IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 25 January 2023, accepted 5 March 2023, date of publication 9 March 2023, date of current version 15 March 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3254915

RESEARCH ARTICLE

NIDS-CNNLSTM: Network Intrusion Detection Classification Model Based on Deep Learning

JIAWEI DU[1], KAI YANG[1], YANJING HU[2], AND LINGJIE JIANG[3]
[1]School of Computer Science, Xijing University, Xi'an, Shaanxi 710123, China
[2]School of Cryptographic Engineering, Engineering University of PAP, Xi'an, Shaanxi 710086, China
[3]School of Electronic Information, Xijing University, Xi'an, Shaanxi 710123, China

Corresponding author: Kai Yang (sydeny-001@163.com)

This work was supported by the High-level Talents Special Fund Project of Xijing University in 2022, "Research on Industrial Internet of Things Intrusion Detection Technology Based on Deep Learning" (XJ22B04).

ABSTRACT Intrusion detection is the core topic of network security, and the intrusion detection algorithm based on deep learning has become a research hotspot in network security. In this paper, a network intrusion detection classification model (NIDS-CNNLSTM) based on deep learning is constructed for the wireless sensing scenario of the Industrial Internet of Things (IIoT) to effectively distinguish and identify network traffic data and ensure the security of the equipment and operation of the IIoT. NIDS-CNNLSTM combines the powerful learning ability of long short-term memory neural networks in time series data, learns and classifies the features selected by the convolutional neural network, and verifies the applicability based on binary classification and multi-classification scenarios. The model is trained using KDD CUP99, NSL_KDD, and UNSW_NB15 classic datasets. The verification accuracy and training loss on the three datasets all show good convergence and level, and the accuracy rate is high when classifying various types of traffic. The overall performance of NIDS-CNNLSTM has been significantly improved compared with the models proposed in previous studies. The effectiveness shows a high detection rate and classification accuracy and a low false alarm rate through experimental results. It is more suitable for large-scale and multi-scenario network data in the IIoT.

INDEX TERMS Network intrusion detection, deep learning, convolutional neural network, long short-term memory neural network.

I. INTRODUCTION

The Industrial Internet of Things (IIoT) has brought new opportunities for global development. However, it will also bring security risks, such as industrial core data leakage and illegally manipulating interconnected terminals. Therefore, the security of the IIoT is facing significant challenges. IIoT security protection mainly adopts active-passive defense modes, such as industrial firewall technology and intrusion detection technology. Firewall technology is a passive defense technology that cannot prohibit the transmission of files and programs with threatening codes, such as viruses or worms. Therefore, intrusion detection technology is introduced to make up for the deficiency of firewall technology. Intrusion detection technology [1], [2] is an active network security protection measure based on traditional static protection. It realizes real-time protection against internal and external intrusions through real-time network monitoring. It is proactive, real-time, dynamic characteristics. Therefore, as a network security protection measure, intrusion detection technology has become a research hotspot in IIoT security. Essentially, intrusion detection uses a classifier to distinguish between normal and abnormal data in the data stream to realize the alarm of attack behavior. This classifier [3] can be based on Bayesian [4], decision tree [5], neural network [6], and support vector machine [7]. In recent years, the research on intrusion detection and classification algorithms [8] has been mainly divided into two categories: based on traditional machine learning [9], [10], [11] and based on deep learning [12], [13], [14]. Facing the increasingly complex IIoT environment, researchers have proposed various intrusion detection models for different network attacks and

The associate editor coordinating the review of this manuscript and approving it for publication was Daniel Augusto Ribeiro Chaves.

applied machine learning algorithms to intrusion detection models. These models have achieved specific results but could still be improved. Deep learning has intense feature extraction and learning classification capabilities. It is a new field of machine learning in recent years and has attracted more scholars' attention at home and abroad. Intrusion detection algorithms based on deep learning have solved many challenging problems.

Xiao et al. [15] adopted an auto-encoder (AE) to reduce the dimension of the data to decrease the interference of redundant features, and a convolutional neural network (CNN) was adopted to identify the intrusion information. Staudemeyer [16] introduced long short-term memory (LSTM) into intrusion detection, explored the correlation of the temporal domain of intrusion information, and effectively reduced the rate of false positives. Zhang et al. [17] proposed an intelligent grid intrusion detection model that combines the genetic algorithm (GA) and extreme learning machine (ELM). The model retains the advantages of the ELM, and the GA is introduced to ensure the optimal parameters of the model. Vinayakumar et al. [18] connected CNN and LSTM. They showed a serial CNN-LSTM intrusion detection system model to extract high-level feature representations representing the abstract form of low-level feature sets of network traffic connections. Yao et al. [19] proposed an AMI intrusion detection model based on the cross-layer feature fusion of CNN and LSTM to obtain comprehensive features with multi-domain characteristics based on the KDD Cup 99 and NSL_KDD datasets. Yang and Wang [20] adopted an improved CNN to identify intrusion information. The CNN is improved to extract features across layers, and feature fusion is used to obtain comprehensive features. Liu and Zhang [21] used CNN to identify intrusions and improved the model's accuracy through data augmentation techniques. Shen et al. [22] applied an ELM to intrusion detection, which improved the model's detection speed and generalization ability. Halbouni et al. [23] established a hybrid intrusion detection system model by using the ability of a CNN to extract spatial features and an LSTM network to extract temporal features. Batch normalization and dropout layers were added to the model to improve its performance. Thaseen and Kumar [24] proposed an intrusion detection model using chi-square feature selection and multi-class support vector machines (SVM). The parameter adjustment technology is used to optimize the radial basis function kernel parameters. A multi-class support vector machine is constructed to reduce the training and testing time and improve the individual classification accuracy of network attacks. Sahu et al. [25] proposed a deep-learning model to solve the intrusion classification problem effectively. Classify benign and malicious traffic on intrusion datasets using LSTM and Fully Connected Network (FCN) deep learning methods to classify multi-class attack patterns more accurately. Ikram et al. [26] utilized an ensemble of different deep neural network (DNN) models, such as multilayer perceptron (MLP), backpropagation network (BPN), and LSTM, to be stacked to build a robust anomaly detection model. XGBoost combines the results of each deep learning model to achieve higher accuracy, utilizing deep learning techniques to identify intrusions with maximum accuracy and reduce false favorable rates.

Based on existing research, this paper constructs a network intrusion detection classification model based on deep learning (NIDS-CNNLSTM) to further improve the detection model's detection rate and classification accuracy. The model employs two deep learning algorithms: CNN [27], [28] and LSTM [29], [30], [31]. CNN and LSTM extract the Spatiotemporal features of network traffic data, which can more effectively identify intrusion information in the IIoT. NIDS-CNNLSTM is evaluated on KDD CUP99, NSL_KDD, and UNSW_NB15 datasets; all three datasets contain rich samples and cover all possible types of attacks in Industrial IoT. Selecting these three classic network intrusion detection public datasets as benchmark datasets is the basis for fair comparison and verification of calculation methods. Training multiple models in comparative experiments can better evaluate the overall performance of intrusion detection models [32], [33]. Based on the applicability of the verification model in binary and multi-classification scenarios, NIDS-CNNLSTM showed better verification accuracy and training loss in both binary and multi-classification scenarios, improving the intrusion detection rate and classification accuracy. Compared with models proposed in previous studies, the model's overall performance has been significantly improved, further demonstrating the model's effectiveness.

The organization of the article is as follows: the first part is the introduction, which introduces the background and significance of the research in this field and the related work at home and abroad; the second part is the preliminary knowledge, which outlines the related theory of intrusion detection model and neural network; the third part is the algorithm model architecture, which introduces the construction process of the NIDS-CNNLSTM model in detail, mainly including data collection, data processing, CNN-LSTM model, decision-making judgment; the fourth part is the simulation experiment, including the experimental environment, evaluation indicators, model performance, and comparative experiments; the fifth part is the summary and the future work.

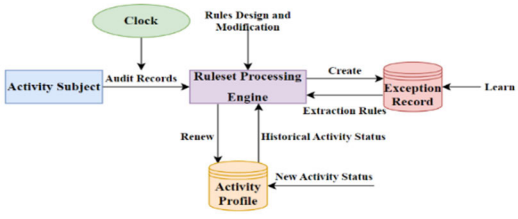

FIGURE 1. General intrusion detection model.

