ITC 2/55
Information Technology and Control
Vol. 55 / No. 2/ 2026
pp. 553-570
DOI 10.5755/j01.itc.55.2.43421

Security Detection Method for Improved DenseNet 1D in the Internet of Things Environment

Received 2025/10/24 | Accepted after revision 2026/01/25

HOW TO CITE: Yang, F. (2026) Security Detection Method for Improved DenseNet 1D in the Internet of Things Environment. *Information Technology and Control*, 55(2), 553-570. https://doi.org/10.5755/j01.itc.55.2.43421

Security Detection Method for Improved DenseNet 1D in the Internet of Things Environment

Fang Yang
School of Computer Engineering, Shanxi Vocational University of Engineering Science and Technology, Jinzhong, 030619, China

Corresponding author: yougwu@outlook.com

The continuous expansion of the scale of the IoT has led to frequent occurrence of network attack events and posed severe challenges to device operation and data security. To accurately track the behavior of IoT devices in the network and detect any malicious activities that may occur, this study proposes a spatiotemporal feature-based IoT security detection method that combines TPA mechanism, BiLSTM, DenseNet 1D, and CNN. The experiment was validated using two publicly available datasets, UNSW-NB15 and Bot-IoT. The research method could effectively identify various types of attacks (such as user to administrator privilege escalation attacks, remote to local attacks, etc.), alleviate the impact of imbalanced dataset categories, and maintain low online detection latency. In the UNSW-NB15, the precision, recall, and F1 value of the research method reached 97.6%, 96.8%, and 97.2%, while in the Bot-IoT, they reached 95.7%, 94.1%, and 94.9%. When the sample size increased to 1×10^6, the running time and energy consumption of the research method were only 13.2 s and 0.9 kWh. The research method effectively enhances the ability to identify multiple types of attack traffic, providing a technical solution that balances accuracy and robustness for malicious behavior detection in IoT environments.

KEYWORDS: Internet of things; One-dimensional dense connected convolutional network; Convolutional neural network; Data balancing processing; Security testing

1. Overview

The Internet of Things (IoT) is a system composed of sensors, software, and communication devices. It can connect various devices and the Internet, and realize smart home, smart city, industrial automation, agricultural monitoring, intelligent transportation and other applications through data exchange and analysis [9, 28]. However, the increasing complexity of IoT environments has led to the emergence of ma-

licious software attacks. Traditional security detection methods are mostly based on shallow models or statistical feature analysis, which makes it difficult to capture the complex nonlinear relationships and temporal dependencies in IoT data, resulting in low accuracy in identifying Malicious Code (MCode) [10, 13]. The advancement of Deep Learning (DL) technology has led many researchers to gradually introduce methods, such as Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), and Graph Neural Network (GNN), into network security detection tasks [12, 15].

Om Kumar et al. utilized Recurrent Kernel CNN (RKCNN) and Modified Monarch Butterfly Optimization (MMBO) algorithm to identify attacks in the network. The RKCNN-MMBO model had an intrusion detection classification precision of 99.95% in the CICIDS-2017 dataset [22]. Al Kahtani et al. used Gated Recurrent Units (GRUs) and LSTM to identify malicious traffic. Even in the face of multiple network attacks, the GRU-LSTM model still achieved a malicious traffic recognition accuracy of 98.86% [3]. Saravanan et al. took a Recurrent Neural Network (RNN) model to detect network intrusions in cloud environments, and introduced the African Buffalo Optimization algorithm in the RNN prediction stage to continuously monitor MCode. The accuracy and recall of MCode detection using this method reached 99.87% and 99.92% [25]. Wang et al. utilized a Generative Adversarial Network (GAN) model for data augmentation, generating more samples of MCode variants, and combined CNN to classify malicious software. Then, they detected malicious software based on the difference in information flow between malicious and benign applications. The GAN-CNN model achieved an accuracy of 97.78% in classifying malicious software [29]. Lu et al. proposed an industrial Internet of Things network security detection method based on Multi-Objective Differential Evolution Optimized (MODEO) and CNN. The experimental results show that the overall performance of the MODEO-CNN model is significantly better than that of traditional deep learning models, and the detection accuracy is increased by an average of approximately 3% [17]. To further enhance the generalization ability of the Intrusion Detection System (IDS) under complex sample conditions, Zeng G Q et al. proposed an automatic adversarial unsupervised anomaly detection method based on deep learning (EvoAAE). The experimental results show that the accuracy rate, recall rate and F1 value of EvoAAE have reached 0.949, 0.971 and 0.960 respectively [31].

However, IoT traffic data not only contains time series information but also spatial information such as flow features and content features [7]. Therefore, it requires good before and after dependency in the transmission between features. In terms of spatial features, Dense Connected Convolutional Networks (DenseNet) can reuse underlying features and extract deeper features [27]. Morshedi et al. addressed the co-adaptation problem caused by information flow by introducing a compression layer into the DenseNet structure. This method could accurately distinguish multiple types of encrypted Internet traffic [1]. Anandhi et al. visualized the malware as Markov images and extracted its texture features using Gabor filters. They then combined Visual Geometry Group-3 (VGG-3) and DenseNet for recognition. The classification accuracy on the Malimg and BIG2015 datasets reached 99.94% and 98.98% [6]. In response to the shortcomings of existing DL-based MCode classification methods in feature extraction, Li et al. proposed a Red-Green-Blue (RGB) visualization detection method built on a hybrid multi-head attention mechanism and DenseNet. The recognition accuracy on Kaggle and DataMen datasets was 99.49% and 97.70% [16]. Pandey et al. used Frost filtering and DenseNet-201 to denoise and enhance the original image, and segmented the preprocessed image using a context encoder-decoder network. They proposed a copy move image forgery detection method. Even in noisy images, the accuracy of malicious activity recognition using this method still reached 97.39% [23].

In summary, MCode attacks seriously hinder the development of current IoT technology. Once a device is infected with a virus, the losses caused to businesses and users are unpredictable. Some researchers have developed various safety detection methods. Deep learning methods can automatically extract complex features, improving detection accuracy and generalization ability. Data augmentation technology can expand sample diversity and enhance the ability to identify new types of malicious code. The combined algorithm can further enhance the model performance and improve the convergence of some traditional methods. However, deep networks are prone to vanishing or exploding gradients, and their training stability is insufficient.

Meanwhile, due to the complexity of malicious code and the rapid growth of variants, the analysis results of existing methods still fail to meet the expected accuracy. To address this issue, this study innovatively combines the Temporal Pattern Attention (TPA) mechanism with Bidirectional LSTM (BiLSTM) to propose a TPA-BiLSTM model. Secondly, in response to the problem of gradient vanishing or exploding that is prone to occur in deep networks, this study further designs an IoT security detection method based on spatiotemporal features and DenseNet 1D-CNN. The above design aims to achieve efficient and accurate identification of MCode in IoT environments, provide a feasible solution for the security protection of IoT systems, and lay a theoretical and practical foundation for the subsequent application and promotion in complex network environments. The overall structure consists of three sections: The first section designs IoT security detection methods named TPA-BiLSTM and DenseNet 1D-CNN. The second section tests and analyzes the proposed method. The third section summarizes the experimental results and indicates future research directions.

2. Methodologies

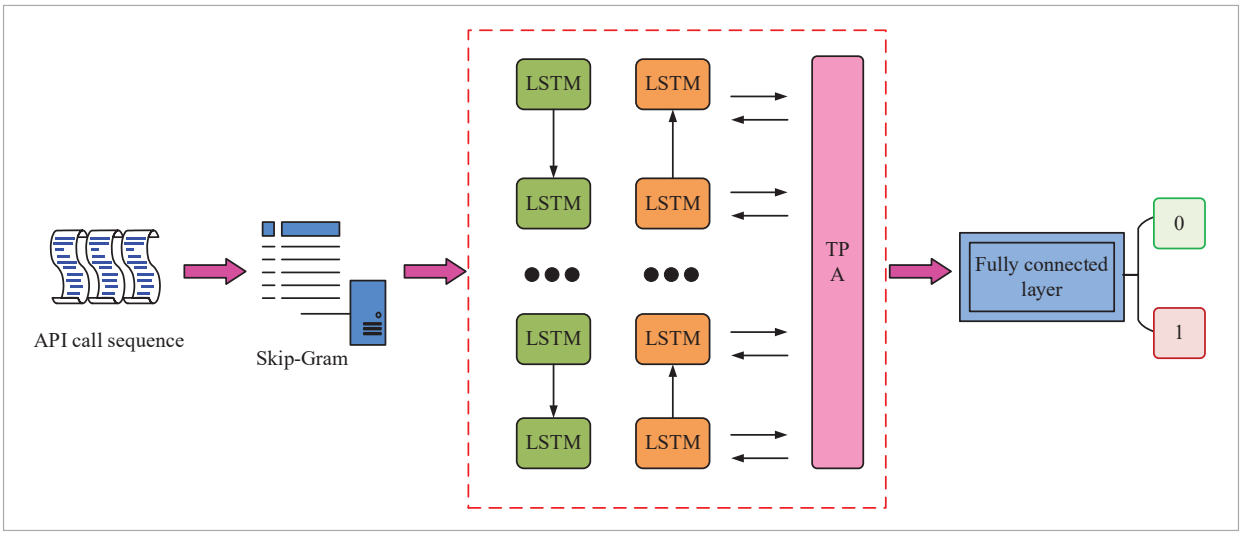
In response to the time dependence and spatiotemporal feature extraction challenges faced by MCode attack detection in IoT environments, this study first proposes a TPA-BiLSTM model, which uses TPA to weight and select key variables, achieving efficient representation of time patterns. Subsequently, this study further constructs the DenseNet 1D-CNN method. It takes the temporal feature matrix output by TPA-BiLSTM as input, extracts local feature patterns through horizontal 1D-CNN, and integrates the entire sequence features through vertical 1D-CNN to achieve deep fusion of temporal and spatial features.

