Frontiers in Computing and Intelligent Systems
ISSN: 2832-6024 | Vol. 4, No. 2, 2023

IoT Intrusion Detection Model based on CNN-GRU

Zhaolian Wang, Hong Huang *, Rui Du, Xing Li, Guotao Yuan

School of Materials Science and Technology, University of Science and Engineering, Yibin 644000, China
*** Corresponding author:** Hong Huang (Email: huanghong@suse.edu.cn)

Abstract: With the rapid development of IoT technology, security concerns surrounding IoT devices have gained attention. An intrusion detection system for IoT can quickly and accurately identify highly redundant data features in IoT traffic categories. To reduce data, feature redundancy during the identification process, this study proposes the use of Extreme Gradient Boosting (XGBoost) for feature selection to obtain an optimal feature subset. Additionally, to improve the accuracy of identifying malicious traffic in IoT devices, a fusion model combining Convolutional Neural Networks (CNN) and Gated Recurrent Units (GRU) for IoT intrusion detection is proposed. Finally, a comparative analysis experiment is conducted between CNN-GRU and CNN-LSTM, demonstrating that the proposed model achieves lower processing time while ensuring accuracy. Furthermore, the proposed method outperforms classical IoT intrusion detection algorithms in terms of precision and recall.

Keywords: IoT Intrusion Detection; CNN; GRU.

1. Introduction

In recent years, with the development of IoT technology, the number and scope of IoT devices have been continuously expanding, becoming an important driving force for digital transformation and intelligent development. At the same time, ensuring the security of IoT devices and applications in small and large networks composed of physical and virtual infrastructure has become crucial. Current IoT devices are susceptible to network attacks such as denial of service [1], botnets [2], and penetration attacks [3]. The Mirai botnet attack is a distributed denial-of-service (DDoS) attack on IoT devices [4]. Mirai can incorporate connected devices extracted from smart homes into a "botnet" [2], leading to the shutdown of Dyn's infrastructure, a prominent US domain name service provider, affecting websites such as Twitter, PayPal, and GitHub.

In terms of feature redundancy, IoT intrusion detection techniques aim to reduce the number of features to not only shorten the model training time but also improve detection effectiveness. Liu et al. [11] proposed an intrusion detection method based on Principal Component Analysis (PCA) and Recurrent Neural Networks (RNN). PCA was used to reduce data dimensionality and noise and identify the most informative subset of principal features. The processed data was then trained using an RNN network with high classification accuracy. Zhou et al. [12] proposed a feature selection method using Gradient Boosting Trees. XGBoost is a nonlinear method of gradient boosting that not only calculates feature importance for feature selection but also prevents overfitting by adding an L2 regularization term to the objective function, effectively addressing the problem of feature redundancy.

In the exploration of IoT intrusion detection models, several methods and techniques have been proposed by researchers. Gurusamy [5] proposed a secure IoT framework using Software-Defined Networking (SDN) in IoT traffic. SVM algorithms were used on the SDN controller to monitor and learn IoT device behavior and detect attacks, achieving high precision, recall, and fast response time. Hodo et al. [6] applied Artificial Neural Network (ANN) algorithms to detect DDoS/DoS attacks based on the characteristics of host-based IDS and network-based IDS, demonstrating good accuracy in traffic classification. Debastrita et al. [7] proposed a set learning method for anomaly detection, utilizing Stacked Autoencoders (SAE) to extract deep features and inputting them into a probabilistic neural network ensemble for single and multiple anomaly detection. The use of autoencoders resulted in good and stable performance. Wang Yun [8] constructed a home IoT intrusion detection model based on Recurrent Neural Networks (RNN) using the NSL-KDD dataset, reducing false positives and improving accuracy. Alkahtani et al. [9] achieved certain results in detecting IoT device botnet attacks using Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) algorithms, showing promising accuracy. Li Xiaojia et al. [10] utilized CNN and Recurrent Neural Networks (RNN) for IoT intrusion detection using the ToN-IoT dataset, achieving good precision and F1 score. 1DCNN has the ability to capture local parallel features in IoT device traffic quickly, while RNN, specifically GRU, can extract long-term dependency learning features. GRU has a simpler internal structure and fewer parameters compared to LSTM, making it suitable for accurately and efficiently identifying traffic within a short period of time.