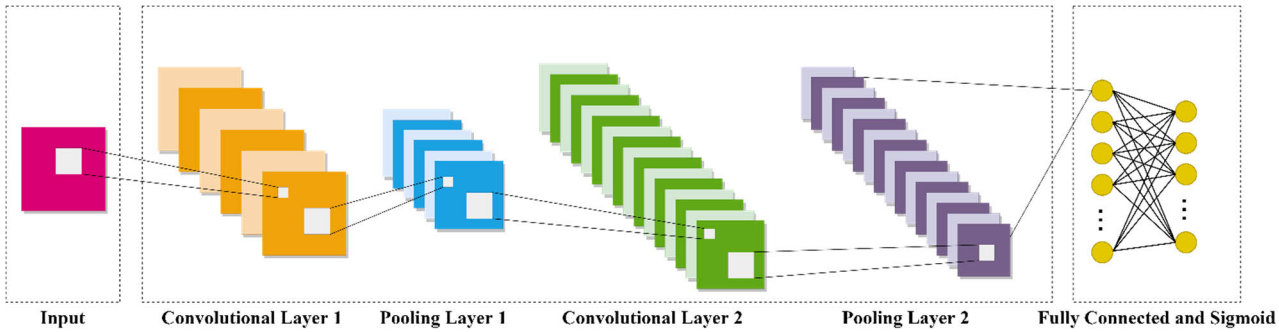

FIGURE 2. Model structure diagram.

II. PRELIMINARY KNOWLEDGE

A. INTRUSION DETECTION MODEL

Intrusion detection is the detection of intrusion behaviors. It collects and analyzes network behaviors, security logs, audit data, and other information available in the network and hints at several critical points in the computer system. It checks whether behaviors violate security policies and signs of being attacked in the network or system. The flowchart of the general intrusion detection model is shown in Figure 1. The schematic diagram of the general intrusion detection model is shown in Figure 1. The model proposed in this paper is based on the general intrusion detection model.

B. CONVOLUTIONAL NEURAL NETWORKS

CNN is a practical algorithm for deep learning. It is a feedforward neural network with convolution calculation and deep structure. It is often designed to process multi-dimensional array data. It can accurately extract the local correlation of features and improve the accuracy of feature extraction. A CNN consists of an input layer, a hidden layer, and an output layer. The input layer is used to receive normalized array data. The hidden layer includes a convolutional layer, a pooling layer, and a fully connected layer. The convolutional layer uses a convolution kernel for feature extraction and features mapping excitation layer. After the convolutional layer performs feature extraction, the output feature map will be passed to the pool. The pooling layer performs feature selection and information filtering; the pooling layer performs down sampling, sparsely processes the feature map, reduces the amount of data computation, dramatically reduces the parameter magnitude, and effectively avoids overfitting; the fully connected layer is usually refitted at the tail of the CNN to minimize the loss of feature information. The output layer directly outputs the classification result of each feature. The CNN structure is shown in Figure 2.

CNN is divided into one-dimensional, two-dimensional, and three-dimensional convolution. The one-dimensional CNN selected in this paper is mainly applied to time series data. Assuming that the lth layer is a convolutional layer, the calculation formula of one-dimensional convolution is shown in (1).

$$x_k^l = f\left(\sum_{i=1}^{N} x_i^{l-1} \times w_{ik}^l + b_k^l\right) \tag{1}$$

where x_k^l represents the kth convolution map of the l layer, f represents the activation function, N represents the number of input convolution maps, $\times$ is the convolution operation, and w_{ik}^l is the weight of the kth convolution kernel of the l layer for the ith operation, b_k^l is the offset of the kth convolution kernel corresponding to the l layer.

This paper uses the maximum pooling method for the pooling layer; the calculation formula is shown in (2).

$$\hat{x}_k^l = max\left(x_k^l : x_{k+r-1}^l\right) \tag{2}$$

where $\hat{x}_k^l$ represents the maximum a value from vector x_k^l to x_{k+r-1}^l. For the sequence x, repeating the max-pooling operation on the continuous vector whose window is r can get the largest feature sequence.

In the last layer of the CNN, the *softmax* function is used to calculate the probability of each output class. The probability is calculated by dividing the exponent of the class of scores by the sum of the exponents of all the scores as:

$$softmax = \frac{exp(y_i)}{\sum_j^c exp(y_i)} \tag{3}$$

The loss function is:

$$H(y', y) = -\sum y' Log(softmax(y_i)) \tag{4}$$

Quantifying the degree of prediction of the calculated probability to the actual class is done by calculating the loss. In the case of the classification probability, it is done by the classification cross-entropy loss function. The predicted class (y') and the actual class (y) two vectors are used to output the total loss, and the cross-entropy loss is computed as the sum of the negative log-likelihoods of the class probabilities, expressed as a function of $H(y', y)$.

C. LONG SHORT-TERM MEMORY NEURAL NETWORKS

CNN mainly analyzes the internal features of a single data packet and lacks the extraction and analysis of the association between sequences. Therefore, the model constructed with

the LSTM network will have a better training effect on network intrusion detection. Due to its ability to maintain long-term memory, LSTM is gradually applied to various network intrusion detection models to solve the problem of gradient disappearance caused by recurrent neural networks. LSTM is a time-cyclic neural network. Based on the traditional cyclic neural network, the LSTM unit replaces the neurons in the recurrent neural network (RNN). The input gate, output gate, and forget gate are introduced in the input, output, and forgotten past information to control and allow information to pass through, maintain the memory ability of long data, and solve the existing long-term dependence problem in the RNN network.

1) FORGET GATE

The forget gate changes with the context and forgets the information that needs to be forgotten. The output of the forget gate is a sigmoid function, the value range is between 0 and 1, and it is multiplied by the cell state at the last moment, and 0 represents the information of this bit is completely forgotten, one represents that the information of this bit is completely retained. The calculation formula is shown in (5).

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \tag{5}$$

2) INPUT GATE

The input gate supplements the information the new cell state needs as much as possible. The output of the input gate is a sigmoid function with a value range of 0∼1, which is multiplied by the current cell state. The calculation formula is shown as follows (6), (7).

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \tag{6}$$

$$\tilde{C}_t = Tanh\left(W_c \cdot [h_{t-1}, x_t] + b_c\right) \tag{7}$$

Then the old and new state information can be merged to form the final new cell state. The calculation formula is shown in (8).

$$C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t \tag{8}$$

3) OUTPUT GATE

The final cell state plus the *Tanh* function is the output. The output gate's output is a sigmoid function with a value between 0 and 1. Select which information can be output. The calculation formula is shown in (9), (10).

$$O_t = \sigma\left(W_o \cdot [h_{t-1}, x_t] + b_o\right) \tag{9}$$

$$h_t = O_t \times Tanh\left(C_t\right) \tag{10}$$

The internal structure of the hidden layer of the LSTM network is shown in Figure 3.

In figure 3, f_t is the output signal of the forget gate, whose value determines the forget ratio of the memory unit c, i_t is the output signal of the output gate, and its value determines how much of the current input information is inputted into the memory unit c, and $\tilde{c}_t$ is the preparatory information to be input into the memory unit c, its value is multiplied by i_t to get the information in the memory unit c, o_t is the output signal of the output gate, and its value determines the proportion of the memory unit c output to the current state h, $\hat{c}_t$ is the preparatory information that will be output to the hidden layer state h, its value is multiplied by o_t to get the information in h, the memory unit c_t at time t has been screened by the forget gate and the input gate, and then the hidden state h_t can be obtained through the screening of the output gate.

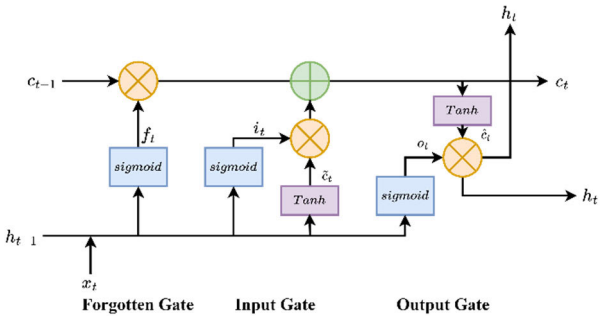

FIGURE 3. Internal structure diagram of hidden layer of LSTM network.

III. ALGORITHM MODEL ARCHITECTURE

The flowchart of NIDS-CNNLSTM proposed in this paper is shown in Figure 4. Firstly, the original network intrusion detection dataset is input into the CNN-LSTM model through data preprocessing operation. Then after the composition and evaluation of the model, the decision classification is finally carried out.