2.1. TPA-BiLSTM Model

MCode attacks usually have a temporal sequence, so the Application Programming Interface (API) functions called during execution also have a clear chronological order [2, 4]. Based on this characteristic, this study proposes a TPA-BiLSTM model, whose overall framework is shown in Figure 1.

In Figure 1, the model mainly consists of a word embedding feature vectorization part and a TPA BiLSTM model classification part. The former adopts Skip-Gram model, which transforms the originally symbolized API call sequence into a continuous low dimensional dense vector by learning the probability relationship between context words and target words. The sequence data processed through this step can more intuitively reflect the inherent connections of MCode in the context, providing stable and discriminative input features for subsequent

Figure 1
The Structure of the TPA-BiLSTM.



classification. The latter processes the text features of the MCode after vectorization. The forward LSTM model learns historical state features, the backward LSTM model learns future state features, and the final output is determined by both. Subsequently, the TPA module uses a scoring function to measure the correlation between various hidden states in the sequence, thereby distinguishing the contribution of different variables in the classification task. It then weights and combines each feature based on the calculated weights, allowing important variables to occupy a higher proportion in the final representation. The influence of irrelevant or noisy features is weakened. Through this process, the model can generate more discriminative feature expressions and complete classification based on them, effectively improving the accuracy and robustness of overall prediction. The calculation formula for the BiLSTM model is shown in Equation (1).

$$\begin{cases} g_t = \sigma(W_g x_t + U_g h_{t-1} + b_g) \\ r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \\ h_t^* = \tanh(W x_t + U(r_t \times h_{t-1}) + b_h) \\ h_t = (1 - g_t) \times h_{t-1} + g_t \times h_t^* \\ y_t = \sigma(W_O \times h_t) \end{cases} \quad (1)$$

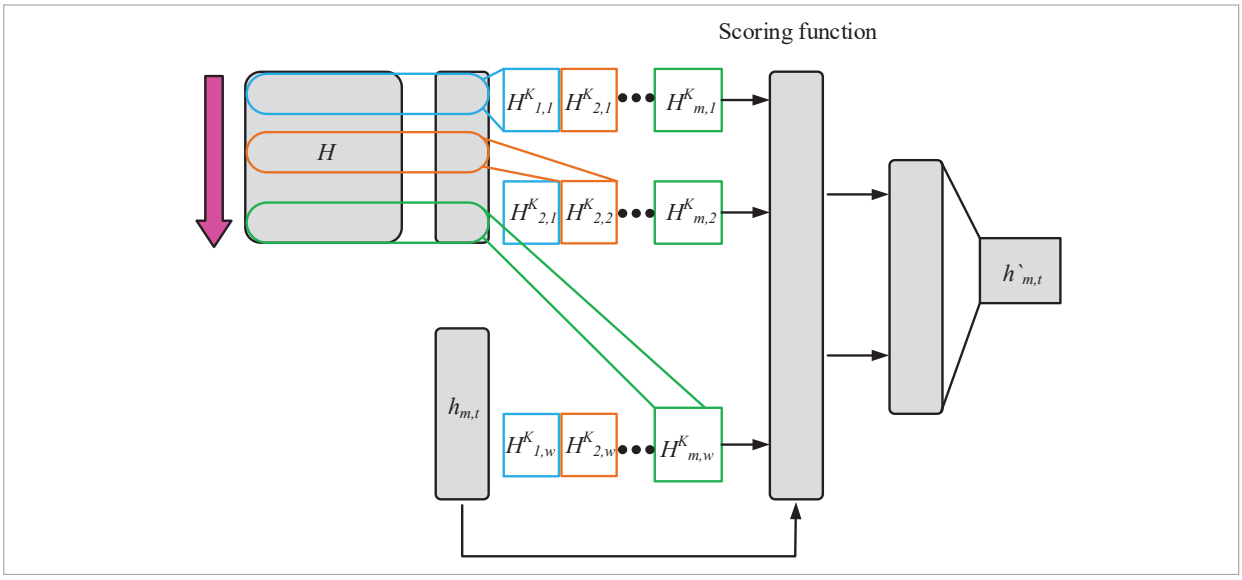
In Equation (1), g_t, r_t, h_t^*, h_t, and y_t represent the update gate output value, reset gate output value, candidate hidden state, hidden state, and output data at time t. σ denotes a Sigmoid function within the range of [0,1]. tanh means the hyperbolic tangent activation function. W, W_g, W_r, W_O, U, U_g, and U_r are all neural network weight matrices. b_g, b_r, and b_h are both bias terms. x_t and h_{t-1} are the input data at the current t and the hidden unit information output at the previous time $t-1$. TPA mainly has a loop layer, a convolution layer, and a temporal pattern attention layer, as shown in Figure 2.

In Figure 2, firstly, the loop layer inputs the preprocessed MCode sequence features into the BiLSTM model and outputs the hidden state matrix as the input $H = \{h_{m,t-w}, h_{m,t-w+1}, \cdots, h_{m,t-1}, h_{m,t}\}$ of TPA. Then, the convolution layer performs convolution operations on each row i of the hidden state matrix to extract potential signal features and construct a Time Pattern Matrix (TPM) $H_{i,j}^K$, as shown in Equation (2).

$$H_{i,j}^K = \sum_{l=1}^{w} H_i(t-w-l+1) \cdot K_{j,T-w+l}. \quad (2)$$

In Equation (2), row i corresponds to the dynamic changes of a single feature at different time steps, while column j represents the state of all features.

Figure 2
Schematic Diagram of TPA Structure.



K is the amount of convolution kernels, and T denotes the maximum length of attention. w and l are the convolution window size and input sequence length. m is the dimension output by BiLSTM. In the temporal pattern attention layer, the row vectors of the temporal pattern matrix will interact with the corresponding hidden state vectors $h_{m,t}$, and the calculation results will be normalized using Sigmoid to obtain weights λ_i representing the strength of the influence of each time series on it, as shown in Equation (3).

$$\lambda_i = Sigmoid(f(H_i^K, h_{m,t})). \tag{3}$$

In Equation (3), $f(H_i^K, h_{m,t})$ is the interaction function between the i-th row of the TPM and the latent vector $h_{m,t}$. Next, weighted summation was performed on each row of the TPM to obtain the context variable R_t that comprehensively considers the impact of time, as shown in Equation (4).

$$R_t = \sum_{i=1}^{m} \lambda_i H_i^K . \tag{4}$$

The context variable is fused with the output $h_{m,t}$ of BiLSTM, and the interaction relationship between time series is embedded into the fused representation through linear mapping operation, thereby generating the final output result X_t, as given by Equation (5).

$$\begin{cases} X_t = W_{h'} h'_{m,t} \\ h'_{m,t} = W_h h_{m,t} + W_v R_t \end{cases}. \tag{5}$$

In Equation (5), $h'_{m,t}$ is the fused output latent vector. W_v and W_h are the weight matrix of the fusion stage and temporal pattern attention layer. Finally, X_t is input into the Softmax to complete binary classification discrimination. This function normalizes the posterior probabilities of categories to reflect the probability distribution of samples belonging to each category, thereby achieving effective differentiation of input data [21]. The model optimization and training phase mainly involves parameter settings for Skip-Gram and TPA-BiLSTM models, as listed in Table 1.

Table 1
Parameter setting.

Models	Parameters	Value
Skip-Gram	The output dimension of the word vector	500
	Number of parallelism in the training process	5
	The corpus will be discarded if its word frequency is less than this number	10
TPA-BiLSTM	The number of neurons	64
	Determine the output dimension	64
	Hidden unit	32
	Filter size	3
	The time step considered by the input value	100
	The sequence number of each time step	105
	Convolutional channel	1
	Convolution step size	1

In Table 1, a higher word vector dimension is selected in the Skip-Gram model to ensure a more comprehensive characterization of the semantic information of API calls, while a threshold for discarding low-frequency words is set to reduce noise interference. In the TPA-BiLSTM model, the hidden units are 32 and the convolution kernel size is 3 to balance feature extraction capability and computational complexity. The input time step is 100, ensuring that the model can cover a sufficiently long call sequence context. The convolutional channel and stride are both set to 1 to extract fine-grained temporal pattern features.