This paper proposes a hybrid model combining Convolutional Neural Networks (CNN) and Gated Recurrent Units (GRU) for the detection of malicious attacks on selected IoT devices. The N-BaIoT dataset is used as the base dataset. Firstly, the dataset is normalized and one-hot encoded. XGBoost is then used for feature selection to reduce data redundancy. Finally, a one-dimensional CNN is used to extract spatial features, and GRU is employed to extract long-term dependency information features and learn the interdependencies between features. The softmax function is used for classification. Comparative analysis experiments demonstrate the applicability of the proposed model. The results show that this method outperforms recent IoT intrusion detection methods in terms of accuracy, recall, and loss based on the aforementioned dataset, effectively improving the accuracy of IoT intrusion detection.

2. Related Work

2.1. XGBoost

XGBoost is a boosting-based ensemble method that utilizes decision trees. The algorithm follows the principle of optimizing the empirical loss function by iteratively fitting the negative gradient of the loss function. It then generates the optimal weak learner using a linear search method. The objective function for the t-th tree is as follows:

$$\mathrm{Obj}^{(t)} = \sum_{i=1}^{n} l(y_i^t, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_t) \tag{1}$$

Where $l(y_i^t, \hat{y}_i)$ is the loss function used to evaluate model error and $\Omega(f_t)$ is the regularization term used to control the complexity of the tree and prevent overfitting.

To find the minimum of the objective function, the objective function is expanded using the second-order Taylor series approximation, and the optimal weights ω and the corresponding objective function value can be obtained:

$$\omega_j^* = -\frac{G_j}{H_j + \lambda} \tag{2}$$

$$X_{obj} = -\frac{1}{2} \sum_{j=1}^{T} \frac{G_j}{H_j + \lambda} + \lambda T \tag{3}$$

Where $G_j = \sum_{i \in I_j} g_i$ represents the sum of the first-order gradients of the loss function for all samples in the j-th node, and $H_j = \sum_{i \in I_j} h_j$ represents the sum of the second-order gradients of the loss function for all samples in the j-th node.

2.2. CNN

In the proposed model architecture, we have introduced a one-dimensional Convolutional Neural Network (1DCNN) component for the identification of malicious IoT traffic. 1DCNN is capable of extracting morphological features from traffic data to enhance the understanding of the traffic. A typical 1DCNN architecture consists of convolutional layers, pooling layers, and fully connected layers.

(1) Convolutional Layers: The convolutional layers in 1DCNN overcome the slow convergence of conventional neural networks. This layer consists of a set of time series graphs, kernels, filters, strides, and neurons. Each neuron in this layer is connected to adjacent neurons. The convolution operation computes the dot product between the corresponding convolution filter and the input time series graph. Through the kernels, it learns the features of the input mapping. The time series input is first fed into the input layer. Then, the output of the convolution process is computed as follows:

$$\bar{y}_m^l = b_m^l + \sum_{i \in L_m} \mathrm{conv1D}\left(\bar{\omega}_{im}^{l-1}, \hat{y}_i^{l-1}\right) \tag{4}$$

Where $\bar{y}_m^l$ and b_m^l represent the output and bias of the m^{th} neuron in the l^{th} layer, respectively. $\bar{w}_{im}^{l-1}$ and $\hat{y}_i^{l-1}$ represent the kernel weights from the i^{th} neuron in the (l-1)th layer to the m-th neuron in the lth layer and the input to the mth neuron in the lth layer from the ith neuron in the $(l-1)^{th}$ layer, respectively. L_m is the activation function applied to the input mapping.

(2) Pooling Layers: The pooling operation in a 1DCNN model is a resampling process that converts multiple elements into a single element. This operation minimizes computational costs, retains important information, and helps prevent overfitting. In this study, we utilize the max pooling model, which selects the maximum value in an array as the output. The formula for reducing the size of time series graphs while maintaining discriminative information is as follows:

$$p = \max\left(\bar{y}R\right) \tag{5}$$

Where p represents the output of the max pooling layer, and $\bar{y}R$ represents the corresponding set of elements within the pooling region.