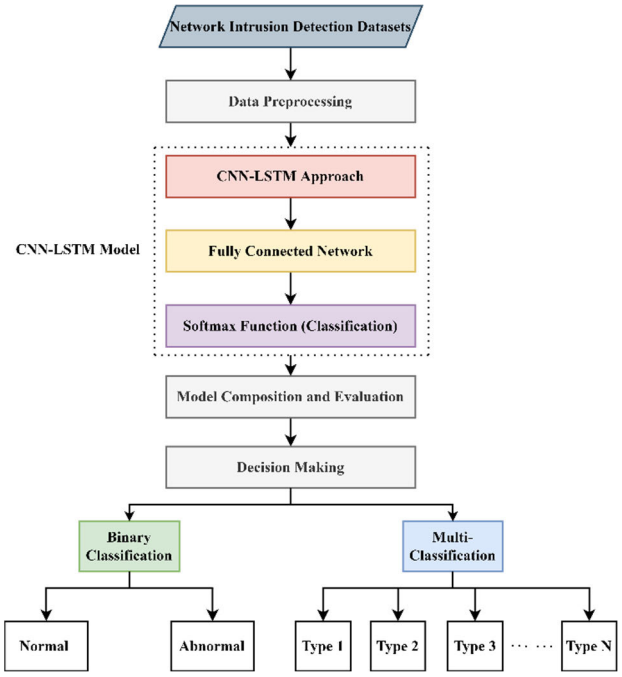

FIGURE 4. NIDS-CNNLSTM flowchart.

From the NIDS-CNNLSTM flow chart, the architecture of the model implementation mainly includes four parts:

TABLE 1. Dataset partition.

DATASET	CLASSIFICATION			
	Binary classification		Multi-classification	
	Label number	Traffic type	Label number	Traffic type
KDD CUP99	0	Normal	0	Normal
			1	DoS
			2	Probing
			3	R2L
	1	Abnormal	4	U2R
			5	Uncertain
NSL_KDD	0	Normal	0	Normal
			1	DoS
			2	Probing
	1	Abnormal	3	R2L
			4	U2R
UNSW_NB15	0	Normal	0	Normal
			1	Fuzzers
			2	Analysis
			3	Backdoors
			4	DoS
			5	Exploits
			6	Generic
	1	Abnormal	7	Reconnaissance
			8	Shellcode
			9	Worms

data collection, data processing, CNN-LSTM model, and decision-making judgment.

A. DATA COLLECTION

The input function is implemented at the bottom of the model, receiving network intrusion datasets from KDD Cup99, NSL_KDD, and UNSW_NB15. KDD CUP99 is often used as an intrusion detection system to provide a unified performance evaluation benchmark, test the quality of intrusion detection algorithms, and lay the foundation for intelligent intrusion detection systems research. In order to make deep learning algorithms better implementable on KDD Cup99, the NSL_KDD dataset was created, which will be more effective for accurately evaluating different learning techniques due to the removal of redundant data. The UNSW_NB15 dataset is a comprehensive network attack traffic dataset widely used in anomaly intrusion detection, which simulates the real attack environment as much as possible. Compared with KDD CUP99 and NSL_KDD datasets are more suitable for the research of intrusion detection systems. The UNSW-NB15 dataset is considered to be a reliable dataset for evaluating existing and novel IDS methods. The KDD Cup99, NSL_KDD, and UNSW_NB15 datasets represent the most classic and latest network attacks. The models are trained on the three datasets to evaluate the models' effectiveness better.

In the KDD CUP99 dataset, each network connection is marked as normal or abnormal, and the types of anomalies are subdivided into four categories: DoS, R2L, U2R, and Probing. The NSL_KDD dataset deletes the duplicate records in KDD Cup99, reduces the amount of data, contains the basic records and data characteristics of the KDD Cup99 dataset, and identifies the same attack categories as the KDD Cup99 dataset. The UNSW_NB15 dataset contains normal network connections and nine attack types: Fuzzer, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms.

In order to better evaluate the applicability and effectiveness of the model proposed in this paper, and to avoid chance, two classification and multi-classification experiments were set up on the three sets of datasets. The datasets were uniformly labeled, marked as normal and abnormal for binary classification experiments, and marked as multiple traffic types for multi-classification experiments. The specific divisions are shown in Table 1.

B. DATA PROCESSING

1) DATA LOADING

The loaded data is stored in a CSV file in PCAP format, and the details of each dataset are read using the Pandas package, and after reading the details of each dataset, all null and duplicate values are cleaned.

2) DATA ENCODING

Processing deep neural networks means processing values, uniformly encoding the traffic labels in the read dataset into numerical types, and using the One-Hot Encoder to encode the value of the label column.

3) DATA SCALING

The Americanized data is normalized, and the data read from the CSV file has different standard deviations and average values, and the difference will affect the learning efficiency. Scaled the input data using Standard Scalar, resulting in a mean of zero and a standard deviation of one. Library

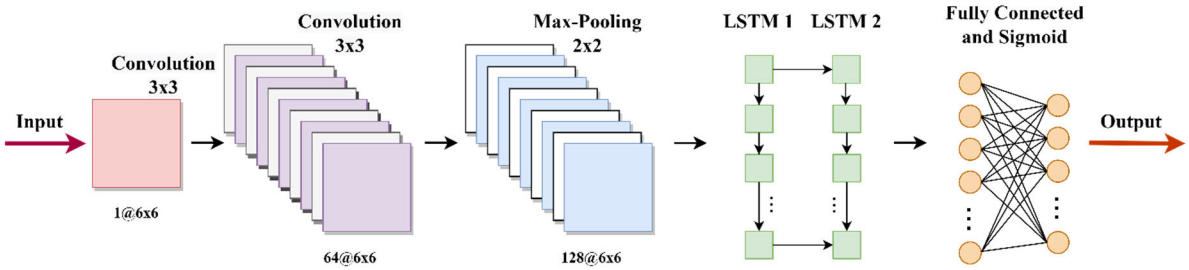

FIGURE 5. Internal structure of CNN-LSTM model.

standard scalars are used to normalize datasets according to sklearn preprocessing.

C. CNN-LSTM MODEL

1) COMPONENTS

The CNN-LSTM model mainly consists of two parts: the CNN network and the LSTM network. First, the preprocessed data is input into the two-layer CNN network, feature selection is performed on the traffic data, and the global average pooling layer is selected to replace the fully connected layer. Data feature extraction and dimensionality reduction are realized through convolution and pooling operations, and the feature matrix is the output. Then input the feature vector into the double-layer unidirectional LSTM network, and combine the powerful time series learning ability of LSTM to learn and classify the features selected by the CNN network. The forget gate, input gate, and output gate in the LSTM network adjust their parameters through continuous iterative training of a large amount of data. Then, from the data extracted by the CNN network, the time-fitting relationship between the data is learned, and the effective dynamic modeling of the input and output data of the forecast time series is carried out. Finally, the CNN-LSTM model fits the trained data and outputs the predicted value through the fully connected neural network.

2) INTERNAL STRUCTURE

For example, the CNN-LSTM model trained on the KDD CUP99 dataset in a multi-classification scenario shows the internal structure of the CNN-LSTM network in Figure 5.

In Figure 5, the preprocessed data is firstly flattened into a 1-dimensional array, and then the data is mapped from low-dimensional to high-dimensional through two layers of 3×3 convolution operations, and the height $\times$ width is 6×6 channels, feature maps of 64 and 128, respectively, in which the meaning of the data represented by different dimensions is also different; then after the maximum pooling operation, the size of the pooling kernel is 2×2, and the height and width of the feature map become 3×3, but the number of channels remains unchanged, which also shows that the maximum pooling operation only changes the size of the feature map without changing its dimension; then the critical features extracted by CNN are used as input data to the LSTM model, a two-layer one-way LSTM is selected, and the output dimension is set to 64, and then through two connected fully connected layers, the function of the first fully connected layer is to flatten the height and width, and convert the data information from the height and width dimension to the depth dimension, the number of input nodes of the fully connected layer is 256 and the number of output nodes is 64; the role of the second fully connected layer is to perform classification output, the number of input nodes in the fully connected layer is 64, and the number of output nodes is determined by the final number of classifications required. In this model, the number of classifications is 6, so the number of output nodes is 6. Finally, the output results are normalized by the *Softmax* function, the probability of the output is mapped to between 0-1, and the probability of all categories is added to 1. The hyperparameter settings of the experiment in this paper: Batch Size is 1024, Learning Rate is 0.001, Optimizer is Adam, and Epoch is 50. The CNN-LSTM model summary is shown in Table 2.

