# Intrusion Detection on IoT Based on Deep Residual Network with Attention Mechanisms

Bo Cui[1,2,*], Yachao Chai[1,2], Zhen Yang[1,2]
[1]College of Computer Science, Inner Mongolia University, Hohhot 010000, China
[2]Engineering Research Center of Ecological Big Data, Ministry of Education, Hohhot 010000, China
cscb@imu.edu.cn, [32009072,32209059]@mail.imu.edu.cn

***Abstract*—Connected devices in IoT systems usually have low computing and storage capacity and lack uniform standards and protocols, making them easy targets for cyber-attacks. Implementing security measures such as cryptographic authentication, authentication, access control, and firewalls for IoT devices and their inherent vulnerabilities are not sufficient to address cyber-attacks in the IoT environment. To improve the defensive capabilities of IoT systems, some research has focused on using deep learning techniques to provide new solutions for intrusion detection systems. However, some existing deep learning-based intrusion detection methods suffer from inadequate feature extraction and insufficient model generalization capability. To address the shortcomings of existing detection methods, we propose an intrusion detection model based on temporal convolutional residual modules. An attention mechanism is introduced to evaluate the feature scores and help the model focus on more important features, thus improving the detection performance of the model. We conducted extensive experiments on the ToNIoT dataset and the UNSW-NB15 dataset, and the proposed model achieves 99.55% and 89.23% accuracy on the ToN_IoT dataset and the UNSW-NB15 dataset, respectively, which is 0.14% and 15.3% improvement compared with the current state-of-the-art models. This indicates that the model has better detection performance.**

***Index Terms*—IoT, Cyber Attacks, Intrusion Detection, Deep Learning, Attention Mechanism**

## I. Introduction

As 5G, artificial intelligence, and big data propel IoT into a new development phase, its security challenges become more pronounced. IoT devices, often with limited computing and storage capabilities and lacking standardized protocols, are vulnerable to cyber-attacks. This vulnerability is exacerbated by manufacturers' inadequate implementation of strict security measures, leaving devices with numerous weaknesses [1]. For instance, the Mirai botnet demonstrated the potential for large-scale distributed denial of service attacks via IoT [2]. Additionally, the common use of wireless networks in IoT allows intruders to eavesdrop and potentially access private information [3]. These cyberattacks threaten not just the functionality of devices but also the privacy and data of users. Thus, developing an effective security strategy to mitigate these risks in IoT is a crucial area of research [4].

Security measures like cryptographic authentication, access control, and firewalls are not fully effective against IoT cyberattacks [5]. Many studies focus on machine or deep learning-based intrusion detection systems (IDS) to improve IoT security [6]. IDS, a proactive network security device, monitors network transmissions and responds to suspicious activities [7]. IDS solutions are categorized into signature-based, anomaly, and hybrid approaches [6]. Signature-based methods are less efficient for IoT due to the need for constant updates, while anomaly-based methods are more effective for unknown attacks and require less human intervention. Hybrid approaches combine both methods. However, in IoT environments, anomaly detection is more crucial due to the limitations of signature-based methods in detecting unknown attacks.

Some existing anomaly intrusion detection systems use traditional machine learning techniques to build IDS models [8]. However, due to the high speed and volume of data generated by the IoT, and traditional machine learning techniques rely heavily on the feature engineering techniques employed to extract representative features from the large amount of unstructured data generated by IoT devices, their performance deteriorates when they are applied to large-scale and high-dimensional data. Therefore, several studies have focused on using deep learning techniques to provide new solutions [9].

Deep learning automates feature extraction and classification by employing multi-layer neural networks and leveraging substantial data and computational resources. This approach excels at feature learning without human intervention and is highly effective for managing extensive or intricate datasets. Deep learning's robust data processing, feature learning capabilities, and capacity to detect unfamiliar attacks make it an advanced solution for IoT intrusion detection [9]–[12]. Consequently, researchers have successfully applied deep learning to IoT intrusion detection, yielding favorable outcomes [9]–[14].

Several issues exist with current deep learning-based IoT intrusion detection methods. Firstly, these methods often rely on outdated data sources like NSL-KDD and KDD-99 for evaluation [9]–[13], lacking current IoT attack data. Secondly, the complexity of many detection models limits their IoT applicability [11]–[14]. Additionally, some methods overlook spatial and temporal features of IoT traffic data, leading to inadequate feature extraction [12]. Also, challenges in handling heterogeneous IoT data result in limited model generalization [15]. Furthermore, the scarcity of labeled IoT data hampers classifier performance.