2.2. DenseNet 1D-CNN Method

Although the TPA-BiLSTM model can effectively capture the temporal characteristics of API call sequences, considering only the temporal characteristics is still insufficient to fully reflect malicious attack behavior. In IoT security detection scenarios, in addition to MCode behavior sequences, there are also spatial features such as network traffic characteristics, protocol interaction patterns, signal amplitude

distribution, and sensor outputs [32, 8]. Therefore, based on TPA-BiLSTM, this study further proposes a DenseNet 1D-CNN method. The traditional DenseNet model was proposed to extract 2D matrix data in the video and image domains. It uses 2D convolution to extract and classify features from input data, which can solve the problem of traditional deep neural networks being unable to fully learn low-level data features [20]. However, the vibration signals and network traffic sequences generated by IoT devices are all 1D temporal signals related to security. If 2D convolution is directly used for processing, it will not only introduce irrelevant parameter redundancy, but also make it difficult to effectively capture key temporal features and dependencies in the sequence, resulting in a decrease in detection performance [26, 18]. Therefore, the DenseNet model used in this study is mainly a 1D network structure. DenseNet 1D enables feature reuse and transfer by connecting the outputs of each layer with those of all previous layers [14]. The DenseNet 1D network structure is shown in Figure 3.

In Figure 3, DenseNet 1D replaces the traditional three channel input with a single channel and replaces 2D convolution and pooling with 1D convolution and 1D pooling [5, 30]. The length of the convolution kernel in the first layer is 3 and the stride is 1. DenseNet 1D mainly has two parts: dense connection blocks and transition layers, which are connected through channel merging [24]. The dense connection block structure is displayed in Figure 4.

Figure 3

Schematic diagram of DenseNet 1D network structure.

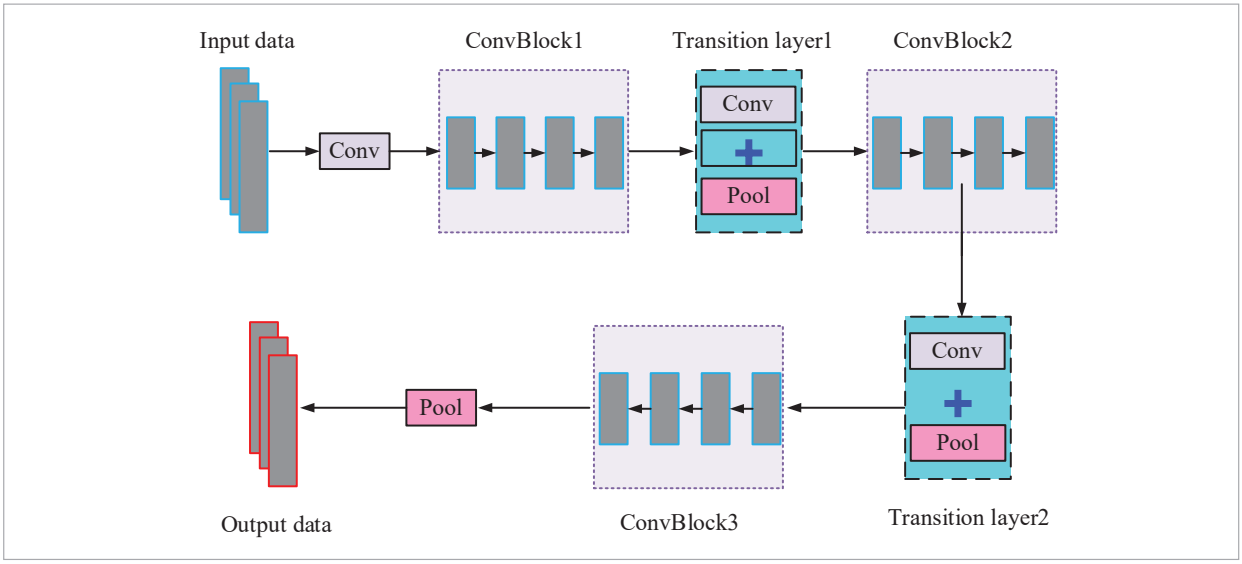
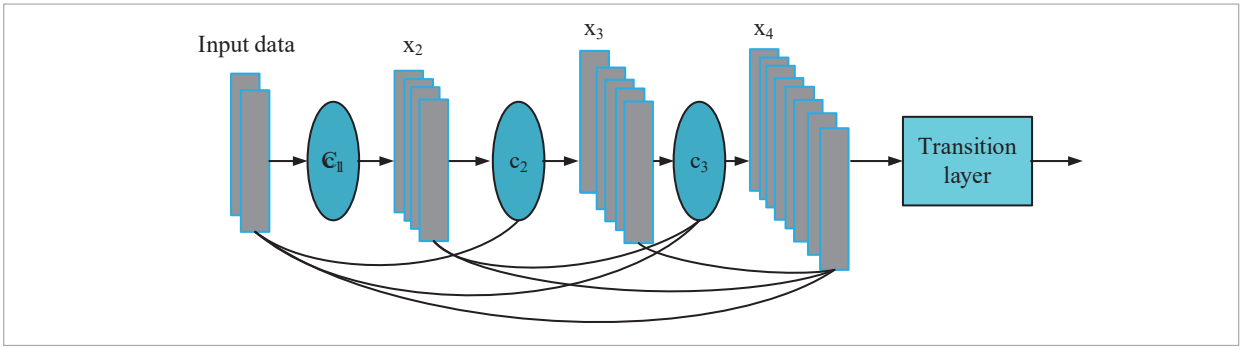


Figure 4

The dense connection block structure.



In Figure 4, each dense connection block is composed of multiple stacked dense connection layers. The input data are sequentially subjected to Batch Normalization (BN), pooling, activation function, and 1D convolution calculation for each layer. The output of the current layer is concatenated with the features of all previous layers before being passed on to subsequent layers. This feature reuse strategy can be expressed as Equation (6).

$$x_c = H_c([x_0, x_c, ..., x_{c-1}]). \quad (6)$$

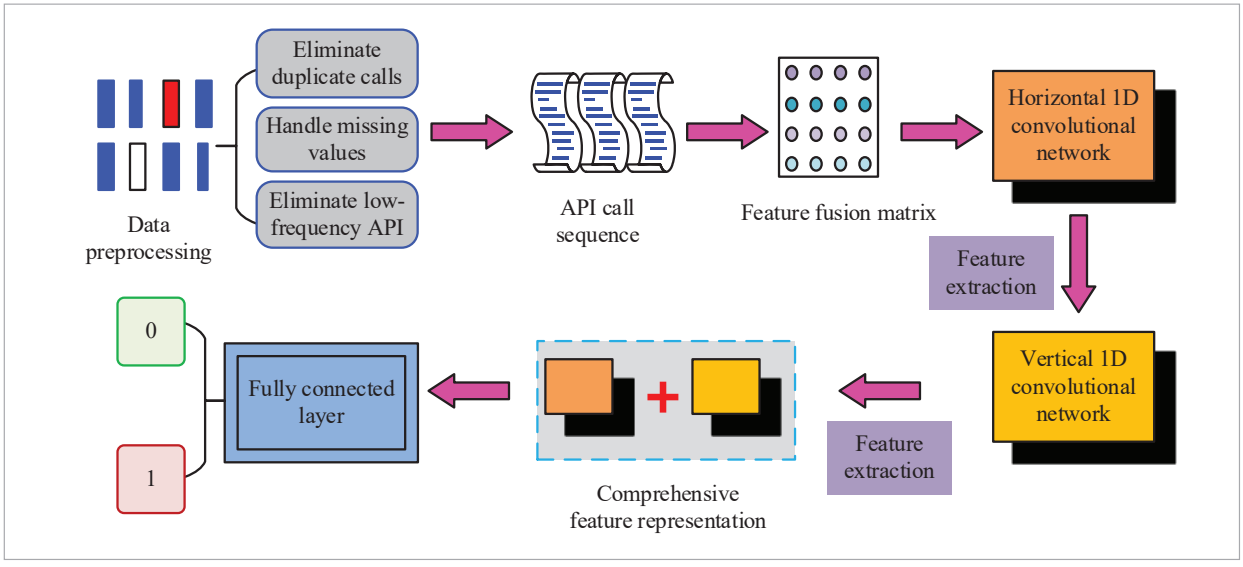
In Equation (6), x_c is the output of the c-th layer. x_0 is the input data. After the feature reuse strategy, a certain layer will generate multiple features, as shown in Equation (7).

$$y = n(c-1) + n_0. \quad (7)$$

In Equation (7), y is the input feature. n is the probability coefficient that controls the increase in the amount of feature channels. n_0 is the number of input layer feature channels. The transition layer is located between adjacent densely connected blocks and consists of 1D convolution and pooling operations with a kernel length of 1. It is used to compress the number of channels, reduce computational complexity, and alleviate overfitting. In response to the problem of gradient vanishing or exploding in deep networks, this study combines 1D convolution and dense structure to construct a DenseNet 1D-CNN model. The framework of this model is shown in Figure 5.

In Figure 5, the DenseNet 1D-CNN model mainly consists of a data preprocessing module (M1), a feature fusion matrix construction module (M2), and a classification module (M3). M1 first cleans the original API call sequence, including removing duplicate calls, handling missing values, and removing low-frequency APIs to reduce noise interference. Subsequently, each API call is encoded and the discrete API names are mapped to continuous feature vectors. At the same time, the time series are standardized by scaling the features of each API call to the same numerical range to enhance the stability and convergence speed of model training. For data of different sequence lengths, a fixed time step length is used for truncation or zero padding to ensure consistent input matrix dimensions. After the data preprocessing is completed, the API call sequence is inputted, and a feature fusion matrix is constructed based on the API features. Subsequently, the matrix is input into both horizontal and vertical 1D-CNN for convolutional feature extraction. The output of two types of convolutions is added to obtain a comprehensive feature representation, which is then input into a Fully Connected Layer (FCL) for

Figure 5
The overall structural framework of the DenseNet 1D-CNN model.