TABLE 2. Model summary.

Layer	Output
Conv1D_Input	[(None, 41, 1)]
Conv1D_1	(None, 39, 64)
Conv1D_2	(None, 37, 128)
MaxPooling1D	(None, 18, 128)
Dropout	(None, 18, 128)
LSTM _1	(None, 18, 128)
LSTM_2	(None, 256)
Dense_1	(None, 64)
Dense_2	(None, 6)

3) IMPROVEMENT PART

•Global average pooling layer instead of fully connected layer

In the CNN part, the convolutional layer before the connection layer is responsible for the feature extraction of the data. After acquiring the features, the traditional method is to connect the fully connected layer and perform activation classification, but the fully connected layer adds training and testing calculations. The amount of parameters reduces the speed, and if the parameter amount is too large, it is easy to overfit. The idea of global average pooling is to replace the fully connected layer with global average pooling, use the pooling layer to reduce dimensionality, and retain the

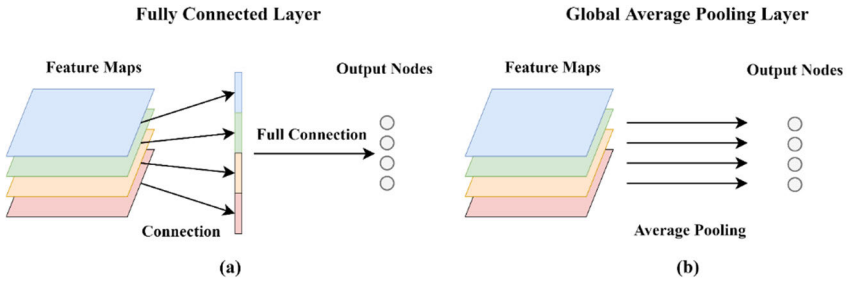

FIGURE 6. Fully connected layer and global average pooling layer.

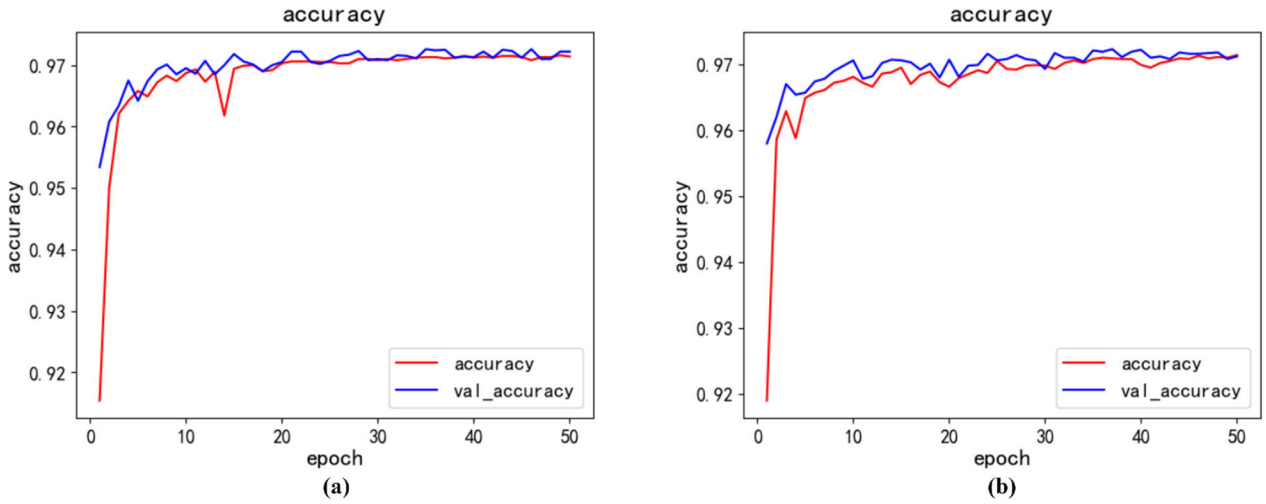

FIGURE 7. One layer of LSTM accuracy and two layers of LSTM accuracy.

spatial information or semantic information extracted by the previous convolutional layers and pooling layers. Therefore, the effect improvement is more evident in practical applications, and global average pooling has no limit to the input size. The fully connected layer expands the convolutional layer into a vector and classifies each feature map. Global average pooling is to combine the above two processes into one, as shown in Figure 6.

Figures 6 (a) and (b) show the process of the fully connected layer and the global average pooling layer, respectively. Global average pooling dramatically reduces the number of network parameters, equivalent to regularizing the entire network structure to prevent over-fitting of the model. It directly removes the features of the black box in the fully connected layer and directly gives each channel the actual category meaning. At the end of the convolution layer, as many categories are output as the map, the average value of the map is directly calculated to obtain the result. Finally, use *Softmax* function for classification.

•Selection of two layers of LSTM

When increasing the number of iterations to 50, compare the accuracy and loss function of one-layer LSTM and two-layer LSTM, as shown in Figure 7 and Figure 8.

Figures 7 (a) and (b) show the accuracy of one layer of LSTM and the accuracy of two layers of LSTM, respectively. Figures 8 (a) and (b) show the loss function of one layer of LSTM and the loss function of two layers of LSTM, respectively. The accuracy curve and loss function of a layer of LSTM have a jagged state with apparent oscillations. Although the results were not significantly affected, this casts doubt on the reliability of this unstable result. In contrast, the accuracy curve of the two-layer LSTM is smoother, and the obtained results are more stable and reliable.

The number of LSTM layers can be manageable in processing time series data. The increase in the number of layers will bring about an exponential increase in time overhead and memory overhead, and then the gradient between layers disappears. When the number of layers exceeds three, the gradient disappearance between layers will become very obvious, resulting in a slowdown of the update iteration of the LSTM layer close to the input layer and a sharp drop in the convergence effect and efficiency. In the face of a large amount of data, more is needed to amplify the neurons of one layer. Currently, the two layers of LSTM can compress the data into highly ''condensed'' data. Adding the number of deep layers can only bring about the loss of information in

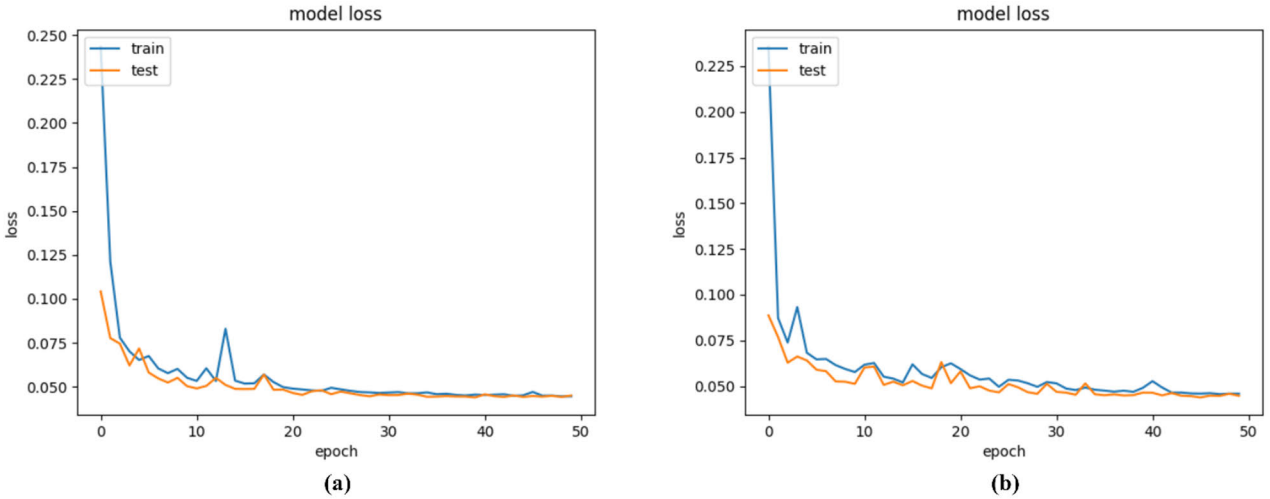

FIGURE 8. One layer of LSTM loss function and two layers of LSTM loss function.

information compression and the disappearance of gradients in the training process. LSTM solves the problem of RNN long-distance dependence and gradient disappearance mainly within each layer, not between layers, so it will be difficult to train if there are too many LSTM layers, and the feature extraction ability of each layer of LSTM is potent. Therefore, this paper chooses a two-layer unidirectional LSTM in constructing the NIDS-CNNLSTM model.