In our work, we introduce an improved residual network with three residual modules, each incorporating a CONV-LSTM sub-network, and an attention module, simplifying the model's structure. Evaluation with the ToN_IoT and UNSW-NB15 datasets shows superior performance of our model. Our

contributions are summarized as follows:

- Considering the spatiotemporal characteristics of IoT traffic data, we proposed an improved residual network structure that avoids the performance impact of extracting only a single feature.
- We introduced an attention mechanism in the model to compute weights representing the importance of different features to help the model focus on the most important features.
- Higher detection accuracy. The performance of the algorithm proposed in this chapter in terms of detection accuracy compared with some current state-of-the-art methods;
- Stronger generalization ability. Under a certain amount of data, this chapter improves the expressiveness of the model by increasing the network width; and optimizes the loss function to reach the global optimum.

## II. Related Work

Recent research in IoT security has explored various machine learning methods. Hasan et al [16] developed a robust algorithm using machine learning for detecting IoT cyber-attacks, showing improved accuracy over existing models. Ravi et al [17] focused on DDoS attack mitigation in IoT using an SDN-Cloud architecture. Zhang et al [18] introduced a method for network traffic classification using PCA and Gaussian Parsimonious Bayes. Despite their advancements, these machine learning methods face limitations, such as heavy reliance on feature engineering, reduced performance with large-scale and high-dimensional data, and challenges in addressing unknown attacks in IoT environments.

Deep learning, with its ability to handle large datasets, offers significant advantages over traditional machine learning, especially in IoT security applications characterized by complex and diverse data. This capability is crucial for modeling complex features and detecting a variety of intrusive behaviors in IoT systems. For instance, Mohamed et al [19] utilized DeepIFS with gated recurrent units and attention mechanisms for intrusion detection. Liu et al [20] applied a federated learning approach with LSTM and convolutional neural networks to capture both temporal and spatial data features. The effectiveness of Recurrent Neural Networks (RNNs) in handling time-series data in IoT security is evident in research like that of Yan et al [21], who employed Variational Auto-Encoders for intrusion detection, and Gao et al [22], who developed an LSTM-GaussianNB architecture for outlier detection. Further advancing this field, Parra et al [23] combined CNNs and LSTMs in a cloud-based system to effectively detect phishing and botnet attacks, surpassing the performance of single LSTM models.

In recent research, Khan et al [24] developed the XSRU-IoMT model for detecting complex attacks in medical IoT, but its testing was limited to a single dataset. Wu et al [25] introduced LuNet, a CNN-RNN network with limited generalization ability, and later improved it with the Densely-ResNet [15] for better security across different IoT layers. However, this model also faced challenges in complexity and generalization. Latif et al [26] suggested the DnRaNN model

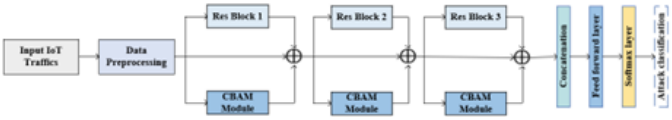



Fig. 1: Model architecture diagram.

to enhance generalization, but lacked sufficient experimental validation. Addressing these gaps, this paper presents a new intrusion detection model featuring temporal convolutional residual modules with CONV-LSTM sub-networks and attention modules. This design simplifies the model while effectively learning spatiotemporal representations of IoT data, ensuring high accuracy and generalization, tested on the ToN_IoT and UNSW-NB15 datasets.

## III. Proposed Model

### A. Model Overview

To address intrusion detection in IoT environments, researchers have explored the use of convolutional neural networks (CNNs) [27] for feature extraction. However, CNNs, primarily suited for static environments, lack the capability to model the sequential data characteristic of IoT. To overcome this, Recurrent Neural Networks (RNNs) have been applied, leveraging their temporal layers to capture and learn from the sequential nature of IoT traffic data [28]. While effective for sequential data management, RNNs face challenges like gradient vanishing or exploding, impacting their prediction accuracy in IoT intrusion detection.

This chapter explores models that blend deep learning with attention mechanisms for analyzing IoT traffic data, as discussed in [29]. It integrates CNN and LSTM units within a revised ResNet architecture [30]. Attention mechanisms in the model assign weights to different features to mitigate the loss of crucial information in high-dimensional data.

The described ResNet model is structured into three main module pairs. Each pair comprises a residual module and a ResBlock-CBAM module, interconnected with jump connections to facilitate information and gradient flow. The network processes the sum of the outputs from the final module pair through a classification layer for prediction. This configuration, depicted in Figure 1, effectively counters typical deep learning challenges such as vanishing and exploding gradients, while demonstrating robust generalization capabilities.