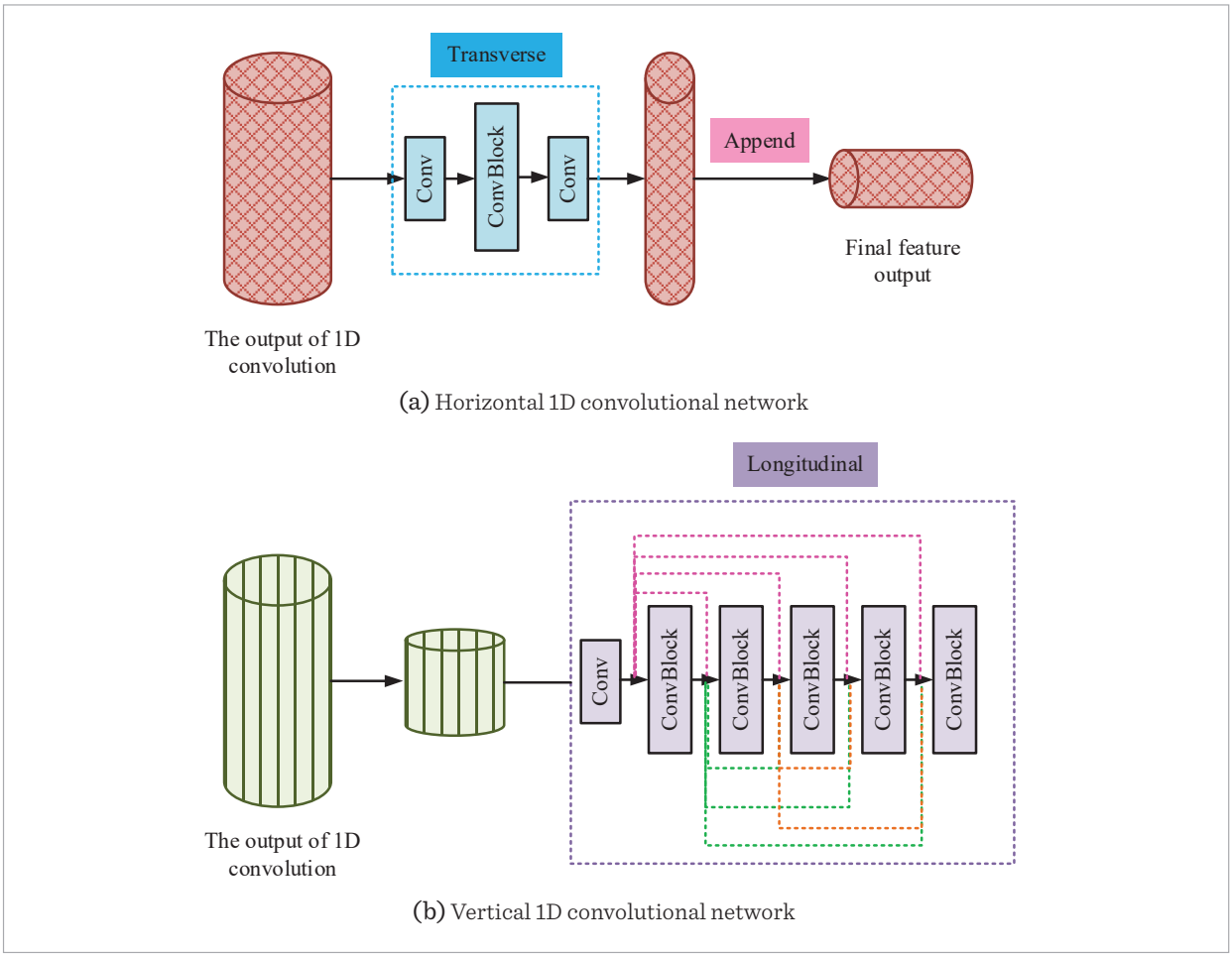


binary classification, thereby distinguishing the samples into normal samples (0) or malicious samples (1). The horizontal and vertical convolution processing procedures are shown in Figure 6.

In Figure 6(a), the horizontal 1D-CNN uses nested skip connections to enhance feature transfer. It first inputs the output of 1D convolution into the convolution block for local feature extraction. Subsequently, through skip connections, the output of the 1D convolution and the output of the convolution block are added element by element as inputs to the subsequent convolutional layers. By maintaining continuous information transmission between layers and avoiding gradient decay, a stable feature representation is ultimately formed. In Figure 6(b), the vertical 1D-CNN adopts a dense structure, where the outputs of all previous layers are added element by element as inputs for each layer, achieving sufficient feature reuse and information integration. This structure also uses convolutional blocks as the basic unit. The input data is extracted through convolution and merged with the previous result before being sent to the next convolutional layer. After multi-level stacking and fusion, the data are fed into the final convolutional layer to form a stable feature representation. The formula for extracting horizontal 1D convolution features is shown in Equation (8).

$$Y_i^c = f_1 \sum_{j=0}^{K-1} W_j \cdot X_{i+j}^{c-1} + b \,. \tag{8}$$

Figure 6
Convolution processing procedure.



In Equation (8), Y_i^c is the output of the c-th layer of the lateral 1D-CNN at position i. X_{i+j}^{c-1} is the input of layer $c-1$ at position $i+j$. W_j is the convolution kernel weight, b is the bias term, and f_1 is the activation function. The vertical 1D convolution feature extraction formula is shown in Equation (9).

$$Y_c^v = f_1(W_c * \sum_{i=0}^{c-1} X_i^v) + b\,. \tag{9}$$

In Equation (9), Y_c^v is the output of the c-th layer of the vertical 1D-CNN. X_i^v is the input feature of position i. W_c is the convolution kernel weight of the c-th layer. The model optimization and training phase mainly involves setting up the DenseNet 1D-CNN model, as exhibited in Table 2.

The overall process of the DenseNet 1D-CNN method is as follows: TPA-BiLSTM first extracts temporal features from the API call sequence and outputs the hidden state matrix for each time step. Subsequently, the hidden state matrix of TPA-BiLSTM is used as the input feature matrix for DenseNet 1D-CNN to further learn spatial features. DenseNet 1D-CNN utilizes dense connection blocks and transition layers to capture local feature patterns through horizontal 1D-CNN, and integrates full sequence features through vertical 1D-CNN, thereby enhancing spatiotemporal feature representation capabilities. Finally, the horizontal and vertical 1D-CNN outputs are fused with the temporal features extracted by TPA-BiLSTM, and binary classification is performed through a FCL to achieve high-precision detection of malicious behavior in IoT environments. This study evaluates the performance of the detection method utilizing indicators including *Accuracy*, *Precision*, *Recall*, and $F1$ value, as shown in Equation (10) [11, 19].

Table 2
Parameter setting.

Network	Parameters	Value
Horizontal 1D-CNN	Kernel size	3
	Stride	1
	Number of convolution blocks	5
	Number of channels	64
	Activation function	ReLU
	BN	Enabled
Vertical 1D-CNN	Kernel size	3
	Stride	1
	Number of convolution blocks	5
	Number of channels	64
	Activation function	ReLU
	Batch Normalization	Enabled
	Input time steps	100
	Sequence length per step	105

$$\begin{cases} Accuracy = \dfrac{TP+TN}{TP+TN+FP+FN} \\ Precision = \dfrac{TP}{TP+FP} \\ Recall = \dfrac{TP}{TP+FN} \\ F1 = \dfrac{Precision \times Recall}{Precision + Recall} \times 2 \end{cases} \tag{10}$$

In Equation (10), TP and TN represent the number of abnormal and normal samples. FP/FN is the number of normal/abnormal samples predicted as abnormal/normal samples. In terms of hyperparameter selection, the Settings of parameters such as the number of hidden units, the size of convolution kernels, the number of convolution layers, and the time step length were initially referred to the Settings in sequence tasks in relevant literature, and empirical optimization was carried out in combination with small-scale experiments. To verify the effectiveness of the model design, the performance of LSTM, BiLSTM and TPA-BiLSTM in terms of accuracy, recall rate, F1 value and average prediction delay was compared in the study to ensure that the selected architecture and hyperparameters performed stably under different datasets and imbalanced sample conditions. Meanwhile, the number of convolutional layers, the number of channels and the batch size were tested in the GPU parallel acceleration environment to ensure low latency and controllable energy consumption under the condition

of high sample size. Thus, the architectures and hyperparameter configurations of TPA-BiLSTM and DenseNet 1D-CNN were finally determined to ensure that the models can fully capture both temporal and spatial features. It also features training stability and reasoning efficiency.

3. Results

To verify the performance of TPA-BiLSTM and DenseNet 1D-CNN methods, this paper first establishes a suitable environment. Secondly, simulation tests are conducted on the proposed model using methods such as ablation testing, comparative testing of the same type, and multi-index testing.