D. DECISION-MAKING JUDGMENT

After obtaining the classification results, the attack traffic can be judged. The detection results are further used to update the knowledge database to improve the system's detection capability. CNN and LSTM implementations for intrusion detection and specific architectures for classifying input data with the best performance, so networks with different structures are designed and implemented and then executed multiple times to determine the best network structure.

IV. SIMULATION EXPERIMENT

A. EXPERIMENTAL ENVIRONMENT

The experimental environment of this paper is shown in Table 3 below.

TABLE 3. Experimental environment.

Module	Parameter
processor	Intel (R) Core (TM) i7-7700HQ
main frequency	2.80GHz
RAM	8.00 GB
operating system	Windows 10
experimental tool	Python3.7 and TensorFlow2.2.0

B. EVALUATION INDICATORS

The confusion matrix is often used in intrusion detection technology as an index to evaluate classification performance. The confusion matrix is a visual display tool to evaluate the advantages and disadvantages of partition models and differences. It contains four elements: TP is the number of normal samples of normal traffic, and FN is the number of normal samples in normal traffic. The number of abnormal samples classified as different abnormal traffic; FP is the number of normal samples classified as different abnormal traffic; TN is the number of abnormal samples classified as normal traffic. The confusion matrix is shown in Table 4.

TABLE 4. Confusion matrix.

Confused Matrix		Prediction Category	
		Positive example	Negative example
Real Category	Positive example	TP	FN
	Negative example	FP	TN

In the process of intrusion detection, indicators such as accuracy (ACC), precision (PR), detection rate (DR), F1 value (F1), and false positive rate (FPR) are usually used to evaluate the effect of the model. ACC refers to the ratio of correctly classified samples to the total number of samples, and its calculation formula is shown in (11).

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

PR represents the percentage of correctly classified samples of normal traffic among the samples predicted to be normal traffic, and its calculation formula is shown in (12).

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

DR is defined as the ratio between the number of correctly identified abnormal samples and the predicted number of abnormal samples. The DR reflects the ability of the model to identify attacks, which is an essential indicator in IDSs, and

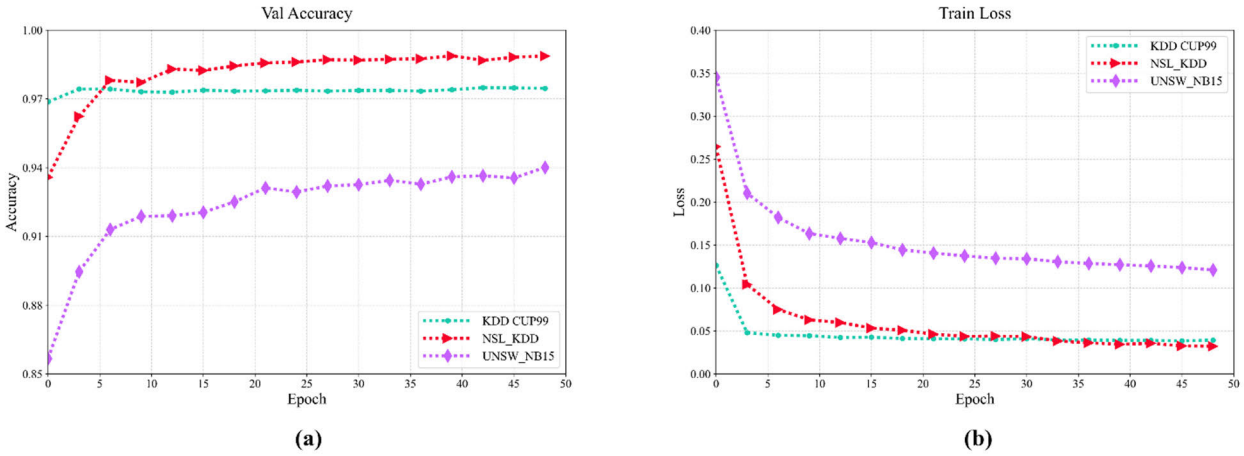

FIGURE 9. NIDS-CNNLSTM model validation accuracy and training loss in the binary classification.

its calculation formula is shown in (13).

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

F1 is the harmonic mean of PR and DR, and its calculation formula is shown in (14).

$$F1 = \frac{2PR \times DR}{PR + DR} \tag{14}$$

FPR is defined as the ratio of misidentified abnormal samples to the predicted number of normal samples, and its calculation formula is shown in (15).

$$FPR = \frac{FP}{TP + FP} \tag{15}$$

Multiple metrics are often used simultaneously in intrusion detection research to comprehensively evaluate a model. In this work, ACC, P, F, DR, and FPR were selected to assess the performance of the proposed NIDS-CNNLSTM model in multiple experiments.

C. MODEL PERFORMANCE

In order to thoroughly verify the classification performance of the NIDS-CNNLSTM model, this paper conducts experiments in binary classification and multi-classification scenarios. Use the training set in the KDD CUP99, NSL_KDD, and UNSW_NB15 datasets to train the classifier, then use the validation set to optimize and adjust the parameters of the classifier, use the test set to calculate the error rate of the optimized classifier, and test the performance of the classifier. The model activation function is set to *sigmoid*, the learning rate is 0.001, and epochs = 50. The fitting curve of validation accuracy and training loss, confusion matrix, and classification accuracy of each component are obtained.

•Binary classification experiments

In the binary classification experiment, after 50 iterations, the fitting curves of the verification accuracy and training loss of the NIDS-CNNLSTM model on the KDD CUP99, NSL_KDD, and UNSW_NB15 datasets can be obtained, as shown in Figure 9.

Figures. 9 (a) and (b) respectively show the relationship between the validation accuracy and the epoch and between the training loss and the epoch in the NIDS-CNNLSTM model. As the number of training rounds increases, the verification accuracy gradually increases and stabilizes. The verification accuracy rates on the KDD CUP99, NSL_KDD, and UNSW_NB15 datasets are 0.974, 0.99, and 0.94, respectively. At the same time, the training loss gradually decreases and finally stabilizes. The training losses on the KDD CUP99, NSL_KDD, and UNSW_NB15 datasets are 0.038, 0.029, and 0.125, respectively. The validation accuracy and training loss fitting curves show good convergence, indicating that the model's structural design and parameter settings are reasonable.

The confusion matrix is mainly used to represent classification accuracy. It uses a tabular diagram with the predicted results on the horizontal axis and the actual results on the vertical axis to display the classification performance of the algorithm visually. From the confusion matrix, the ACC, PR, DR, F1, and FPR of the normal and abnormal traffic of the NIDS-CNNLSTM model in the three sets of datasets can be obtained. The experimental confusion matrix is shown in Figure 10.

Figures 10 (a), (b), and (c) respectively show the evaluation parameters of normal and abnormal traffic in the three sets of datasets obtained from the confusion matrix and obtain the ACC, PR, DR, F1, and FPR of the NIDS-CNNLSTM model. The ACC, PR, DR, F1, and FPR of the model on the KDD CUP99 dataset are 0.97, 0.935, 0.98, 0.96, and 0.00086, respectively. The ACC, PR, DR, F1, and FPR of the model on the NSL_KDD dataset are 0.99, 0.99, 0.99, 0.99, and 0.0145, respectively. The ACC, PR, DR, F1, and FPR of the model on the UNSW_NB15 dataset are 0.94, 0.93, 0.935, 0.935, and 0.0397, respectively.

In the binary classification experiment, the NIDS-CNNLSTM model obtained the classification accuracy of

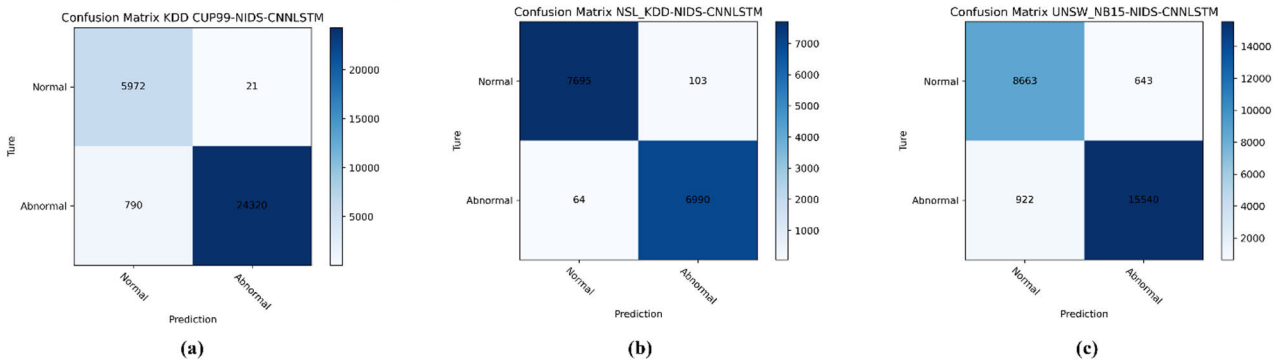

FIGURE 10. Confusion matrix in the binary classification.