## IV. Residual Network Architecture

The residual module in this chapter, detailed in Figure 2, integrates a CNN unit with an LSTM unit. The CNN unit comprises three distinct convolutional layers. The first layer, with its 1x1 kernel, reduces the dimensionality of the input feature map. The second layer, the core of the CNN, uses a 3x3 kernel for extracting and transforming features through nonlinear processing. Finally, the third layer, also with a 1x1 kernel, restores the feature map to its original dimension. After processing through these convolutional and pooling layers, the extracted features are then fed into the LSTM unit for further analysis and transformation.

Assuming that the input data is x, the convolution layer in the proposed model extracts the spatial features of the given data and performs the convolution operation on the input data to obtain the output:

$$X = f(\mathbf{w}_i \otimes \mathbf{x}_i + b_i). \tag{1}$$

where $f(\cdot)$ is the activation function, $w_i$ is the convolution kernel weight, and $b_i$ is the corresponding bias. and using Rectified Linear Unit (ReLU) as the activation function:

$$f(x) = \max(0, X). \tag{2}$$

The first convolution operation is performed in the residual unit, using a 1 × 1 convolution kernel, reducing the number of channels of the input feature map to C/4 and the size of the output feature map to H × W × C/4; Batch Normalization (BN) and ReLU activation operations are performed on the output feature map:

$$y_i = \lambda x_i' + \varphi. \tag{3}$$

where $\lambda$ and $\varphi$ are learning parameters and $x_i'$ is calculated as follows:

$$\mu_\beta = \frac{1}{m}\sum_{i=1}^{m} x_i. \tag{4}$$

$$\sigma_\beta^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_\beta)^2. \tag{5}$$

$$x_i' = \frac{x_i - \mu_\beta}{\sqrt{\sigma_\beta^2 - \varepsilon}}. \tag{6}$$

The second and third convolution operations are performed using 3×3 and 1×1 con-volution kernels, respectively, and each convolution operation is followed by a BN and ReLU activation operation. The output of the previous layer is then downsampled by the pooling layer to compress its features, making the data feature dimensionality reduced by:

$$h_j^l = \text{down}\left(h_j^{(l-1)} M^l\right). \tag{7}$$

The maximum pooling layer ($M^l$) in convolutional neural networks is crucial for enhancing model efficiency and generalization. It reduces feature map size through downsampling, leading to computational efficiency, prevention of overfitting, feature invariance, and selection. In summary, maximum pooling compresses features efficiently, ensures stability against input variations, and prioritizes important data characteristics. The pooling operation converts $M$ to $Z = [z^1, z^2, z^3, \ldots, z^C]$ of size $1 \times 1 \times C$, where $z$ is calculated as

$$z^C = \max(m_{i,j}). \tag{8}$$

After local features are extracted by CNN, long-distance features of these local features are extracted using LSTM. The following is the process of transforming the specific input features of the LSTM unit:

The input vector $\mathbf{X}$ at the current moment is input to the LSTM cell. The output value $i_t$ of the input gate of the LSTM cell is calculated by the sigmoid function, and this value determines how the input vector $\mathbf{x}$ affects the state $C_t$. $i_t$ ranges between 0 and 1. The output value $f_t$ of the forgetting gate is computed by the sigmoid function. This value determines which information in the state $C_{t-1}$ of the previous moment needs to be forgotten. $f_t$ ranges from 0 to 1. Updating the state. The current moment's state $C_t$ can be updated by adding the result of the dot product of $i_t$ and $\mathbf{X}$ to the previous moment's state $C_{t-1}$ and subtracting the result of the dot product of $f_t$ and the previous moment's state $C_{t-1}$. The output value $o_t$ of the output gate is calculated by the sigmoid function, which determines which information in the current moment's state $C_t$ needs to be output. $o_t$ ranges from 0 to 1. The hidden state $h_t$ of the current moment can be obtained by performing a dot product operation between the state $C_t$ of the current moment and the output value $o_t$ of the output gate. Finally, the output is passed to the next layer for calculation.

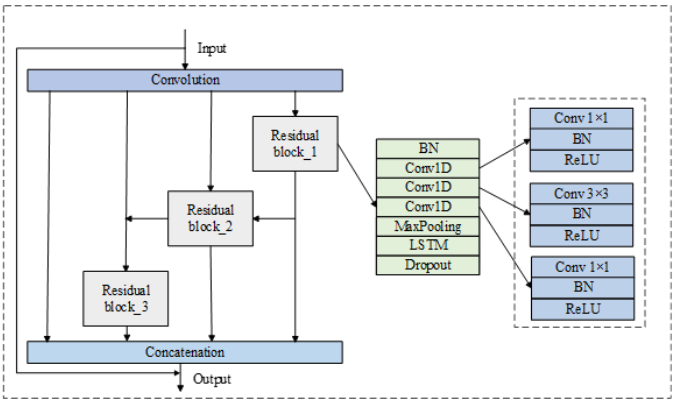



Fig. 2: Residual network structure diagram.

