the faster the model speed is. Therefore, it should be preserved as few features as possible while ensuring accuracy. Therefore, it can be considered that the 13th feature subset can achieve the best detection efficiency of the model. This feature subset includes v (k) The first 32 dimensional features after sorting, therefore, the

Table 5 Selected effective features

Serial Number	Feature Name	Serial Number	Feature Name
1	Protocol	17	UDP source port number
2	Total packet length	18	UDP destination port number
3	Destination MAC	19	Udp Length
4	Source Mac Address	20	TCP source port number
5	Network layer protocol type	21	TCP destination port number
6	Distinguish service domains	22	TCP confirmation number
7	IP length	23	TCP flag field
8	life	24	TCP receive window
9	Source IP Address	25	Trdp serial number
10	Destination IP Address	26	Trdp message type
11	Sender MAC address	27	Trdp communication identifier
12	Sender IP address	28	ETB topology counter
13	destination mac address	29	Train operation topology counter
14	Destination IP address	30	Trdp length
15	ICMP type	31	Requested communication identifier
16	ICMP code	32	Reply IP address

effective features obtained after the second stage of feature selection are shown in Table 5.

It should be noted that the selected features are based on feedback from experimental results. Therefore, the contribution of corresponding features in intrusion detection experiments varies depending on the source of data acquisition and the type of attack. For example, in the training data used in this article, there is little difference between attack data and normal data in IP flags (related to IP sharding). However, in other possible ETCN scenarios, this feature may also undergo significant changes when facing other types of attacks that may exist. Under these limitations, the features selected in this chapter are based on the proposed feature selection method and obtained from ETI data, mainly reflecting the attack simulation experiments conducted on the experimental platform in this paper; In practical scenarios, it is necessary to further select corresponding features based on the relevant intrusion data in the scene according to the methods in this chapter.

4.4 Intrusion Detection Experiment

This section conducts intrusion detection experiments on the ETI dataset to evaluate the actual performance of the proposed improved CNN BiLSTM.

Table 6 Sample distribution after division

Subset Name	Number of Samples	Proportion to Total Sample Size/%
Training set	857918	80
Validation set	107239	10
Test set	107239	10

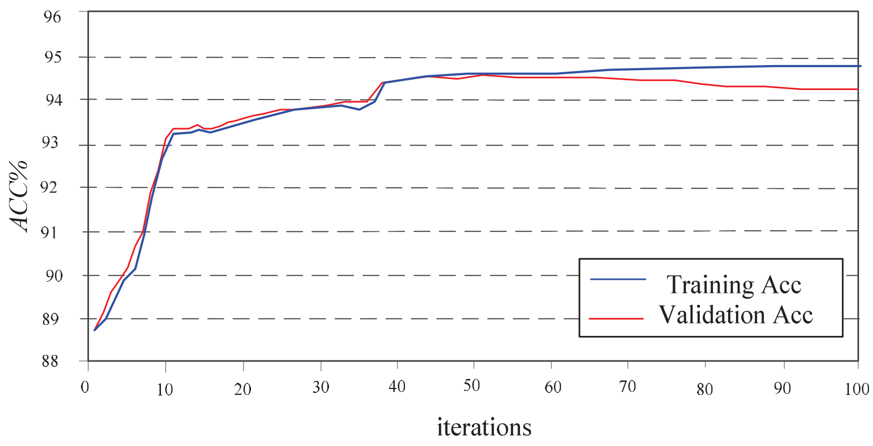


Figure 7 Changes in training accuracy and validation accuracy during training.

Divide the ETI dataset processed by feature engineering into training, validation, and testing sets in an 8:1:1 ratio. The sample distribution after partitioning is shown in Table 6.

Use the training set to train and improve the CNN BiLSTM model, and use the validation set to observe the performance of the model during the training process. The training accuracy and validation accuracy during this process are shown in Figure 7.

As shown in Figure 7, in the first 10 rounds of training, the training accuracy and validation accuracy of the model rapidly increase. This indicates that the model learns a large amount of information in this stage of iteration, which effectively determines the splitting paths of the first few basic decision trees, thereby rapidly improving the overall testing ability of the model. After this stage of training, the difficulty of learning the effective information contained in the data gradually increases, slowing down the learning speed of the model, and the accuracy changes show an overall upward trend, accompanied by small local fluctuations, which may be caused by learning too much data noise. In the 38th round of training, the accuracy of both training and validation showed a sudden increase. From a theoretical perspective, this

Table 7 Overall test results of improved CNN BiLSTM

P /%	R/%	F1/%	Acc/%	FPR/%	Training Time/s	Test Time/s
94.36	94.49	94.42	94.58	5.33	8.0746	0.1664

may be due to the model jumping out of the local space during training and finding a critical splitting path, resulting in a new significant improvement. Starting from the 50th round of training, although the training accuracy of the model has slightly increased, its numerical value is significantly higher than the validation accuracy. This indicates that the model has overfitting, and new iterations are difficult to obtain more effective information. At the same time, more data noise is gradually superimposed and learned. Therefore, the optimal model is selected based on the results of the 50th round of iteration, and the model parameters and structure are locked in. After locking the optimal model, the test set was used for testing, and the results of various evaluation indicators are shown in Table 7.