(3) Fully Connected Layers: Following the pooling layer, the input is flattened into a one-dimensional array and connected to a layer where each input is connected to the output with weights. The structure of this layer is similar to a conventional artificial neural network, and the neurons in this layer can be computed as follows:

$$u_k = \sum_{i=1}^{m} w_{i,k} p_i + b_k \tag{6}$$

Where u_k represents the output value of the kth neuron, $w_{i,k}$ is the weight value of the ith input for the kth neuron, p_i is the ith input value, and b_k is the bias value of the kth neuron. After obtaining the output from the fully connected layer, an activation function is applied. In this model, the chosen activation function is SoftMax, which is used for multi-class classification.

2.3. GRU

GRU (Gated Recurrent Unit) is a type of recurrent neural network that is simpler compared to the LSTM structure because it does not have a memory cell. GRU offers a lower computational cost while delivering performance comparable to LSTM. The GRU structure consists of two units, namely the update gate and the reset gate. The update gate determines the past information that needs to be retained or forgotten in the time series to meet the current prediction requirements. The reset gate, on the other hand, decides the amount of information that needs to be memorized. The illustration in Figure 1 demonstrates this process, as the Figure 1.

In GRU, the current input $\bar{x}_t$ and the previous output h_{t-1} are first concatenated as the input to the update gate and the reset gate. Additionally, each value in every unit or component of the GRU can be obtained as follows:

$$z_t = \sigma\left(W_z \bullet \left[h_{t-1}, \bar{x}_t\right] + b_z\right) \tag{7}$$

$$\bar{r}_t = \sigma\left(W_{\bar{r}} \bullet \left[h_{t-1}, \bar{x}_t\right] + b_{\bar{r}}\right) \tag{8}$$

$$\bar{h}_t = \tanh\left(W_{\bar{h}} \bullet \left[\bar{\Upsilon}_t \bullet h_{t-1}, \bar{x}_t\right] + b_{\bar{h}}\right) \tag{9}$$

Where z_t, $\bar{r}_t$, and $\bar{h}_t$ represent the outputs of the update gate, reset gate, and candidate hidden state, respectively. σ and $\tanh$ represent the activation functions sigmoid and hyperbolic tangent function,

respectively. W_z, $W_{\bar{r}}$, $W_{\bar{h}}$ and b_z, $b_{\bar{r}}$, $b_{\bar{h}}$, and b are the weight matrices and bias vectors associated with the update gate, reset gate, and candidate hidden state.

The output of the t^{th} GRU can be obtained as follows:

$$h_t = (1 - z_t) \times h_{t-1} + z_t \times \bar{h}_t \tag{10}$$

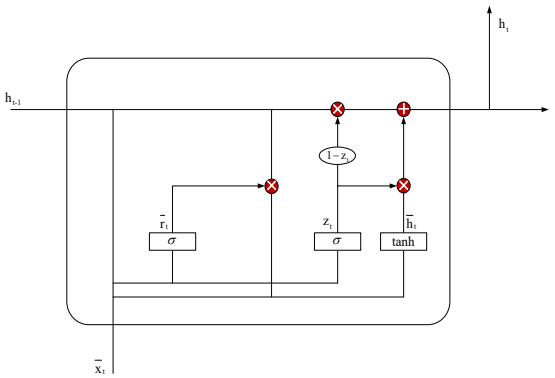

Figure 1. GRU Architecture

3. Model Design

To improve the detection performance of IoT intrusion detection, a deep learning approach based on XGBoost feature selection and CNN-GRU is proposed. The model consists of three modules: data preprocessing module, feature selection module, and classifier module. The design process of the model is as follows:

Input: Relevant traffic data collected from IoT devices. The publicly available N-BaIoT dataset, specifically the data related to baby monitor devices, is used in this experiment.

Output: Attack categories targeting IoT devices.

Step 1: Perform missing value handling on the initial input dataset.

Step 2: Normalize the dataset N, convert non-numeric features to numerical values, and encode the labels to obtain the preprocessed dataset *N*.

Step 3: Split the dataset into training set N_1 and test set N_2. Perform feature selection using XGBoost feature importance scoring to obtain the optimal feature subset *F*.

Step 4: Utilize the selected optimal feature set F to obtain the training set Q_1 and test set Q_2, removing redundant features.