In this way, the input features can be better represented by the transformation of the LSTM cells, thus improving the performance of the model. The dropout layer is used to avoid the overfitting problem during training and to improve the model generalization ability. ReLU is then chosen as the activation function to overcome the gradient disappearance problem and to speed up the training speed.

## V. Convolutional Block Attention Mechanism

In the IoT environment, network traffic data often contains different features that are of different importance for intrusion detection. Therefore, this chapter introduces the CBAM module [31] in the model to help the model distinguish the importance of features in order to obtain important features and improve the performance of the model. As shown in Figure 3. The structures of the channel attention module and the spatial attention module are shown in Figure 4 and Figure 5, respectively. The specific computational procedure of the CBAM module is described as follows: The specific structure of the channel attention module is shown in Figure 4. For a given feature map F, the channel attention module first performs Global Average Pooling (GAP) to obtain the average value of each channel, and then performs feature mapping through two fully connected layers to obtain the weight coefficients of each channel. Finally, the weight coefficients are applied to the feature map to emphasize the important channels and suppress the unimportant ones. The whole process is as follows:

$$F' = M_C(F) \otimes F \quad . \tag{9}$$

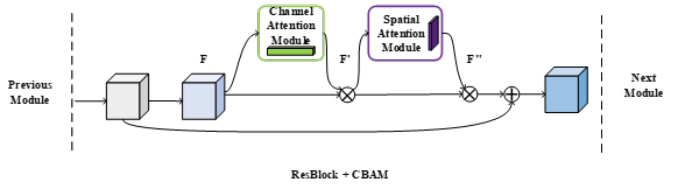



Fig. 3: CBAM module diagram.

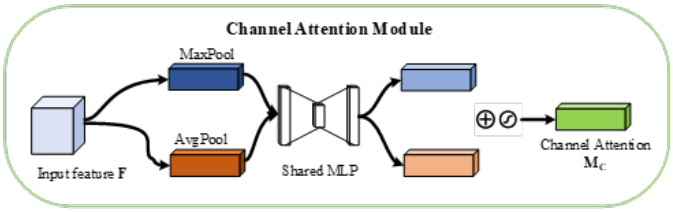



Fig. 4: Channel attention module diagram.

$$F'' = M_C(F') \otimes F'. \tag{10}$$

where $\otimes$ denotes element-by-element multiplication. The channel attention module focuses on what is meaningful in the input data. CBAM has two pools: maximum pooling (MaxPool) and average pooling (AvgPool). Pooling allows extracting high-level features, and different pooling means extracting richer high-level features. Each channel of the feature map contains valuable feature information, with some channels being more important than others. To efficiently compute channel attention, the channel attention module employs a spatial dimensionality method that compresses the input feature map. Instead of using a single pooling method, the module utilizes both average pooling and maximum pooling methods, demonstrating that the dual pooling method offers more robust representational power. The computational procedure is as follows:

$$M_C(F) = \sigma\left(\text{MLP}\left(\text{AvgPool}(F)\right) + \text{MLP}\left(\text{MaxPool}(F)\right)\right). \tag{11}$$

Figure 5 illustrates the structure of the spatial attention module. It processes feature maps through two convolution layers to obtain weight coefficients of each pixel point, which are then applied to the feature map to emphasize important pixel points and suppress unimportant ones. This enhances model generalization ability and complements the channel attention module, as shown below:

$$M_C(F) = \sigma\left(\text{Conv}\left[\text{AvgPool}(F); \text{MaxPool}\right]\right). \tag{12}$$

where $\sigma(\cdot)$ denotes the Sigmoid function, MaxPool$(\cdot)$ denotes the maximum pooling, AvgPool$(\cdot)$ denotes the average pooling, MLP$(\cdot)$ denotes the multilayer perceptron, and Conv$(\cdot)$ denotes the 3D convolutional layer. Finally, the model uses the Softmax layer to calculate the final traffic class using the output of the residual module and the CBAM module. The outputs of the residual and CBAM modules and the raw inputs are fed into the fully connected layer after manipulation. The fully connected layer then multiplies the weight matrix with the input vector and adds the bias as follows:

$$z_j = \mathbf{w}_j \cdot \mathbf{X} + b_j = w_{j1}X_1 + w_{j2}X_2 + \cdots + w_{jn}X_n + b_j. \tag{13}$$

where $\mathbf{X}$ is the input of the fully connected layer, $\mathbf{w}_j$ is the weight of the $j$-th class of features, and $b_j$ is the bias term.