3.1. Performance Testing of TPA-BiLSTM Model

This study utilizes the UNSW-NB15 and Bot-IoT datasets as data sources. UNSW-NB15 was constructed by the Australian Centre for Cyber Security (ACCS) at the University of New South Wales in 2015, and includes normal traffic and attack traffic (Fuzzers, Analysis, Backdoor, Exploits, Generic, Reconnaissance, Shellcode, Worms). There are approximately 560,000 normal samples and 1,580,000 attack samples, with a rich distribution of categories that can comprehensively reflect different types of network behavior. Bot-IoT was built by the Australian Defence Force Academy (ADFA) in 2018, mainly for large-scale botnet attack data in IoT environments, including Denial-of-Service (DOS), Distributed DOS (DDoS), information theft, Keylogging, and data penetration. Among them, the normal sample size is about 477,000, far less than the attack sample, showing obvious imbalance characteristics. This feature provides an ideal validation scenario for testing the performance and robustness of the model under extreme data imbalance conditions. To fully capture the time-dependent characteristics in the API call sequence, this study uniformly divides the API call sequence in the dataset into long sequences (200 calls), and based on this, systematically tests the performance of the model. Table 3 shows the API call sequence running environment.

The hardware acceleration process is as follows: First, the sequence of API calls to be processed is loaded into the GPU memory in batches to reduce the data transmission overhead between the CPU and the GPU. The lateral and vertical convolution of DenseNet 1D-CNN is achieved through CUDA thread blocks for parallel acceleration. Meanwhile, Tensor Cores are utilized to efficiently calculate 1D convolution and matrix operations, thereby enhancing the speed of feature extraction. In the TPA-BiLSTM model, the hidden state update of bidirectional LSTM and the calculation of attention weights are

Table 3
Experimental environment for API call sequence execution.

Serial number	Environment	Configuration
1	Processor	Intel(R) Xeon(R) Gold 6226R CPU @ 2.90GHz×16
2	Operating system	Ubuntu 20.04 LTS (64-bit)
3	Memory	128 GB DDR4
4	Software stack	Python 3.9, TensorFlow 2.11, PyTorch 1.13, Scikit-learn 1.2
5	Guest OS	Windows 10 (64-bit)
6	Guest Software	Wireshark 3.6, Nmap 7.92, Metasploit 6.1
7	Virtual platform	VMware Workstation Pro 16
8	Database system	MySQL 8.0 for data storage and management
9	Network setting	Gigabit Ethernet (1 Gbps), isolated laboratory network
10	Hardware accel	NVIDIA Tesla V100 GPU (32 GB VRAM), CUDA 11.7, cuDNN 8.6

completed in parallel on the GPU, reducing the latency of time-dependent computations. Ultimately, after the model completes the inference on the GPU, the prediction results are immediately sent back to the CPU side, achieving real-time monitoring of abnormal behaviors in the API call sequence. According to Table 3, the impact of various word embedding models on the performance of TPA-BiLSTM is first explored, as shown in Figure 7.

Figures 7(a)-(b) show the changes in loss values for different word embedding models in UNSW-NB15 and Bot-IoT. In Figure 7(a), different models in UNSW-NB15 show a gradual decrease in loss values during training, but there are significant differences in convergence speed and final convergence effect. The Word to Vector (Word2Vec), Global Vectors for Word Representation (GloVe), and Fast Text Representation for Words (FastText) models converge to loss values of 0.095, 0.143, and 0.195 at round 30 of iteration. Skip-Gram experiences a rapid decrease in loss value during the initial training phase and stabilizes after approximately 15 iterations. The final convergence loss value is the lowest, only 0.028, indicating its ability to better capture contextual semantic relationships in API call sequences. In Figure 7(b), in Bot-IoT, Skip-Gram still performs the best, with the fastest convergence speed of loss values during training, and ultimately stabilizes at 0.052, verifying its strong robustness and adaptability under imbalanced data conditions. Based on the results of the two sets of experiments, it can be known that the Skip-Gram model has the best effect in extracting semantic information and context dependence of IoT malicious behavior data. However, other word embedding models are inferior to Skip-Gram in terms of training convergence speed and final loss value due to insufficient sensitivity to long-tail features of sequences or overly sparse global vectors. This leads to relatively weak performance in IoT malicious code detection tasks. Therefore, the adoption of the Skip-Gram model can provide higher-quality input features for the subsequent time feature modeling of the TPA-BiLSTM model, thereby improving the overall detection performance. In addition, this study also compares the classification accuracy of LSTM and BiLSTM as benchmark models, as shown in Figure 8.

Fig.8 shows the accuracy of the model in two datasets. In the UNSW-NB15 of Fig.8 (a), the performance of the traditional LSTM model is limited by the unidirectional information flow, making it difficult to fully capture the bidirectional dependency relationship between the preceding and following sequences, and thus it underperforms in the feature extraction of long sequences. Its accuracy rate for detecting long sequence attacks after training is only 69.1%. The BiLSTM model, by introducing bidirectional information transmission, fuses the forward and backward context information of the sequence, increas-

Figure 7

The variation results of loss values in different word embedding models.

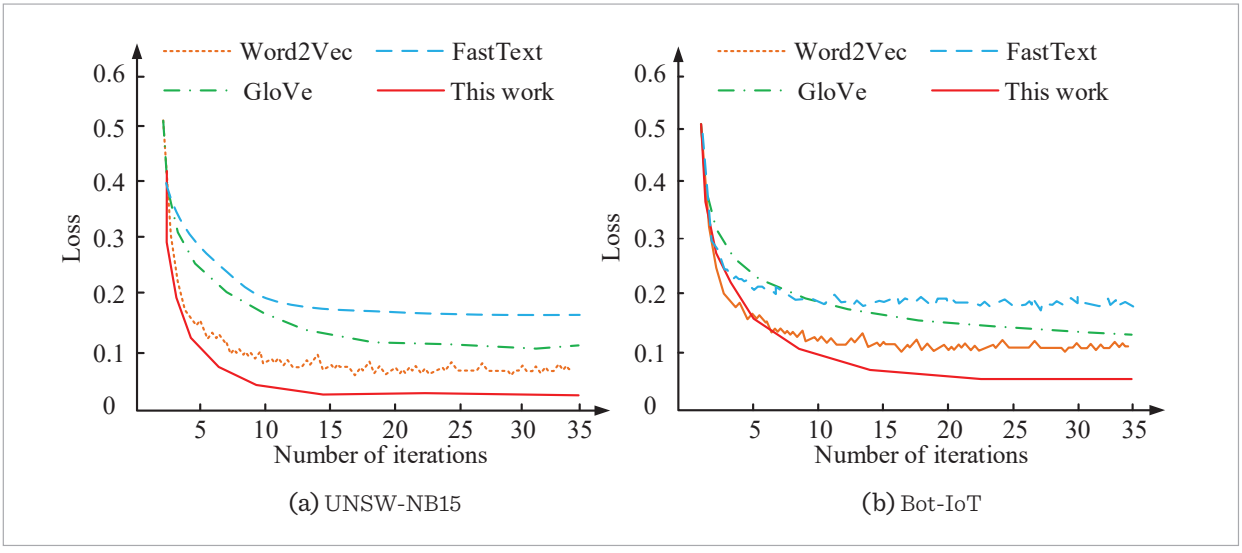


Figure 8
Accuracy test results.