Step 5: Train the CNN-GRU model using the Adam optimizer to optimize the constructed CNN-GRU classifier.

Step 6: Evaluate the proposed CNN-GRU classifier model using metrics such as accuracy, precision, recall, and F1-score.

3.1. Data Preprocessing Module

Non-numeric features are encoded using one-hot encoding to convert them into numerical values. Then, the feature data is standardized to eliminate the scale differences between features. The standardization process is shown in equations (13)-(15):

$$t'_{ij} = \frac{t_{ij} - AVG_j}{\sigma_j} \tag{11}$$

$$AVG_j = \frac{1}{n}(t_{1j} + t_{2j} + \cdots + t_{nj}) \tag{12}$$

$$\sigma_j = \sqrt{\frac{1}{n}\left[(t_{1j} - AVG_j)^2 + (t_{2j} - AVG_j)^2 + \cdots + (t_{nj} - AVG_j)^2\right]} \tag{13}$$

Here, t_{ij} represents the feature value, t_{ij}' represents the standardized feature value, AVG_j represents the average value of the feature, and σ_j represents the standard deviation of the feature.

3.2. Feature Selection Module

In this study, the XGBoost model is used as a feature selector. Information gain is employed to filter the features, and its formula is as follows:

$$Gain = \frac{1}{2}\left(\frac{G_L^2}{\lambda + H_L} + \frac{G_R^2}{\lambda + H_R} - \frac{(G_L + G_R)^2}{H_L + H_r + \lambda}\right) - \gamma \tag{14}$$

Where G_L represents the sum of the first-order gradients of the loss function for the left leaf node, H_L represents the sum of the second-order gradients of the loss function for the left leaf node, λ controls the score of the leaf node, and γ controls the number of leaf nodes.

By using information gain, the model performs feature selection. A higher information gain indicates a larger decrease in loss, indicating a better split at the current node. By calculating the information gain for each feature, the model identifies the optimal split point that maximizes the information gain. This process iteratively selects features, retaining those with the highest discriminative power while removing unnecessary features, aiming to improve classification performance and reduce computational complexity.

In this study, XGBoost feature selection is employed to select the top 30 features, addressing the challenge of complex and diverse malicious traffic features in IoT.

3.3. Fused Classifier Module

(1) CNN-GRU Hybrid Deep Model: The basic structure includes an Input Layer, a Conv1D Layer, a MaxPool1D Layer, Batch Normalization (BN) Layer, a Gated Recurrent Unit (GRU) Layer, a Flatten Layer, and a Dense Layer.

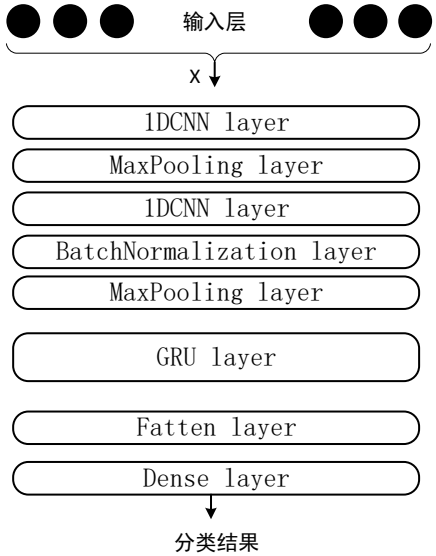

Figure 2. CNN-GRU Model

Firstly, each selected traffic feature is transformed into a (30,1) tensor and input to the Input Layer. Two layers of 1D convolutions are applied to extract features from IoT traffic, with multiple channels set to capture multi-dimensional features. The trained features are then passed to the GRU layer, which uses the tanh activation function with 64 neurons.

Finally, the Flatten Layer flattens the 2D feature maps, and the features are fed into the Dense Layer for classification.

The commonly used activation functions for classification problems are Sigmoid and Softmax. In this model, only the Softmax activation function is used, with the number of neurons in this layer set to 10.

To improve the accuracy of the model, various parameters such as learning rate and weights need to be adjusted. Techniques such as L1 regularization, L2 regularization, and Batch Normalization can be used to prevent overfitting. Deep neural networks have a large number of parameters, making the model complex and prone to overfitting. The use of Batch Normalization helps to mitigate the occurrence of internal covariate shift. By integrating the convolutional features with the Batch Normalization layer, overfitting of the data is avoided.