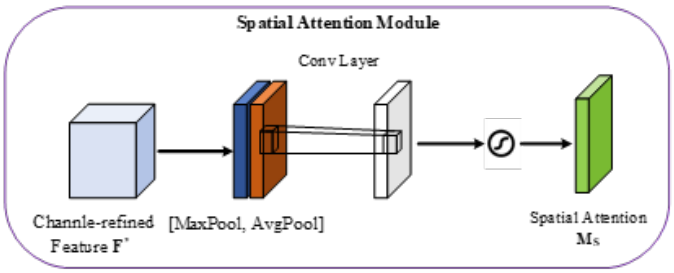



Fig. 5: Spatial attention module diagram.

The feedforward layer represents the captured spatiotemporal features as a linear representation suitable for predicting the final category labels using Softmax operations, where each category is assigned a certain probability score and the category with the highest probability is considered as the final model prediction, as shown below:

$$p = \text{SoftMax}(z_j) = \frac{e^{z_j}}{\sum_K e^{z_j}}. \tag{14}$$

$$\hat{y} = \arg\max(p). \tag{15}$$

where $z_j$ denotes the output of the feedforward layer and $p$ denotes the probability score. The training model is calculated to minimize the cross-entropy loss according to equation (16):

$$\text{Loss} = -\sum \left(y_i \cdot \log \hat{y}_i + (1 - y_i) \cdot \log(1 - \hat{y}_i)\right). \tag{16}$$

where $y_i$ denotes the actual label and $\hat{y}_i$ denotes the model prediction label.

## VI. Model Evaluation and Discussions

### A. Dataset Description

In our experiments, we assess the model's generalization ability, a critical performance metric indicating how well a model can perform on new datasets. To evaluate our proposed model, we utilize the ToN_IoT dataset [13], which originates from a real IoT system developed by the Cyber IoT Laboratory at ADFA in New South Wales, Australia. This dataset comprises both normal data and data from nine attack categories, including password, scan, ransomware, back-door, denial of service, distributed denial of service, MITM, injection, and XSS attacks. The raw network packets for the UNSW-NB15 [32] dataset are created by the IXIA PerfectStorm tool at the Australian Cyber Security Centre's Cyber Scope Lab, which can be used to generate a mix of modern normal activity and synthetic contemporary attack behavior. The dataset contains a total of 49 features and 9 common attacks. The normal information accounts for 88% of the dataset size and the attack information accounts for 12%. We used this dataset to test the generalization ability of the model.

We use several metrics to evaluate the performance of the proposed model, i.e., Accuracy, Precision, Recall, and F1-score because they are widely used to evaluate deep learning algorithms.

TABLE I: Performance variation at different parameters.

| Performance Parameters | | Batch Size | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | 16 | 32 | 64 | 128 | 256 | 512 | 1024 |
| LR=0.01 | Accuracy | 0.9452 | 0.9476 | **0.9682** | 0.9360 | 0.9508 | 0.9156 | 0.9074 |
| | Precision | 0.9352 | 0.9383 | **0.9642** | 0.9269 | 0.9445 | 0.9022 | 0.8955 |
| | Recall | 0.9473 | 0.9487 | **0.9660** | 0.9340 | 0.9478 | 0.9193 | 0.9038 |
| | F1-score | 0.9406 | 0.9430 | **0.9651** | 0.9302 | 0.9461 | 0.9092 | 0.8993 |
| LR=0.001 | Accuracy | 0.9870 | 0.9883 | 0.9931 | 0.9945 | 0.9943 | **0.9955** | 0.9908 |
| | Precision | 0.9862 | 0.9866 | 0.9920 | 0.9937 | 0.9931 | **0.9951** | 0.9888 |
| | Recall | 0.9852 | 0.9876 | 0.9929 | 0.9941 | 0.9943 | **0.9950** | 0.9911 |
| | F1-score | 0.9857 | 0.9871 | 0.9925 | 0.9939 | 0.9937 | **0.9950** | 0.9899 |
| LR=0.0001 | Accuracy | 0.9869 | **0.9924** | 0.9923 | 0.9759 | 0.9856 | 0.9552 | 0.9280 |
| | Precision | 0.9857 | **0.9918** | 0.9910 | 0.9719 | 0.9839 | 0.9505 | 0.9244 |
| | Recall | 0.9856 | **0.9915** | 0.9920 | 0.9753 | 0.9845 | 0.9511 | 0.9162 |
| | F1-score | 0.9856 | **0.9916** | 0.9915 | 0.9735 | 0.9842 | 0.9508 | 0.9201 |