As shown in Table 7, the improved CNN BiLSTM achieved high F1 score (94.42%), high accuracy (94.58%), and low false alarm rate (5.33%) on the ETI dataset, indicating that the model has good detection accuracy; On this basis, the model consumed very little training time (8.0746s) and testing time (0.1664s). Overall, the model can achieve good detection accuracy while quickly completing training and testing, indicating that the intrusion detection model established based on the improved CNN BiLSTM designed in this chapter has excellent detection efficiency performance.

5 Conclusion

The paper proposes an improved CNN BiLSTM based industrial Internet of Things ARP virus attack detection method. Construct an industrial Internet of Things ARP virus intrusion dataset, achieve industrial Internet of Things ARP feature extraction through feature extraction, data cleaning, feature transformation, and feature selection, construct an improved CNN-BiLSTM structure, use CNN to filter out packets unrelated to attacks in the data, extract prominent features in the data, label the feature data extracted by CNN with time labels through timeDistribution, flatten it into one-dimensional data through the flat layer, and use it as input to the BILSTM layer. We used a bidirectional long short-term memory network (BILSTM) to train industrial IoT ARP virus attacks and output the final ARP virus attack detection results. The experimental results show that in the first 10 rounds of training, the training accuracy and validation accuracy of the model increase rapidly. This indicates that the model learns a large amount of information in the iteration of this stage. After this stage of training, the learning difficulty of the effective information contained in the data gradually increases, causing the learning speed of the model to slow down, and the change in accuracy shows an overall upward trend. In the 38th round of training, the accuracy of training and validation increases sharply at the same time. We achieved high F1 score (94.42%), high accuracy (94.58%), and low false alarm rate (5.33%) on the ETI dataset. The model consumed very little training time (8.0746s) and testing time (0.1664s). Verified the effectiveness of the design model.

References

[1] Qian Hongbing, Li Yanli. Design and implementation of a university network attack detection platform based on the Spark framework [J]. Digital Technology and Applications, 2023, 41 (5): 214–217.

[2] Shen Wuqiang, Cui Lei, Xu Mingjie, et al. Research on SQL injection attack detection based on ABLSTM [J]. Micro Computer Application, 2023, 39 (3): 43–46.

[3] Sheng Quanwei. An Intelligent Detection Method for Network Vulnerability Attacks Based on Correlation Analysis [J]. Information and Computer (Theoretical Edition), 2022, 34 (13): 238–240.

[4] Niu Xiaojun. Autonomous detection method for DDoS attacks based on BP neural network [J]. Communication Power Technology, 2023, 40 (3): 153–155.

[5] Li Tongxin, Wang Yong, Zou Chunming, et al. An IFAR attack detection algorithm based on DNP3 protocol [J]. Microcomputer Applications, 2022, 38 (11): 1–5.

[6] Wu Lian, Zhao Chenjie, Wei Pingping, et al. A Computer Virus Detection Method Based on Lightweight Deep Networks [J]. Computer Engineering and Design, 2022, 43 (3): 632–638.

[7] Yang Shixin, Fan Jiulun, Huang Wenhua, et al. Ransom virus detection system based on Android dynamic static fusion industrial equipment [J]. Journal of Xi'an University of Posts and Telecommunications, 2022, 27 (02): 95–101.

[8] Ying Xianer, Chen Xiner, Sun Leyao, et al. Research on Virus Transmission Network Intrusion Detection Based on Graph Neural Networks [J]. Industrial Control Computer, 2023, 36 (5): 104–105.

[9] Yuan Huihua. Research on Anomaly Threat Detection Based on Network Full Traffic Analysis Technology [J]. Changjiang Information and Communication, 2022, 35 (11): 137–139.

[10] Hao Wentao. Research on Traffic Based Intrusion Detection Technology in Industrial Control Networks [J]. Computer Application Abstracts, 2023, 39 (16): 97–100.

[11] Jiang Xiaojing, Wei Yifei. Research and exploration on the detection and prevention of unknown Trojan viruses [J]. China Financial Computer, 2023 (8): 88–90.

[12] Jia Junjie, Duan Chaoqiang. Torch attack detection algorithm based on score dispersion [J]. Computer Engineering and Science, 2022, 44 (03): 554–562.

[13] Fan Yuchen, Liu Xiangkun, Zhu Jiansheng, et al. Research on Web Attack Detection of Service Websites Based on BERT [J]. Computer Technology and Development, 2022, 32 (08): 168–173.

[14] Li Peng, Wang Fangyuan. Design of Network Illegal Intrusion Detection System Based on Big Data Environment [J]. Information Recording Materials, 2022, 23 (11): 223–225.

[15] Feng Guocong, Fan Kai, Ye Wanqi. Design of Network Intrusion Detection System Based on Convolutional Neural Network [J]. Microcomputer Applications, 2023,39 (05): 141–143+154.

[16] Yu Ning, Gu Liang, Di Ting. A Network Attack Detection Model Based on Deep Learning [J]. Fire and Command Control, 2023, 48 (5): 66–74.

[17] Hu Yidan. Cross site Script Attack Detection Model Based on Convolutional Neural Networks [J]. Ship Electronics Engineering, 2023, 43 (6): 110–115.

[18] Sun Qian, Wu Ming. Simulation of Network Client Virus Defense Method under N-Gram Model [J]. Computer Simulation, 2022,39 (10): 400–404.

Biography

Jianhua Wang, date of birth: July 12, 1977, male, Han nationality. Native place: Langfang City, Hebei Province, Master's degree, lecturer, Research interests: Information and signal processing.