4. Experimental Results

4.1. Dataset and Data Preprocessing

The experiment utilizes the N-BaIoT dataset created by Meidan et al. [14]. The dataset captures traffic by monitoring devices during Bashlite and Mirai botnet attacks, with traffic packets captured using Wireshark. A total of 115 features were extracted from each device, collected considering five-time intervals: 1 minute, 10 seconds, 1.5 seconds, 500 milliseconds, and 100 milliseconds (referred to as L0.01, L0.1, L1, L3, and L5, respectively). For example, the feature value H_L5_mean represents the average statistical feature obtained within a 100-millisecond timeframe of host IP traffic data. For the collected IoT traffic, categorical features are numerically encoded using one-hot encoding, and the feature data is standardized using MinMax to eliminate the scale differences between features, resulting in a high-dimensional and standardized dataset.

Table 1. Description of Specific Statistical Features on the N-BaIoT Dataset

Feature Type	Abbreviation	Details
Host-IP	H	Traffic data from specific Internet Protocol addresses
Host-MAC&IP	MI	Traffic data from specific Internet Protocol and Media Access Control (MAC) addresses
Channel	HH	Traffic collected between specific hosts
Socket	HpHp	Traffic collected between specific hosts with port information
Network jitter	HH_jit	Statistical values of features related to time intervals between received data packets in the channel category

4.2. Feature Selection

The XGBoost algorithm is tested through extensive experiments, which include multiple thresholds for feature selection based on feature importance scores. For each input variable, the feature importance scores allow testing of subsets of features based on their importance. Initially, all 115 features are used, and subsets containing the essential features are obtained. The model's performance tends to overfit and results in decreased prediction accuracy as the number of selected features increases, creating a trade-off between the number of features and test accuracy. Therefore, the optimal combination of 30 features is selected from the initial set of 115 features. The detailed information of the selected features is provided in Table 2. These selected features are then fed into the CNN-GRU model for the classification of IoT malicious traffic.

Table 2. Top 30 Features Selected from the N-BaIoT Dataset of Baby Monitor Devices

No.	Feature	Index	No.	Feature	Index
1	HpHp_L0.01_weight	89	16	MI_dir_L3_mean	4
2	HH_L5_magnitude	21	17	MI_dir_Lo.1_variance	11
3	MI_dir_L5_variance	2	18	HH_L1_magnitude	35
4	HH_L0.1_covariance	44	19	MI_dir_L3_weight	3
5	MI_dir_L3_variance	5	20	MI_dir_L1_weight	6
6	MI_dir_L0.01_variance	14	21	HH_L0.01_covariance	51
7	MI_dir_L0.01_mean	13	22	HH_jit_L1_variance	58
8	MI_dir_L0.1_weight	9	23	MI_dir_L5_weight	0
9	MI_dir_L1_mean	7	24	HH_L0.01_magnitude	49
10	HH_L0.1_magnitude	42	25	HH_L0.01_pcc	52
11	MI_dir_L0.1_mean	10	26	HpHp_L3_std	72
12	HH_L3_magnitude	28	27	HH_L1_weight	32
13	HH_jit_L0.01_mean	61	28	HH_L5_mean	19
14	HH_L3_covariance	30	29	HH_jit_L0.1_variance	60
15	HH_jit_L0.1_mean	59	30	HH_jit_L5_mean	53

4.3. Performance Evaluation Metrics

We use standard evaluation metrics such as accuracy, precision, recall, and F1 score to assess the performance of the proposed architecture. To compute these metrics, we need to calculate true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN).

Accuracy:

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{15}$$

Precision:

$$precision = \frac{TP}{TP + FP} \tag{16}$$

Recall:

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{17}$$

F1-score:

$$F1 - score = \frac{2 \bullet precision \bullet recall}{precision + \text{recall}} \tag{18}$$

4.4. Experimental Results

The experiments were conducted in the following environment: CPU - i7-11800H, 16GB RAM, operating system - Win10, development environment - Python 3.7.12, Keras 2.4.3, and TensorFlow 2.3.0. The dataset used is the N-BaIoT dataset, specifically the traffic data from the Philips B120N/10 baby monitor device.