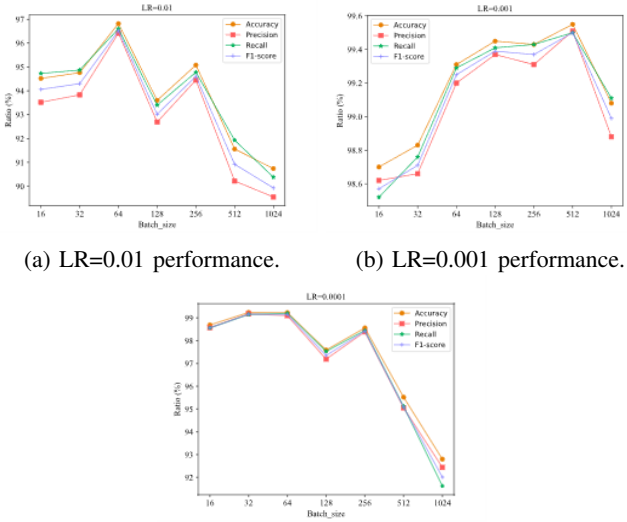
### *B. Hyperparameter Settings*

Hyperparameters in deep learning are crucial settings that are manually adjusted during training to optimize model performance. In our experiments, we focus on three key hyperparameters: learning rate (LR), batch size, and epoch. The details are summarized in Table I.

- Learning rate (LR): LR is a critical hyperparameter that controls the rate of learning in the neural network. We experiment with three LR values to determine the best performance.
- Batch size: Batch size refers to the number of samples used for each training iteration. It impacts memory usage, optimization, and training speed. We consider a range of batch sizes, including 16, 32, 64, 128, 256, 512, and 1024.
- Epochs: Epochs represent the number of times the entire training dataset is processed during training. Choosing an appropriate number of epochs is crucial to prevent overfitting or underfitting. In our experiments, we set the number of epochs to 100.

We split the ToN_IoT dataset into an 80% training set and a 20% test set. Initially, we set the learning rate to 0.01, fixed the epoch at 100, and experimented with batch sizes ranging from 16 to 1024. Results for these experiments are summarized in Table 3, with detailed performance comparisons in Figure 6a. The model achieved its highest accuracy of 96.82% with a learning rate of 0.01 and a batch size of 64. The lowest accuracy for this learning rate was 90.74% with a batch size of 1024. Other batch sizes consistently yielded accuracy and performance scores above 90%.

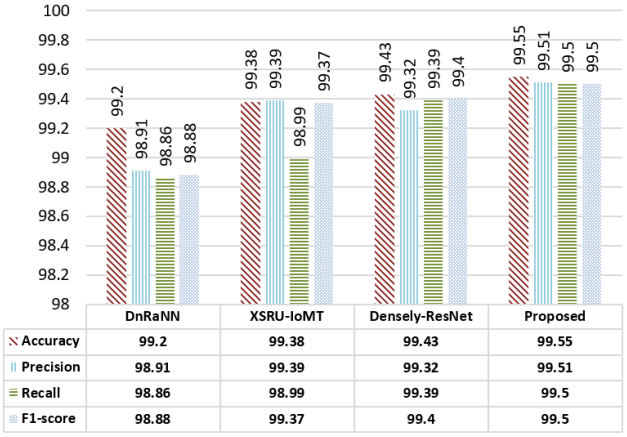
In the second stage, we kept the learning rate at 0.001 and the epoch fixed at 100 while varying batch sizes from 16 to 1024. Results are presented in Table 3, and performance comparisons are shown in Figure 6b. The model achieved its best test accuracy of 99.55% with a batch size of 512. Batch sizes of 64, 128, 256, and 1024 also yielded accuracies exceeding 99%, while smaller batch sizes of 16 and 32 slightly decreased accuracy but remained close to 99%. Lastly, with a learning rate of 0.0001 and keeping other parameters consistent with the first two stages, performance comparisons are depicted in Figure 6c. The highest test accuracy was 99.24% with a batch size of 32. Batch sizes of 16, 64, and 256 also resulted in accuracies above 98%. For other batch sizes, accuracy and performance scores decreased, but remained above 90%. Overall, the comparative experimental results suggest that the proposed model performs best with a learning rate of 0.001, 100 epochs, and a batch size of 512.




(a) LR=0.01 performance. (b) LR=0.001 performance.

(c) LR=0.0001 performance.