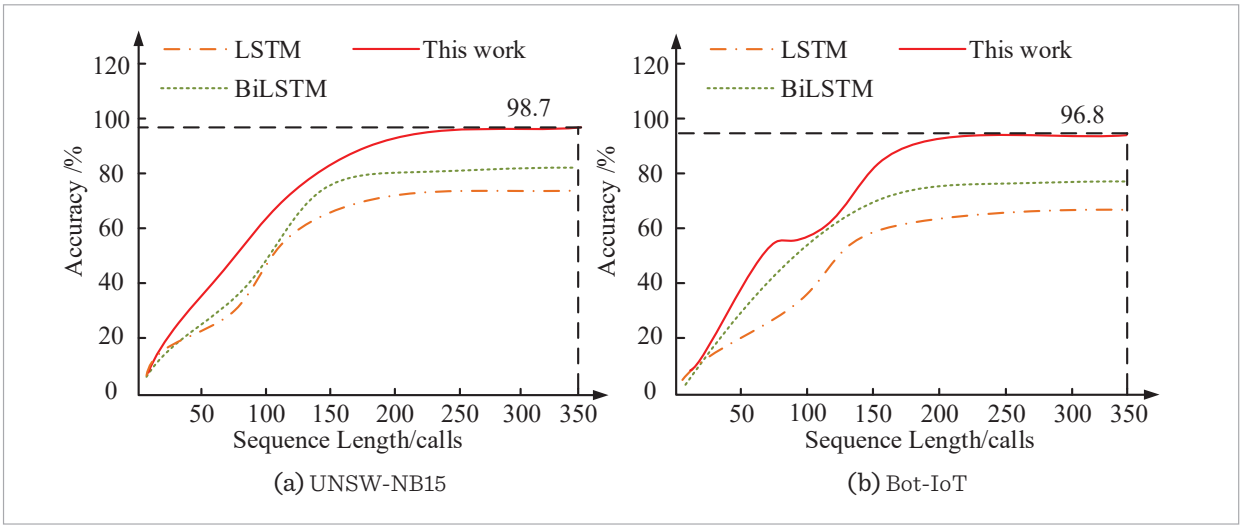
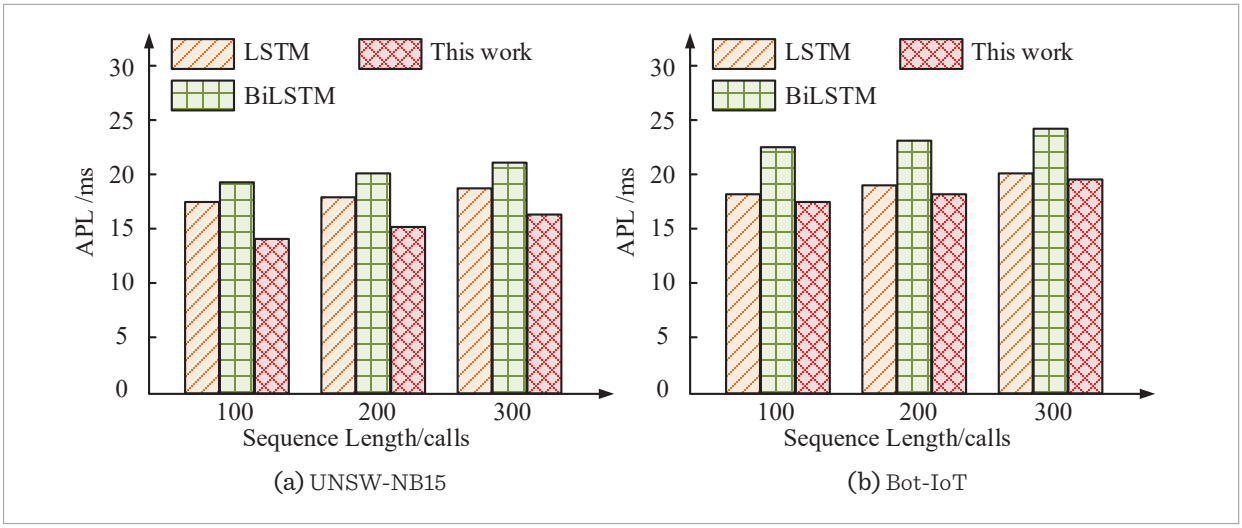


Figure 9
APL test results.



ing the accuracy rate to 80.2%. However, due to its reliance only on the overall features of the sequence without specifically modeling the temporal features, BiLSTM still has certain limitations when capturing the temporal dependencies in the API call sequence. The TPA-BiLSTM model based on time features proposed by the research institute effectively captures fine-grained time series patterns in the sequence, and the accuracy rate is further improved to 98.7%. In the Bot-IoT shown in Figure 8(b), due to severe data imbalance, the accuracy of LSTM and BiLSTM is only 62.8% and 77.6%. TPA-BiLSTM maintains an accuracy of around 96.8% in multiple experiments, with improvements of 34.0% and 19.2% compared to LSTM and BiLSTM. TPA and Skip-Gram models can effectively enhance the modeling ability of long sequence dependencies, thereby improving classification performance under extremely imbalanced data conditions. Finally, to validate the response speed in continuous API call sequence detection, this study also tests model using Average Prediction Latency (APL) as a reference metric, as shown in Figure 9.

In Figure 9(a), in the UNSW-NB15, the APL of LSTM is 18.4 ms, BiLSTM is 20.7 ms, and TPA-BiLSTM is 16.8 ms. In Figure 9(b), in the Bot-IoT, due to data imbalance, the APL of LSTM and BiLSTM are 20.2 ms and 24.5 ms, while the research model is 19.8 ms. This indicates that with an increase in sequence length, TPA-BiLSTM maintains high classification performance while maintaining reasonable delay control.

3.2. Simulation Performance Testing of IoT Security Detection Methods

After verifying the ability of TPA-BiLSTM to capture temporal features, further simulation performance tests are conducted to comprehensively evaluate the DenseNet 1D-CNN method. The testing focuses on verifying the method's ability to capture both temporal and spatial features simultaneously, as well as its detection performance and stability under imbalanced data conditions. The experiment continues to use UNSW-NB15 and Bot-IoT as the testing basis. This study first conducts ablation testing on the proposed method, as shown in Figure 10.

In Figure 10(a), the accuracy of TPA-BiLSTM, DenseNet 1D-CNN, and the final model in UNSW-NB15 is 96.3%, 97.1%, and 99.3%. In Figure 10(b), in the Bot-IoT, due to severe data imbalance issues, the detection accuracy of using only TPA-BiLSTM or DenseNet 1D-CNN is 87.3% and 89.1%, while the fusion model's accuracy combining the two is improved to 95.6%. This shows that the fusion model can effectively alleviate the performance degradation of a single model under extremely unbalanced data. The synergistic effect of horizontal and vertical 1D-CNN plays a key role in feature extraction. Horizontal 1D convolution mainly captures local temporal patterns and the dependencies between variables. Vertical 1D convolution integrates the entire sequence information to achieve global feature fusion, thereby enhancing the model's ability to recognize abnormal behavior. In addition, this study also introduces popular IoT security detection methods for comparison. For example, Floating Centroids Method (FCM), Expectation-Maximum Hidden Markov Model (EMHMM), and DL-based Dynamic Adaptive Network Security Policy System (DL-DANSPS). The experiment tests four common network attacks: DOS, User to Root (U2R), Remote to Local (R2L), and Cross-Site scripting (XSS). At this point, the confusion matrix for network attack detection classification of the four models is shown in Figure 11.

Figures 11(a)-(d) show a comparison of four matrices: FCM, EMHMM, DL-DANSPS, and the research model. Among the four types of attack detection tasks, FCM has a relatively accurate classification of high-frequency attacks (DoS) (84%), but there are high misclassifications in U2R and R2L attacks, resulting in insufficient overall discrimination abil-

Figure 10
Ablation test results.

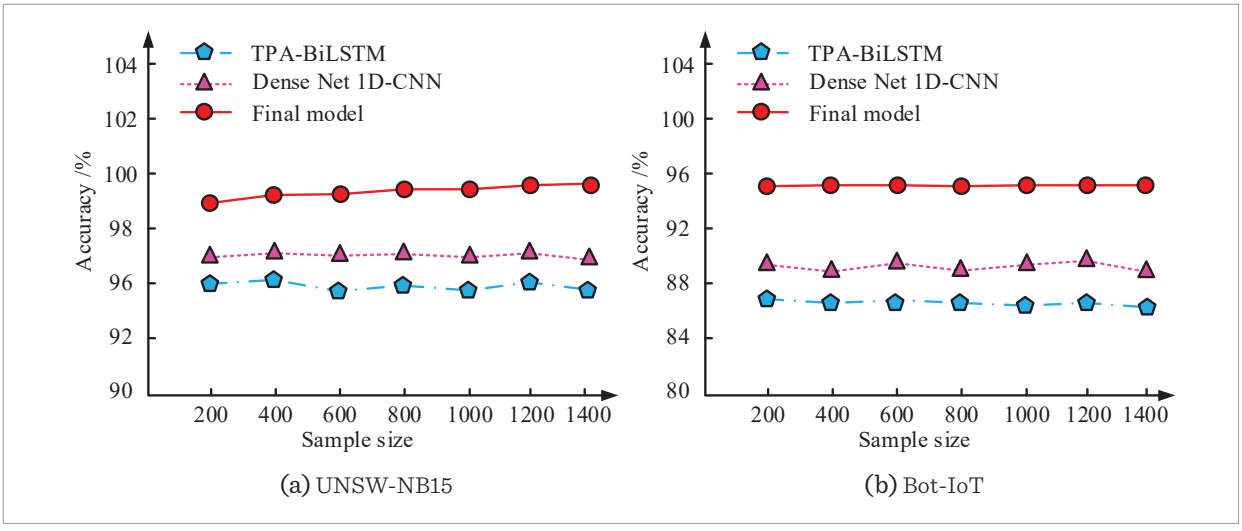
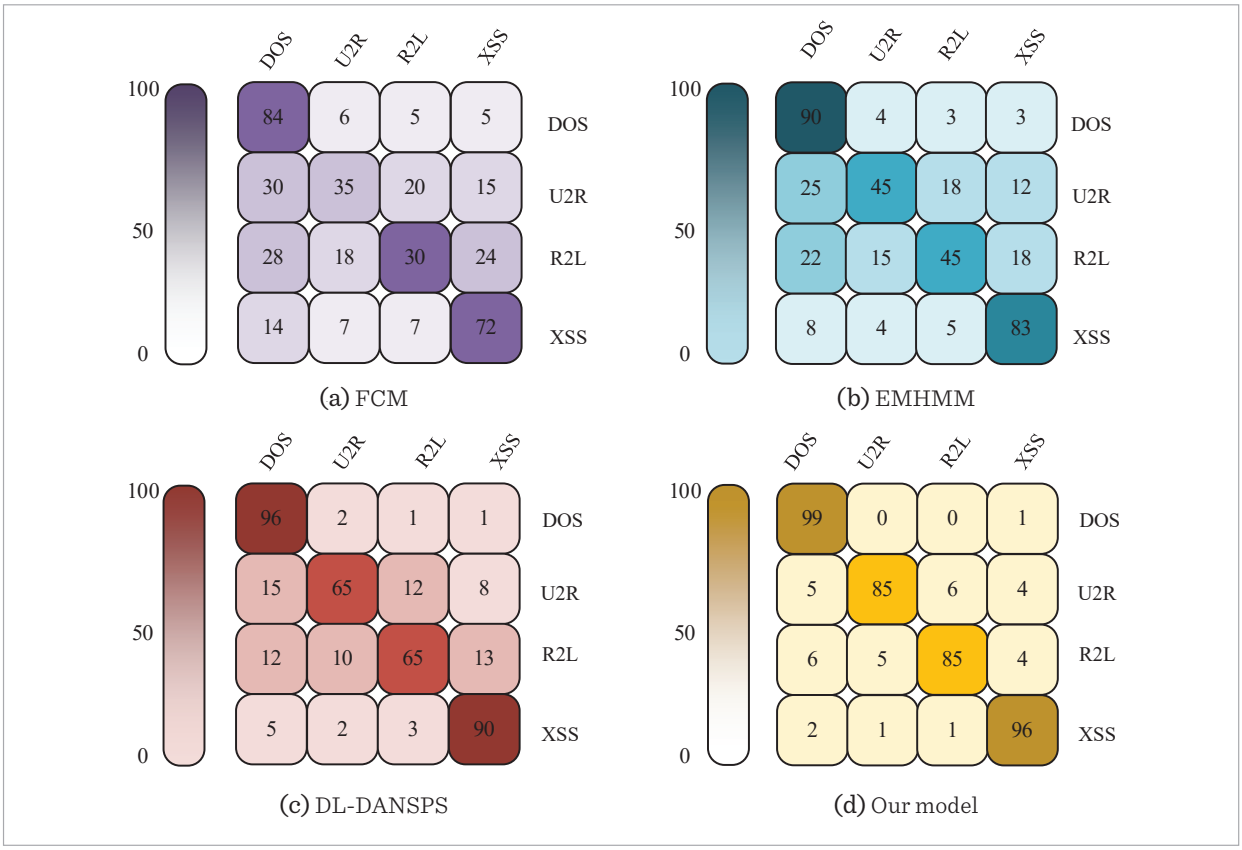