To evaluate and examine the proposed CNN-GRU model for detecting malicious IoT traffic, the model uses the Adam optimizer with a learning rate of 0.001. The hyperparameters beta_1 and beta_2 is set to 0.9 and 0.999, respectively. The model is trained for 20 epochs with a batch size of 256. The training and validation accuracy and loss are illustrated in Figure 3.

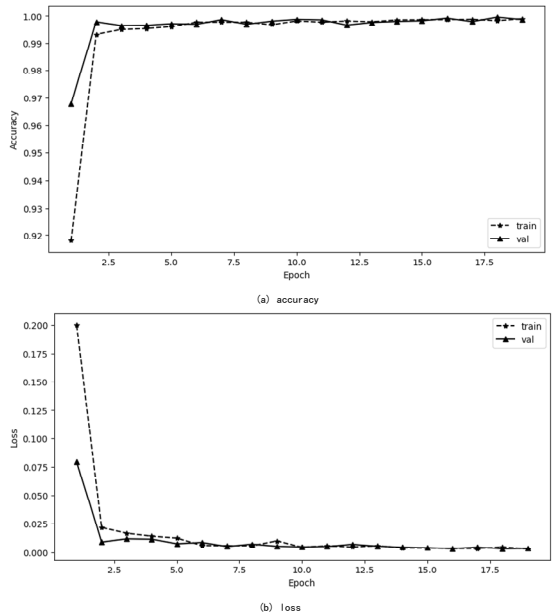

Figure 3. Training and Validation Accuracy and Loss

To highlight the effectiveness of the proposed approach, we compare it with another popular hybrid deep model, CNN-LSTM. For evaluation, we use the same metrics for both models, and they are tested and trained on the same dataset. The CNN-GRU model achieves an overall accuracy of 99.39%, precision of 99.78%, recall of 98.68%, and F1-score of 99.59%, which slightly outperforms the CNN-LSTM model in the accuracy of traffic detection for each category. The details of the models are provided in Figure 4.

Training time is an important metric for evaluating model performance. The training time for CNN-GRU is 8 milliseconds, which is significantly faster than the training time of 28 milliseconds for CNN-LSTM, indicating that the proposed model ensures efficient intrusion detection in IoT while providing faster response time, as shown in Table 3.

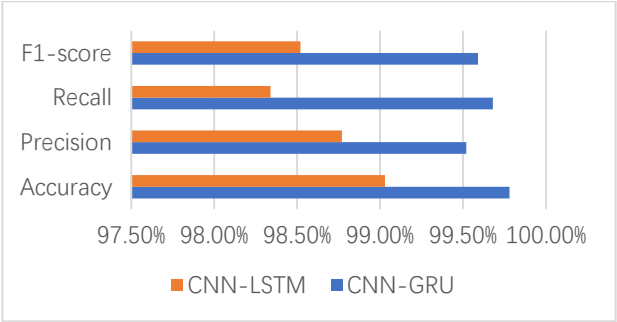

Figure 4. Comparison of CNN-LSTM and CNN-GRU experiments

Table 3. Training Time for CNN-LSTM and CNN-GRU

Model	Training Time
CNN-LSTM	28ms
CNN-GRU	8ms

Furthermore, to demonstrate the effectiveness of the model, Table 4 presents a comparison with some classical classifiers from domestic literature. It can be observed that the proposed CNN-GRU model outperforms other models in terms of accuracy, F1-score, precision, and recall, achieving 99.78%, 99.59%, 99.52%, and 99.68% respectively.

Table 4. Experimental Result Comparison (%)

Model	Dataset	Accuracy	Precision	Recall	F1-score
LR-ANN [13]	N-BaIoT	-	90.99%	90.09%	90.93%
SVM [14]	N-BaIoT	94.94%	90.84%	-	-
AE [15]	N-BaIoT	98.85%	98.88%	99.10%	98.95%
My model	N-BaIoT	99.78%	99.52%	99.68%	99.59%

5. Conclusion

In this paper, we proposed an intrusion detection method for the Internet of Things (IoT) and developed a hybrid deep learning model to address the diverse potential security threats in IoT devices. We utilized the N-BaIoT dataset and applied XGBoost feature selection to reduce feature redundancy in the IoT dataset. By integrating CNN and GRU, we achieved comprehensive and effective feature learning. Furthermore, we conducted a comprehensive performance analysis by comparing our proposed model with state-of-the-art classification models. The experimental results demonstrate that our proposed IoT intrusion detection model performs well in terms of accuracy, precision, recall, and F1-score, while maintaining a shorter training time.