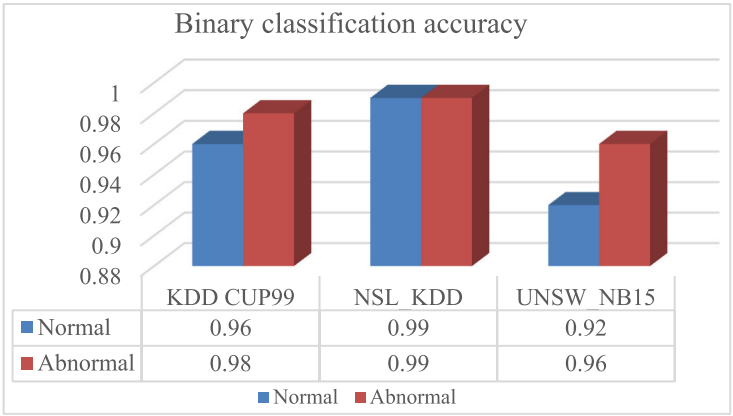

FIGURE 11. The classification accuracy of each component in the binary classification.

normal and abnormal traffic for the KDD CUP99, NSL_KDD, and UNSW_NB15 datasets, as shown in Figure 11.

As shown in Figure 11, in the binary classification scenario, the model's classification accuracy for Normal and Abnormal traffic in the three datasets is more significant than 0.9. The model has a relatively balanced classification accuracy for the Normal and Abnormal traffic in the KDD CUP99 dataset. The model has the highest classification accuracy of Normal and Abnormal traffic in the NSL_KDD dataset, reaching 0.99. The classification accuracy of the model for Normal in the UNSW_NB15 dataset is low, mainly due to the imbalance of the dataset and the limited number of training samples for Normal. It can be concluded that the NIDS-CNNLSTM model proposed in this paper has better performance in classifying two kinds of traffic in the binary classification scenario.

•Multi-classification experiments

In the multi-class experiment, after 50 iterations, the fitting curves of the verification accuracy and training loss of the NIDS-CNNLSTM model on the KDD CUP99, NSL_KDD, and UNSW_NB15 datasets can be obtained, as shown in Figure 12.

Figures 12 (a) and (b) show the relationship between the validation accuracy and the epoch and between the training loss and the epoch in the NIDS-CNNLSTM model, respectively. As the number of training rounds increases, the verification accuracy gradually increases and stabilizes. The verification accuracy rates on the KDD CUP99, NSL_KDD, and UNSW_NB15 datasets are 0.9705, 0.992, and 0.829, respectively. At the same time, the training loss gradually decreases and finally stabilizes. The training losses on the KDD CUP99, NSL_KDD, and UNSW_NB15 datasets are 0.045, 0.022, and 0.408, respectively. The validation accuracy and training loss fitting curves show good convergence, indicating that the model's structural design and parameter settings are reasonable.

From the confusion matrix, the ACC, PR, DR, F1, and FPR of the NIDS-CNNLSTM model in various types of traffic in the three sets of datasets can be obtained. The experimental confusion matrix is shown in Figure 13.

Figures 13(a), (b), and (c) respectively show the evaluation parameters of various types of traffic in the three sets of datasets obtained from the confusion matrix and obtain the ACC, PR, DR, F1, and FPR of the NIDS-CNNLSTM model. The ACC, PR, DR, F1, and FPR of the model on the KDD CUP99 dataset are 0.9705, 0.825, 0.843, 0.825, and 0.0059, respectively. The ACC, PR, DR, F1, and FPR of the model on the NSL_KDD dataset are 0.992, 0.918, 0.9, 0.908, and

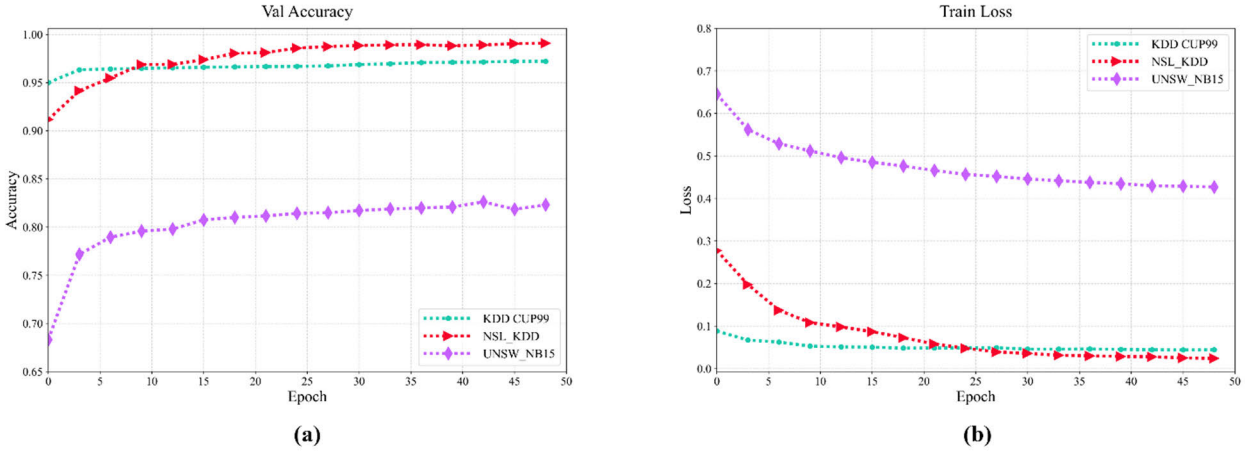

FIGURE 12. NIDS-CNNLSTM model validation accuracy and training loss in the multi-classification.

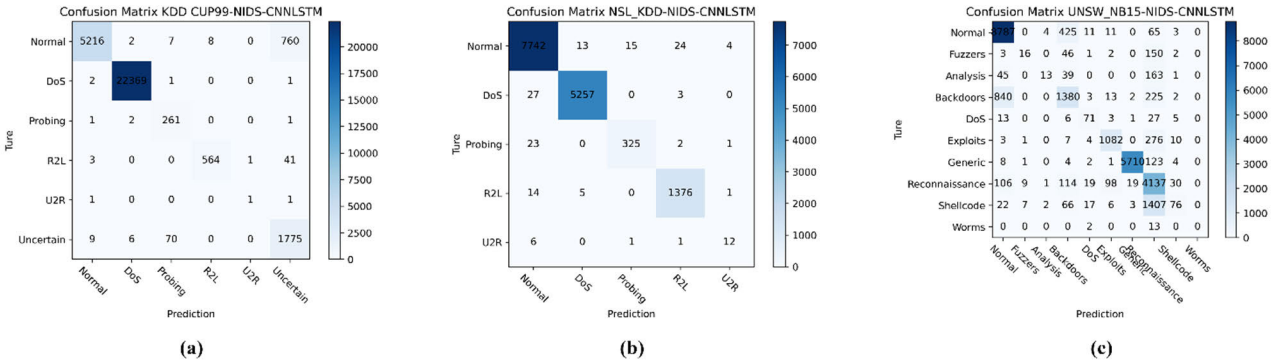

FIGURE 13. Confusion matrix in the multi-classification.

0.0029, respectively. The ACC, PR, DR, F1, and FPR of the model on the UNSW_NB15 dataset are 0.829, 0.631, 0.49, 0.495, and 0.0014, respectively.

In the multi-classification experiment, the NIDS-CNNLSTM model obtained the classification accuracy of Normal, DoS, Probing, R2L, U2R, and Uncertain six types of traffic for the KDD CUP99 dataset. For the NSL_KDD dataset, the classification accuracy of Normal, DoS, Probing, R2L, and U2R's five types of traffic is obtained. For the UNSW_NB15 dataset, the classification accuracy of Normal, Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms ten types of traffic is obtained, as shown in Figure 14.