Figure 11
Internet attack confusion matrix results for various models.



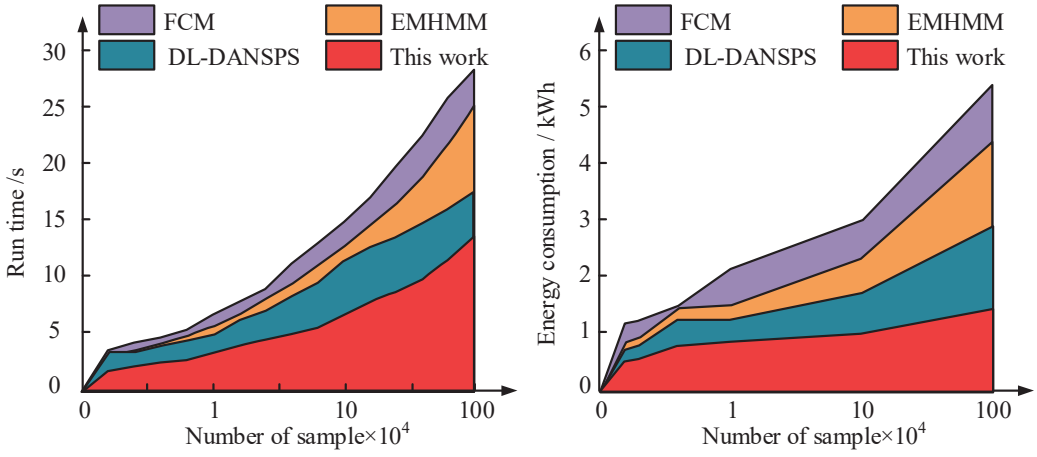
ity. The classification of DoS and XSS attacks in EMHMM has improved (90% and 83%), but there is still a lot of confusion between U2R and R2L, and their sensitivity to low-frequency attacks is poor. The overall performance of DL-DANSPS in four types of attacks is better than the first two, with a significant reduction in misclassification for DoS and XSS (96% for DoS and 90% for XSS), but there is still a certain error rate for U2R and R2L. The classification results of DenseNet 1D-CNN method are clearer on all types of attacks (DoS 99%, U2R 85%, R2L 85%, XSS 96%), and the diagonal elements of the confusion matrix are significantly higher than other models. On low-frequency attacks such as U2R and R2L, the number of misclassifications is significantly reduced, indicating that this method can maintain stable detection performance in both high-frequency and low-frequency attack scenarios, and has strong generalization and robustness. To more accurately quantify the performance of each method, this study continues to test with precision, recall, and F1 value as reference indicators (Table 4).

In Table 4, FCM performs the worst in multi criteria testing, followed by EMHMM, DL-DANSPS, and DenseNet 1D-CNN. In the UNSW-NB15, the precision, recall, and F1 value of FCM are 82.4%, 74.6%, and 78.3%, while the research model achieves 97.6%, 96.8%, and 97.2%. Compared with FCM, the improvement in research methods is 15.2%, 22.2%, and 18.9%. In the Bot-IoT, the precision, recall, and F1 value of FCM are 75.1%, 68.3%, and 71.5%, while the research model achieves 95.7%, 94.1%, and 94.9%, with improvement rates of 20.6%, 25.8%, and 23.4%. The DenseNet 1D-CNN method has relatively good performance and is more suitable for the current stage of network intrusion detection work. Finally, this study also compares the running time and energy consumption of the above four methods under different feature quantities, as shown in Figure 12.

Table 4
Comparison test results of multiple indicators.

Data set	Models	Precision/%	Recall/%	F1 value/%
UNSW-NB15	FCM	82.4	74.6	78.3
	EMHMM	87.2	82.5	84.8
	DL-DANSPS	92.5	89.7	91.0
	This work	97.6	96.8	97.2
Bot-IoT	FCM	75.1	68.3	71.5
	EMHMM	81.3	76.4	78.8
	DL-DANSPS	88.6	84.2	86.3
	This work	95.7	94.1	94.9

Figure 12
Performance comparison of different methods.



(a) Run time (b) Energy consumption

In Figure 12(a), when the sample size is small (1×10^4), the running times of the four methods are all less than 10.0 s. The running times of FCM, EMHMM, DL-DANSPS, and research methods are 6.8 s, 4.6 s, 3.9 s, and 2.1 s. As the sample size increases to 1×10^6, the running times of the four methods show an approximately linear increase, reaching 27.3 s, 24.7 s, 16.2 s, and 13.2 s, indicating that the proposed method still maintains significant advantages in large-scale data processing. In Figure 12(b), hardware configuration has a certain impact on the results in terms of energy consumption, but the overall trend is similar to the running time. When the sample size is 1×10^4, the energy consumption values of the FCM, EMHMM, DL-DANSPS, and research methods are all less than 3.0 kWh, which are 2.1 kWh, 1.3 kWh, 1.1 kWh, and 0.4 kWh, respectively. When the sample size increases to 1×10^6, the energy consumption of the four methods reaches 5.5 kWh, 4.1 kWh, 2.5 kWh, and 0.9 kWh. The energy cost of the single model FCM and EMHMM is significantly higher, while the research model effectively reduces feature redundancy and overall energy consumption cost under the feature channel compression mechanism. This indicates that the method balances computational efficiency and energy utilization while ensuring detection accuracy.

4. Summary

The scalability, resource limitations, and number of connected devices of IoT made its related devices and systems more vulnerable to network attacks. Based on this, this study first proposed a detection model that integrates temporal and spatial features to address security detection issues in IoT environments. By introducing the TPA-BiLSTM structure, the model could fully capture the time-dependent characteristics of API call sequences when executing MCode. Meanwhile, utilizing DenseNet 1D-CNN for efficient feature reuse and cross-layer transfer further enhanced the learning ability of complex spatial patterns. In the experiment, compared with word embedding models, like Word2Vec, GloVe, and FastText, Skip-Gram performed the best in extracting semantic information and contextual dependencies from IoT malicious behavior data, providing higher quality input features for subsequent TPA-BiLSTM time feature modeling. The convergence loss value ultimately stabilized at 0.028. In the UNSW-NB15 dataset, the APL value of TPA-BiLSTM was 16.8 ms, while in Bot-IoT it was 19.8 s. This indicated that with an increase in sequence length, the research model maintained high classification performance while maintaining reasonable delay control. The precision, recall, and F1 values of the final model on UNSW-NB15 were 97.6%, 96.8%, and 97.2%. In addition, the final model had significantly less running time and energy overhead compared to other methods. When the sample size increased to 1×10^6, the running time and energy consumption were 13.2 s and 0.9 kWh. The research model effectively reduced overall energy consumption while ensuring high detection accuracy, reflecting its practical application value in resource constrained environments. However, there is still room for improvement in the generalization ability of the research model when facing extremely sparse abnormal sequences or new unknown attack samples. Future work can further combine incremental learning and adaptive feature selection mechanisms to enhance the real-time detection capability in dynamic IoT environments.

Funding

The research is supported by Science and Technology Department of Shanxi Province' Technology Strategy Project, Research on the Strategy of Network Security Innovation Talent Alliance Based on Shanxi, No.: 202204031401130.