In future work, our goal is to further enhance the model's detection adaptability by training it on a larger variety of IoT device data, ensuring efficient detection performance.

References

[1] Gupta B B, Dahiya A. Distributed Denial of Service (DDoS) Attacks: Classification, Attacks, Challenges and Countermeasures [M]. CRC press, 2021.

[2] Chaabouni N, Mosbah M, Zemmari A, et al. Network intrusion detection for IoT security based on learning techniques [J]. IEEE Communications Surveys & Tutorials, 2019, 21(3): 2671-2701.

[3] Makhdoom I, Abolhasan M, Lipman J, et al. Anatomy of threats to the internet of things[J]. IEEE communications surveys & tutorials, 2018, 21(2): 1636-1675.

[4] Angrishi K. Turning internet of things (iot) into internet of vulnerabilities (iov): Iot botnets[J]. arXiv preprint arXiv: 1702. 03681, 2017.

[5] Bhunia S S, Gurusamy M. Dynamic attack detection and mitigation in IoT using SDN[C]//2017 27th International telecommunication networks and applications conference (ITNAC). IEEE, 2017: 1-6.

[6] Hodo E, Bellekens X, Hamilton A, et al. Threat analysis of IoT networks using artificial neural network intrusion detection system [C]//2016 International Symposium on Networks, Computers and Communications (ISNCC). IEEE, 2016: 1-6.

[7] Chakraborty D, Narayanan V, Ghosh A. Integration of deep feature extraction and ensemble learning for outlier detection [J]. Pattern Recognition, 2019, 89: 161-171.

[8] Wang Yun, Yu Yao, Zhao Yujia, et al. Intrusion Detection and Defense Mechanism Based on SDN in Home Internet of Things [J]. Control Engineering, 2021, 28(05): 1027-1032. DOI: 10. 14107/ j.cnki.kzgc.20200900.

[9] Alkahtani H, Aldhyani T H H. Botnet attack detection by using CNN-LSTM model for Internet of Things applications[J]. Security and Communication Networks, 2021, 2021: 1-23.

[10] Alkahtani H, Aldhyani T H H. Botnet attack detection by using CNN-LSTM model for Internet of Things applications[J]. Security and Communication Networks, 2021, 2021: 1-23.

[11] Li Xiaojia, Zhao Guosheng, Wang Yang, Ning Ke. IoT Intrusion Detection Model for CNN and RNN Improvement [J/OL]. Computer Engineering and Applications: 1-10 [2023-04-21]. http://kns.cnki. net/kcms/ detail/11. 2127. TP. 2023 0302. 1415.004.html.

[12] Liu J, Sun X, Jin J. Intrusion detection model based on principal component analysis and recurrent neural network[J]. Journal of Chinese Information Processing, 2020, 34(10): 105-112.

[13] Zhou Jieying, He Pengfei, Qiu Rongfa, et al. Intrusion Detection Research Based on the Fusion of Random Forest and Gradient Boosting Tree[J]. Journal of Software, 2021, 32(10): 3254-3265. DOI: 10.13328/j.cnki.jos.006062.

[14] Abbasi F, Naderan M, Alavi S E. Intrusion detection in iot with logistic regression and artificial neural network: further investigations on n-baiot dataset devices[J]. Journal of Computing and Security, 2021, 8(2): 27-42.

[15] Nõmm S, Bahşi H. Unsupervised anomaly based botnet detection in IoT networks [C]//2018 17th IEEE international conference on machine learning and applications (ICMLA). IEEE, 2018: 1048-1053.

[16] Hezam A A, Mostafa S A, Baharum Z, et al. Combining Deep Learning Models for Enhancing the Detection of Botnet Attacks in Multiple Sensors Internet of Things Networks[J]. JOIV: International Journal on Informatics Visualization, 2021, 5(4): 380-387.