As shown in Figure 14, in the multi-classification scenario, the classification accuracy distribution of the model for various types of traffic in the three sets of datasets is more reasonable. The model has a good classification effect on Normal, Dos, R2L, and Uncertain in the KDD CUP99 dataset, but the classification effect of Probing and U2R needs to be better. Due to the relatively small data size, the model training sample data is insufficient. The model has a low classification accuracy of 0.78 for U2R in the NSL_KDD dataset, and the classification accuracy rates of Normal, DOS, Probing, and R2L traffic types are all greater than 0.96. The model's classification accuracy for nine types of attack traffic and one type of Normal traffic in the UNSW_NB15 dataset reflects the data distribution. The UNSW_NB15 dataset is more comprehensive than the KDD99 and NSL KDD datasets. It combines normal activities and synthetic attack behaviors and can simulate the existing network environment. The NIDS-CNNLSTM model proposed in this paper performs better in classifying various traffic in multi-classification scenarios.

The experiments of binary classification and multi-classification accuracy fully demonstrate the effectiveness of the NIDS-CNNLSTM model, which is accurate and reliable.

D. COMPARATIVE EXPERIMENTS

To further evaluate the effectiveness of the NIDS-CNNLSTM model, the proposed model is compared with methods proposed in previous studies on the KDD CUP99, NSL_KDD, and UNSW_NB15 datasets. The hyperparameter settings in the comparison experiment are consistent with the experiments in this paper, and the ACC, DR, and FPR of the models

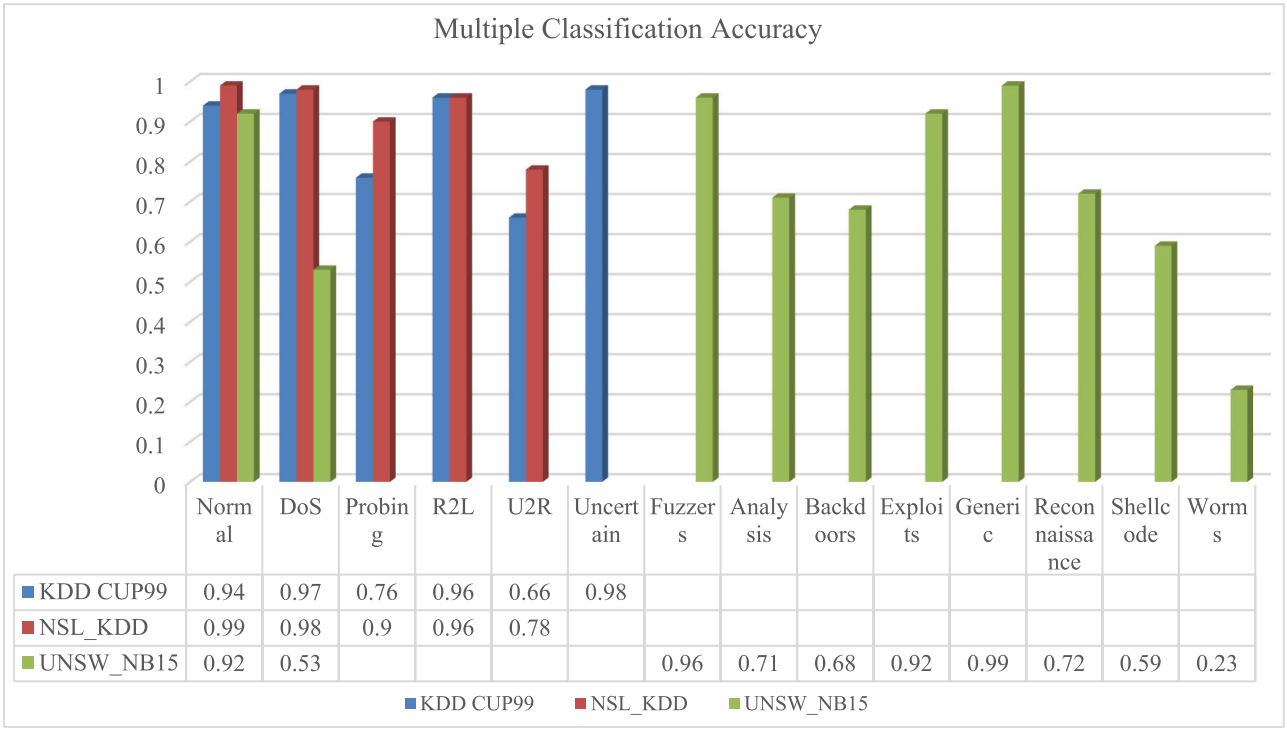

FIGURE 14. The classification accuracy of each component in the multi-classification.

TABLE 5. Performance comparison.

Systems	KDD CUP99			NSL_KDD			UNSW_NB15		
	ACC	DR	FPR	ACC	DR	FPR	ACC	DR	FPR
AE-CNN [15]	0.9399	0.7794	0.0682	/	/	/	/	/	/
LSTM [16]	0.9411	0.7707	0.0018	/	/	/	/	/	/
LSTM-RNN [16]	0.9693	0.9888	0.1004	/	/	/	/	/	/
GA-ELM [17]	0.989	0.9916	0.0136	/	/	/	/	/	/
CNN-LSTM [18]	0.997	0.996	/	/	/	/	/	/	/
CNN-LSTM [19]	**0.9995**	0.9991	**0.0003**	0.9979	**0.9992**	0.0034	/	/	/
ICNN [20]	/	/	/	0.9536	0.9699	0.0076	/	/	/
CNN [21]	/	/	/	0.9707	0.9714	0.0087	/	/	/
ELM [22]	/	/	/	0.9758	0.9769	0.0222	/	/	/
SVM [23]	/	/	/	/	/	/	0.6242	0.8858	/
ICVAE-DNN [23]	/	/	/	/	/	/	0.8908	0.9568	0.1901
DBN [23]	/	/	/	/	/	/	0.8577	**0.989**	0.3032
CNN-LSTM [23]	/	/	/	/	/	/	0.9378	0.9453	0.06
NIDS-CNNLSTM	0.9705	**0.9998**	0.0059	**0.999**	0.999	**0.0029**	**0.9443**	0.935	**0.0397**

are compared, respectively. The performance improvement effect of the NIDS-CNNLSTM model is obtained, and the comparison results are shown in Table 5.

The comparison results in table 5 show that the proposed NIDS-CNNLSTM model has significantly improved performance in terms of ACC, DR, and FPR compared with the models proposed in previous studies. Considering various evaluation indicators comprehensively, the model proposed in this paper effectively improves the detection rate and accuracy of the intrusion detection model and dramatically reduces the false positive rate. This result was also confirmed on different datasets, which may lead to different final results due to the randomness of the training set and test set selection. However, it can still prove that the proposed model is superior to the existing models proposed in previous studies. It can also more confidently adapt to network attack traffic in different scenarios. In the intrusion detection scenario, the efficiency is maximized while ensuring accuracy, and the overall performance of the intrusion detection system is improved.

V. SUMMARY

This paper proposes NIDS-CNNLSTM to solve the problems of low detection rate and classification accuracy and high

false detection rate of traditional intrusion detection models in the IIoT. In NIDS-CNNLSTM, the two-layer CNN layer and the two-layer unidirectional LSTM layer are improved and superimposed. The ability of CNN to extract spatial features and LSTM to extract temporal features is used. The model is evaluated using the KDD CUP99, NSL_KDD and UNSW_NB15 datasets, and the validation accuracy and training loss on the KDD CUP99, NSL_KDD, and UNSW_NB15 datasets show good convergence and level. The model's applicability has been verified based on binary classification and multi-classification scenarios. When classifying various types of traffic, it has high accuracy and is true and reasonable. Compared with the existing models proposed in previous studies, NIDS-CNNLSTM has significantly improved classification accuracy, detection rate and false detection rate. It is an intrusion detection classification model that is superior to existing models. In future work, we will further improve the imbalance of the dataset, improve the classification accuracy of small sample traffic, and continue to enhance the model's overall performance. NIDS-CNNLSTM can effectively support the edge wireless perception information security of the ubiquitous IIoT, thereby escorting the safe operation of key infrastructure related to the national economy and people's livelihood, such as the industrial Internet and new infrastructure, and avoiding major information security accidents. This paper has important theoretical significance and extensive practical application value.

REFERENCES

[1] S. J. Jian, Z. G. Lu, D. Du, B. Jiang, and B. X. Liu, "Overview of network intrusion detection technology," *J. Cyber Secur.*, vol. 5, no. 4, pp. 96–122, 2020.