Fig. 6: Performance at different learning rate




| | DnRaNN | XSRU-IoMT | Densely-ResNet | Proposed |
|---|---|---|---|---|
| Accuracy | 99.2 | 99.38 | 99.43 | 99.55 |
| Precision | 98.91 | 99.39 | 99.32 | 99.51 |
| Recall | 98.86 | 98.99 | 99.39 | 99.5 |
| F1-score | 98.88 | 99.37 | 99.4 | 99.5 |

Fig. 7: Performance comparison with other models.

### *C. Generalization Performance Comparison*

We compared our proposed model with current state-of-the-art methods to assess its effectiveness and performance differences. The results are depicted in Figure 7. It's evident from the figure that our proposed model outperforms existing models on the ToN_IoT dataset across various evaluation metrics, including Accuracy, Precision, Recall, and F1-score. The improvements are notable, with an increase of 0.35% in Accuracy, 0.17% in Precision, and 0.12% in Recall, underscoring the validity and superiority of our model. In addition, we compared the generalization ability of the model with some DL methods on the UNSW-NB15 dataset, as shown in Table 4. The values of our proposed model for each metric are 89.23%, 88.83%, 87.77% and 88.25%, respectively, which is a 15.3% improvement compared to the results of the Densely-ResNet model. However, the reason for the poor detection results is the severe class imbalance problem in the UNSW-NB15 benchmark, which often leads to poor model generalization performance. As can be seen from the table, the proposed model significantly outperforms the compared models in terms of accuracy, precision, recall, F1-score, and other evaluation

TABLE II: Comparison of model generalization performance.

| Model | Accuracy | Precision | Recall | F1-score |
|---|---|---|---|---|
| LSTM | 0.7015 | 0.7754 | 0.7724 | 0.8628 |
| LuNet | 0.7267 | 0.7850 | 0.8290 | 0.8750 |
| Densely-ResNet | 0.7393 | 0.8094 | 0.8668 | 0.8811 |
| CNN | 0.8163 | 0.8094 | 0.8578 | 0.8121 |
| LSTM-ResNet | **0.8923** | **0.8883** | **0.8777** | **0.8825** |

metrics, except for the Densely-ResNet model. This indicates that the proposed model has better generalization capability.

## VII. Conclusion

In this paper, we proposed a deep learning model based on the temporal convolution residual module and attention mechanism for IoT anomaly detection and analyze the model in depth on the ToN_IoT dataset, the latest publicly available IoT dataset. In addition, the proposed model is evaluated on the UNSW-NB15 dataset and its generalization performance is compared with several deep learning methods. As a result, ResNet can achieve state-of-the-art detection accuracy on UNSW-NB15 benchmarks while maintaining a low false positive rate. We determined the effectiveness and stability of the proposed model based on the evaluation results and encourage researchers to use it for other intrusion detection tasks in the future.

## VIII. Acknowledgment

This work is supported by the National Natural Science Foundation of China (61962042) and Science and Technology Program of Inner Mongolia Autonomous Region (2020GG0188), and Natural Science Foundation of Inner Mongolia (2022MS06020), and the Central Government Guides Local Science and Technology Development Fund (2022ZY0064), and the University Youth Science and Technology Talent Development Project (Innovation Group Development Plan) of Inner Mongolia A. R. of China (Grant No. NMGIRT2318).

## References

[1] Bertino, E., Islam, N.: Botnets and internet of things security. Computer 50(2), 76-79 (2017).

[2] Kolias, C., Kambourakis, G., Stavrou, A., et al.: DDoS in the IoT: Mirai and other Botnets. Computer 50(7), 80-84 (2017).

[3] Abomhara, M., Koien, G, M.: Cyber security and the internet of things: vulnerabilities, threats, intruders and attacks. Journal of Cyber Security and Mobility, 65–88 (2015).

[4] Thakkar, A., Lohiya, R.: A review on machine learning and deep learning perspectives of IDS for IoT: recent updates, security issues, and challenges. Archives of Computational Methods in Engineering 28, 3211-3243 (2021).

[5] Al-Garadi, M, A., Mohamed, A., Al-Ali, A, K., et al.: A survey of machine and deep learning methods for internet of things (IoT) security. IEEE Communications Surveys & Tutorials 22(3), 1646-1685 (2020).

[6] Babu, M, R., Veena, K, N.: A survey on attack detection methods for IoT using machine learning and deep learning. In: 2021 3rd International Conference on Signal Processing and Communication (ICPSC). pp. 625-630. IEEE, (2021).

[7] Denning, D, E.: An intrusion-detection model. IEEE Transactions on Software Engineering (2), 222-232 (1987).

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

[9] Alsoufi, M, A., Razak, S., Siraj, M, M., et al.: Anomaly-based intrusion detection systems in IoT using deep learning: a systematic literature review. Applied sciences 11(18), 8383 (2021).