Conflicts of Interest

The author declares that there is no conflict of interest.

References

1. Adigun, E. B., Ismaila, W. O., Baale, A. A. Optimized DenseNet Architecture for Efficient Classification of Encrypted Internet Traffic. Asian Journal of Research in Computer Science, 2025, 18(2), 197-205. https://doi.org/10.9734/ajrcos/2025/v18i2571
2. Al-Eryani, A. M., Omara, F. A., Hossny, E. A Deep Learning GRU-BiLSTM for DDoS Attack Detection. SN Computer Science, 2025, 6(6), 1-17. https://doi.org/10.1007/s42979-025-04110-1
3. Al-Kahtani, M. S., Mehmood, Z., Sadad, T. Intrusion Detection in the Internet of Things Using Fusion of GRU-LSTM Deep Learning Model. Intelligent Automation and Soft Computing, 2023, 37(2), 2279-2290. https://doi.org/10.32604/iasc.2023.037673
4. Alqahtani, A. Optimized Deep Autoencoder and BiLSTM for Intrusion Detection in IoTs-Fog Computing. Multimedia Tools and Applications, 2025, 84(8), 4907-4943. https://doi.org/10.1007/s11042-024-18919-0
5. Alzubi, O. A., Qiqieh, I., Alzubi, J. A. Fusion of Deep Learning Based Cyberattack Detection and Classification Model for Intelligent Systems. Cluster Computing, 2023, 26(2), 1363-1374. https://doi.org/10.1007/s10586-022-03686-0
6. Anandhi, V., Vinod, P., Menon, V. G. Malware Visualization and Detection Using DenseNets. Personal and Ubiquitous Computing, 2024, 28(1), 153-169. https://doi.org/10.1007/s00779-021-01581-w
7. Ayas, S., Ayas, M. S. A Modified DenseNet Approach with NearMiss for Anomaly Detection in Industrial Control Systems. Multimedia Tools and Applications, 2022, 81(16), 22573-22586. https://doi.org/10.1007/s11042-021-11618-0

8. Bi, S., Wang, J., Song, J. Research on the Intrusion Detection Model for Power Internet of Things Combining Deep Belief Network and BiLSTM. Journal of Cyber Security and Mobility, 2025, 14(3), 653-672. https://doi.org/10.13052/jcsm2245-1439.1436
9. Dandapat, A., Mondal, B. Design of Intrusion Detection System Using GA and CNN for MQTT-Based IoT Networks. Wireless Personal Communications, 2024, 134(4), 2059-2082. https://doi.org/10.1007/s11277-024-10984-w
10. Eid, A. M., Soudan, B., Nassif, A. B. Enhancing Intrusion Detection in IoT: Optimized CNN Model with Multi-Class SMOTE Balancing. Neural Computing and Applications, 2024, 36(24), 14643-14659. https://doi.org/10.1007/s00521-024-09857-x
11. Groumpos, P. P. A Critical Historic Overview of Artificial Intelligence: Issues, Challenges, Opportunities, and Threats. Artificial Intelligence Applications, 2023, 1(4), 197-213. https://doi.org/10.47852/bonviewAIA3202689
12. Hanafi, A. V., Ghaffari, A., Rezaei, H. Intrusion Detection in Internet of Things Using Improved Binary Golden Jackal Optimization Algorithm and LSTM. Cluster Computing, 2024, 27(3), 2673-2690. https://doi.org/10.1007/s10586-023-04102-x
13. Hintaw, A. J., Manickam, S., Aboalmaaly, M. F. MQTT Vulnerabilities, Attack Vectors and Solutions in the Internet of Things (IoT). IETE Journal of Research, 2023, 69(6), 3368-3397. https://doi.org/10.1080/03772063.2021.1912651
14. Kanimozhi, V., Jacob, T. P. The Top Ten Artificial Intelligence-Deep Neural Networks for IoT Intrusion Detection System. Wireless Personal Communications, 2023, 129(2), 1451-1470. https://doi.org/10.1007/s11277-023-10198-6
15. Kumar, A. A., Karne, R. K. IoT-IDS Network Using Inception CNN Model. Journal of Trends in Computer Science and Smart Technology, 2022, 4(3), 126-138. https://doi.org/10.36548/jtcsst.2022.3.002
16. Li, S., Wang, J., Song, Y. Tri-Channel Visualised Malicious Code Classification Based on Improved ResNet. Applied Intelligence, 2024, 54(23), 12453-12475. https://doi.org/10.1007/s10489-024-05843-x
17. Lu, K., Huang, J., Zeng, G. Multi-Objective Discrete Extremal Optimization of Variable-Length Blocks-Based CNN by Joint NAS and HPO for Intrusion Detection in IIoT. IEEE Transactions on Dependable and Secure Computing, 2025, 22(4), 4266-4283. https://doi.org/10.1109/TDSC.2025.3545363
18. Lu, Z., Qin, S., Lv, P., Sun, L., Tang, B. Real-Time Continuous Detection and Recognition of Dynamic Hand Gestures in Untrimmed Sequences Based on End-to-End Architecture with 3D DenseNet and LSTM. Multimedia Tools and Applications, 2024, 83(6), 16275-16312. https://doi.org/10.1007/s11042-023-16130-1
19. Maniveena, C., Kalaiselvi, R. A Security and Privacy Preserving Approach Based on Social IoT and Classification Using DenseNet Convolutional Neural Network. Automatika, 2024, 65(1), 333-342. https://doi.org/10.1080/00051144.2023.2296788
20. Manoharan, P., Hong, W., Yin, J. Optimising Insider Threat Prediction: Exploring BiLSTM Networks and Sequential Features. Data Science and Engineering, 2024, 9(4), 393-408. https://doi.org/10.1007/s41019-024-00260-z
21. Morshedi, R., Matinkhah, S. M., Sadeghi, M. T. Intrusion Detection for IoT Network Security with Deep Learning. Journal of AI and Data Mining, 2024, 12(1), 37-55. https://doi.org/10.21203/rs.3.rs-2648993/v1
22. Om Kumar, C. U., Marappan, S., Murugeshan, B. Intrusion Detection Model for IoT Using Recurrent Kernel Convolutional Neural Network. Wireless Personal Communications, 2023, 129(2), 783-812. https://doi.org/10.1007/s11277-022-10155-9
23. Pandey, A., Lal, N., Chakravert, A. K. Hybrid DenseNet201 with CBPN-Based Image Pixel Enhancement and Optimized ADBGAN for Image Copy-Move Forgery Detection. Signal, Image and Video Processing, 2025, 19(6), 1-13. https://doi.org/10.1007/s11760-025-04502-z
24. Saleh, I. F. Enhanced Malware Detection for IoT Networks Utilizing a 2D-CNN with Data Augmentation on the Malimg Dataset. Journal of Al-Qadisiyah for Computer Science and Mathematics, 2025, 17(1), 179-191. https://doi.org/10.29304/jqcsm.2025.17.11973
25. Saravanan, V., Madiajagan, M., Rafee, S. M. IoT-Based Blockchain Intrusion Detection Using Optimized Recurrent Neural Network. Multimedia Tools and Applications, 2024, 83(11), 31505-31526. https://doi.org/10.1007/s11042-023-16662-6
26. Sarker, I. H., Khan, A. I., Abushark, Y. B., Alsolami, F. Internet of Things (IoT) Security Intelligence: A Comprehensive Overview, Machine Learning Solutions and Research Directions. Mobile Networks and Appli-

cations, 2023, 28(1), 296-312. https://doi.org/10.1007/s11036-022-01937-3

27. Sharma, O., Sharma, A., Kalia, A. Windows and IoT Malware Visualization and Classification with Deep CNN and Xception CNN Using Markov Images. Journal of Intelligent Information Systems, 2023, 60(2), 349-375. https://doi.org/10.1007/s10844-022-00734-4
28. Torre, D., Chennamaneni, A., Jo, J. Toward Enhancing Privacy Preservation of a Federated Learning CNN Intrusion Detection System in IoT: Method and Empirical Study. ACM Transactions on Software Engineering and Methodology, 2025, 34(2), 1-48. https://doi.org/10.1145/3695998
29. Wang, Z., Wang, W., Yang, Y. CNN- and GAN-Based Classification of Malicious Code Families: A Code Visualization Approach. International Journal of Intelligent Systems, 2022, 37(12), 12472-12489. https://doi.org/10.1002/int.23094
30. Yan, F., Zhang, G., Zhang, D., Sun, X., Hou, B., Yu, N. TL-CNN-IDS: Transfer Learning-Based Intrusion Detection System Using Convolutional Neural Network. Journal of Supercomputing, 2023, 79(15), 17562-17584. https://doi.org/10.1007/s11227-023-05347-4
31. Zeng, G. Q., Yang, Y. W., Lu, K. D. Evolutionary Adversarial Autoencoder for Unsupervised Anomaly Detection of Industrial Internet of Things. IEEE Transactions on Reliability, 2025, 74(1), 3454-3468. https://doi.org/10.1109/TR.2025.3528256
32. Zhong, N., Qian, Z., Zhang, X. Sparse Backdoor Attack Against Neural Networks. The Computer Journal, 2024, 67(5), 1783-1793. https://doi.org/10.1093/comjnl/bxad100