[2] K. Wu, Z. Chen, and W. Li, "A novel intrusion detection model for a massive network using convolutional neural networks," *IEEE Access*, vol. 6, pp. 50850–50859, 2018.

[3] T. Acharya, I. Khatri, A. Annamalai, and M. F. Chouikha, "Efficacy of heterogeneous ensemble assisted machine learning model for binary and multi-class network intrusion detection," in *Proc. IEEE Int. Conf. Autom. Control Intell. Syst. (I2CACIS)*, Jun. 2021, pp. 408–413.

[4] M. Injadat, F. Salo, A. B. Nassif, A. Essex, and A. Shami, "Bayesian optimization with machine learning algorithms towards anomaly detection," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2018, pp. 1–6.

[5] Y. J. Chew, S. Y. Ooi, K.-S. Wong, Y. H. Pang, and N. Lee, "Adoption of IP truncation in a privacy-based decision tree pruning design: A case study in network intrusion detection system," *Electronics*, vol. 11, no. 5, p. 805, Mar. 2022.

[6] L. L. Ray, "Training and testing anomaly-based neural network intrusion detection systems," *Int. J. Inf. Secur. Sci.*, vol. 2, no. 2, pp. 57–63, 2013.

[7] A. Ponmalar and V. Dhanakoti, "An intrusion detection approach using ensemble support vector machine based chaos game optimization algorithm in big data platform," *Appl. Soft Comput.*, vol. 116, Feb. 2022, Art. no. 108295.

[8] M. Mehmood, T. Javed, J. Nebhen, S. Abbas, R. Abid, G. R. Bojja, and M. Rizwan, "A hybrid approach for network intrusion detection," *Comput., Materials Continua*, vol. 70, no. 1, pp. 91–107, 2022.

[9] M. U. Ilyas and S. A. Alharbi, "Machine learning approaches to network intrusion detection for contemporary internet traffic," *Computing*, vol. 104, no. 5, pp. 1061–1076, May 2022.

[10] Z. K. Maseer, R. Yusof, N. Bahaman, S. A. Mostafa, and C. F. M. Foozy, "Benchmarking of machine learning for anomaly based intrusion detection systems in the CICIDS2017 dataset," *IEEE Access*, vol. 9, pp. 22351–22370, 2021.

[11] Z. Li, A. L. G. Rios, G. Xu, and L. Trajkovic, "Machine learning techniques for classifying network anomalies and intrusions," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2019, pp. 1–5.

[12] T. Saba, A. Rehman, T. Sadad, H. Kolivand, and S. A. Bahaj, "Anomaly-based intrusion detection system for IoT networks through deep learning model," *Comput. Electr. Eng.*, vol. 99, Apr. 2022, Art. no. 107810.

[13] Y. Fu, Y. Du, Z. Cao, Q. Li, and W. Xiang, "A deep learning model for network intrusion detection with imbalanced data," *Electronics*, vol. 11, no. 6, p. 898, Mar. 2022.

[14] 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.

[15] Y. Xiao, C. Xing, T. Zhang, and Z. Zhao, "An intrusion detection model based on feature reduction and convolutional neural networks," *IEEE Access*, vol. 7, pp. 42210–42219, 2019.

[16] R. C. Staudemeyer, "Applying long short-term memory recurrent neural networks to intrusion detection," *South Afr. Comput. J.*, vol. 56, no. 1, pp. 136–154, 2015.

[17] K. Zhang, Z. Hu, Y. Zhan, X. Wang, and K. Guo, "A smart grid AMI intrusion detection strategy based on extreme learning machine," *Energies*, vol. 13, no. 18, p. 4907, 2020.

[18] R. Vinayakumar, K. P. Soman, and P. Poornachandran, "Applying convolutional neural network for network intrusion detection," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Sep. 2017, pp. 1222–1228.

[19] R. Yao, N. Wang, Z. Liu, P. Chen, and X. Sheng, "Intrusion detection system in the advanced metering infrastructure: A cross-layer feature-fusion CNN-LSTM-Based approach," *Sensors*, vol. 21, no. 2, p. 626, Jan. 2021.

[20] H. Yang and F. Wang, "Wireless network intrusion detection based on improved convolutional neural network," *IEEE Access*, vol. 7, pp. 64366–64374, 2019.

[21] G. Liu and J. Zhang, "CNID: Research of network intrusion detection based on convolutional neural network," *Discrete Dyn. Nature Soc.*, vol. 2020, pp. 1–11, May 2020.

[22] Y. Shen, K. Zheng, C. Wu, M. Zhang, X. Niu, and Y. Yang, "An ensemble method based on selection using bat algorithm for intrusion detection," *Comput. J.*, vol. 61, no. 4, pp. 526–538, 2018.

[23] A. Halbouni, T. S. Gunawan, M. H. Habaebi, M. Halbouni, M. Kartiwi, and R. Ahmad, "CNN-LSTM: Hybrid deep neural network for network intrusion detection system," *IEEE Access*, vol. 10, pp. 99837–99849, 2022.

[24] I. S. Thaseen and C. A. Kumar, "Intrusion detection model using fusion of chi-square feature selection and multi class SVM," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 29, no. 4, pp. 462–472, 2016.

[25] S. K. Sahu, D. P. Mohapatra, J. K. Rout, K. S. Sahoo, Q.-V. Pham, and N.-N. Dao, "A LSTM-FCNN based multi-class intrusion detection using scalable framework," *Comput. Electr. Eng.*, vol. 99, Apr. 2022, Art. no. 107720.

[26] S. T. Ikram, A. K. Cherukuri, B. Poorva, P. S. Ushasree, Y. Zhang, X. Liu, and G. Li, "Anomaly detection using XGBoost ensemble of deep neural network models," *Cybern. Inf. Technol.*, vol. 21, no. 3, pp. 175–188, Sep. 2021.

[27] D. Nedeljkovic and Z. Jakovljevic, "CNN based method for the development of cyber-attacks detection algorithms in industrial control systems," *Comput. Secur.*, vol. 114, Mar. 2022, Art. no. 102585.

[28] Y. Zhou, X. Zhu, S. Hu, D. Lin, and Y. Gao, "Intrusion detection based on convolutional neural network in complex network environment," in *Artificial Intelligence in China*. Springer, 2020, pp. 229–238.

[29] C. Xu, J. Shen, X. Du, and F. Zhang, "An intrusion detection system using a deep neural network with gated recurrent units," *IEEE Access*, vol. 6, pp. 48697–48707, 2018.

[30] W.-C. Shi and H.-M. Sun, "DeepBot: A time-based botnet detection with deep learning," *Soft Comput.*, vol. 24, no. 21, pp. 16605–16616, Nov. 2020.

[31] B. Chakravarthi, S.-C. Ng, M. R. Ezilarasan, and M.-F. Leung, "EEG-based emotion recognition using hybrid CNN and LSTM classification," *Frontiers Comput. Neurosci.*, vol. 16, Oct. 2022.

[32] S. Zwane, P. Tarwireyi, and M. Adigun, "Performance analysis of machine learning classifiers for intrusion detection," in *Proc. Int. Conf. Intell. Innov. Comput. Appl. (ICONIC)*, Dec. 2018, pp. 1–5.

[33] C. Kaya, O. Yildiz, and S. Ay, "Performance analysis of machine learning techniques in intrusion detection," in *Proc. 24th Signal Process. Commun. Appl. Conf. (SIU)*, May 2016, pp. 1473–1476.

JIAWEI DU received the B.E. degree from Xijing University, Xi'an, Shaanxi, China, in 2021, where she is currently pursuing the master's degree in electronic information. Her research interest includes network and information security.

KAI YANG received the B.S. and Ph.D. degrees in computer science and technology from Xidian University, in 2005 and 2011, respectively. He is an Associate Professor with the School of Computer, Xijing University, Xi'an, Shaanxi, China. Since 2011, he has been a Lecturer and an Associate Professor with the Engineering University of PAP. His research interest includes network and information security.

YANJING HU received the B.S. degree in computer science and technology from Xi'an University of Technology, in 2009, and the Ph.D. degree in computer science and technology from Xidian University, in 2017. He is an Associate Professor with the School of Cryptographic Engineering, Engineering University of PAP, Xi'an, Shaanxi, China. Since 2003, he has been a Lecturer and an Associate Professor with the Engineering University of PAP. His research interest includes network and information security.

LINGJIE JIANG received the bachelor's degree in engineering from Xijing University, Xi'an, Shaanxi, China, in 2021, where he is currently pursuing the master's degree in electronic information. His research interests include deep learning and object detection.