[10] Selvapandian, D., Santhosh, R.: Deep learning approach for intrusion detection in IoT-multi cloud environment. Automated Software Engineering 28(2), 1-17 (2021).

[11] Khoa, T, V., Saputra, Y, M., Hoang, D, T., et al.: Collaborative learning model for cyberattack detection systems in IoT industry 4.0. In: 2020 IEEE Wireless Communications and Networking Conference (WCNC). pp. 1-6. IEEE, (2020).

[12] Haider, A., Khan, M, A., Rehman, A., et al.: A real-time sequential deep extreme learning machine cybersecurity intrusion detection system. Computers Materials Continua 66(2), 1785-1798 (2021).

[13] Booij, T, M., Chiscop, I., Meeuwissen, E., et al.: ToN_IoT: the role of heterogeneity and the need for standardization of features and attack types in IoT network intrusion data sets. IEEE Internet of Things Journal 9(1), 485-496 (2021).

[14] Tsimenidis, S., Lagkas, T., Rantos, K.: Deep learning in IoT intrusion detection. Journal of Network and Systems Management 30, 1-40 (2022).

[15] Wu, P., Moustafa, N., Yang, S., et al.: Densely connected residual network for attack recognition. In: 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom). pp. 233-242. IEEE, (2020).

[16] Hasan, M., Islam, M, M., et al.: Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches. Internet of Things 7, 100059 (2019).

[17] Ravi, N., Shalinie, S, M.: Learning-driven detection and mitigation of DDoS attack in IoT via SDN-cloud architecture. IEEE Internet of Things Journal 7(4), 3559-3570 (2020).

[18] Zhang, B., Liu, Z., Jia, Y., et al.: Network intrusion detection method based on PCA and Bayes algorithm. Security and Communication Networks 2018, 1-11 (2018).

[19] Abdel-Basset, M., Chang, V., Hawash, H., et al.: Deep-IFS: Intrusion detection approach for industrial internet of things traffic in fog environment. IEEE Transactions on Industrial Informatics 17(11), 7704-7715 (2020).

[20] Liu, Y., Garg, S., Nie, J., et al.: Deep anomaly detection for time-series data in industrial IoT: A communication-efficient on-device federated learning approach. IEEE Internet of Things Journal 8(8), 6348-6358 (2020).

[21] Li, L., Yan, J., Wang, H., et al.: Anomaly detection of time series with smoothness-inducing sequential variational auto-encoder. IEEE Transactions on Neural Networks and Learning Systems 32(3), 1177-1191 (2020).

[22] Gao, J., Gan, L., Buschendorf, F., et al.: Omni SCADA intrusion detection using deep learning algorithms. IEEE Internet of Things Journal 8(2), 951-961 (2020).

[23] Parra, G, D, L, T., Rad, P., Choo, K, K, R., et al.: Detecting internet of things attacks using distributed deep learning. Journal of Network and Computer Applications 163, 102662 (2020).

[24] Khan, I, A., Moustafa, N., Razzak, I., et al.: XSRU-IoMT: Explainable simple recurrent units for threat detection in internet of medical things networks. Future Generation Computer Systems 127, 181-193 (2022).

[25] Wu, P., Guo, H.: LuNET: a deep neural network for network intrusion detection. In: 2019 IEEE Symposium Series on Computational Intelligence (SSCI). pp. 617-624. IEEE, (2019).

[26] Latif, S., Huma, Z., Jamal, S, S., et al.: Intrusion detection framework for the internet of things using a dense random neural network. IEEE Transactions on Industrial Informatics 18(9), 6435-6444 (2021).

[27] Ferrag, M, A., Maglaras, L., Moschoyiannis, S., et al.: Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. Journal of Information Security and Applications 50, 102419 (2020).

[28] Tsimenidis, S., Lagkas, T., Rantos, K.: Deep learning in IoT intrusion detection. Journal of Network and Systems Management 30, 1-40 (2022).

[29] Guo, M, H., Xu, T, X., Liu, J, J., et al.: Attention mechanisms in computer vision: a survey. Computational Visual Media 8(3), 331-368 (2022).

[30] He, K., Zhang, X., Ren, S., et al.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 770-778. IEEE, (2016).

[31] Woo, S., Park, J., Lee, J, Y., et al.: CBAM: Convolutional block attention module. In: 2018 European Conference on Computer Vision (ECCV). pp. 3-19. Springer, Heidelberg, (2018).

[32] Moustafa, N., Slay, J.: UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In: 2015 military communications and information systems conference (MilCIS). pp. 1-6. IEEE, (2015).