future internet

MDPI

Article

Intrusion Detection in IoT Using Deep Residual Networks with Attention Mechanisms

Bo Cui [1,2], Yachao Chai [1,2], Zhen Yang [1,2] and Keqin Li [3,*]

[1] College of Computer Science, Inner Mongolia University, Hohhot 010021, China; cscb@imu.edu.cn (B.C.); 32009072@mail.imu.edu.cn (Y.C.); 32209059@mail.imu.edu.cn (Z.Y.)
[2] Engineering Research Center of Ecological Big Data, Ministry of Education, Hohhot 010021, China
[3] Department of Computer Science, State University of New York, New Paltz, NY 12561, USA
* Correspondence: lik@newpaltz.edu

Abstract: Connected devices in IoT systems usually have low computing and storage capacity and lack uniform standards and protocols, making them easy targets for cyberattacks. Implementing security measures like cryptographic authentication, access control, and firewalls for IoT devices is insufficient to fully address the inherent vulnerabilities and potential cyberattacks within 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 assess feature scores and enhance the model's ability to concentrate on critical features, thereby boosting its detection performance. We conducted extensive experiments on the ToN_IoT dataset and the UNSW-NB15 dataset, and the proposed model achieves accuracies of 99.55% and 89.23% on the ToN_IoT and UNSW-NB15 datasets, respectively, with improvements of 0.14% and 15.3% compared with the current state-of-the-art models. These results demonstrate the superior detection performance of the proposed model.

Keywords: IoT; cyber attacks; intrusion detection; deep learning; attention mechanism

Citation: Cui, B.; Chai, Y.; Yang, Z.; Li, K. Intrusion Detection in IoT Using Deep Residual Networks with Attention Mechanisms. *Future Internet* **2024**, *16*, 255. https://doi.org/10.3390/fi16070255

Academic Editors: Christos Tryfonopoulos and Nicholas Kolokotronis

Received: 13 June 2024
Revised: 15 July 2024
Accepted: 16 July 2024
Published: 18 July 2024

1. Introduction

With the leap forward in next-generation information technology such as 5G, artificial intelligence, and big data, IoT has entered a new era of development. However, connected devices in IoT systems often have low computing and storage capacity and lack uniform standards and protocols, making them easy targets for cyber attacks. Secondly, IoT device manufacturers do not implement strict security measures when producing devices, which makes current IoT devices have many vulnerabilities, which also leads to many potential security threats in the IoT system composed of IoT devices [1]. For example, the Mirai botnet is capable of executing large-scale distributed denial of service attacks via IoT devices [2]; IoT devices are usually connected through wireless networks, and intruders can obtain private information from communication channels through eavesdropping [3]. These cyberattacks not only disrupt the availability of device functionality but may also steal important information and data about the device and the users who use it. Therefore, designing a security approach that can effectively address the cyberattacks faced by IoT has been the focus of research [4].

Implementing security measures such as cryptographic authentication, access control, and firewalls for IoT devices does not fully address the cyberattack problem or guarantee absolute security in the IoT environment [5]. To improve the defense capability of IoT systems, many studies have focused on machine learning-based or deep learning-based intrusion detection methods [6]. An intrusion detection system (IDS) [7] is a network

security device that continuously monitors network traffic and issues alerts or takes proactive measures when suspicious activities are detected. The difference with other network security measures is that IDS is a proactive security protection technology. IDS solutions can be classified into three approaches: signature-based, anomaly-based, and hybrid [6]. In general, signature-based approaches are effective against known attacks. However, due to the heterogeneity, dynamics, and complexity of IoT networks, the signature-based approach is less efficient for IoT because it requires continuous human intervention to extract attack patterns and update the IDS model. On the other hand, anomaly-based approaches are effective for unknown attacks, making them advantageous in IoT as they can detect zero-day attacks with minimal human intervention. The hybrid approach combines signature-based and anomaly-based techniques. However, the use of signature-based intrusion detection methods in IoT environments is limited due to their inability to detect unknown attacks. Therefore, anomaly-based intrusion detection systems play a crucial role in IoT security.

Some existing anomaly intrusion detection systems use traditional machine learning techniques to build IDS models [8]. These traditional techniques, however, encounter challenges with the high speed and volume of data generated by IoT devices. They rely extensively on feature engineering to extract representative features from the unstructured data, leading to performance degradation when applied to large-scale and high-dimensional datasets. Consequently, several studies have shifted focus toward employing deep learning techniques to develop more effective solutions [9].

Deep learning automates feature extraction and classification by designing multilayer neural networks into architectures and using large amounts of data and computation. The advantage of this approach is that it can perform feature learning without human intervention and can achieve good results when dealing with large or complex datasets. The powerful data processing and feature learning capabilities and the ability to detect unknown attacks that deep learning has can provide advanced solutions for IoT intrusion detection [9–12]. Therefore, researchers have applied deep learning to the field of IoT intrusion detection and have achieved good results [9–14].

However, there are several obvious problems with some current deep learning-based IoT intrusion detection methods. First, most of the current methods use previous data sources, i.e., NSL-KDD, KDD-99, etc., for evaluation [9–13], which do not contain current and up-to-date attack data against the IoT. Secondly, the detection models mentioned in a significant number of methods have a complex structure [11–14], which leads to the limitation of the application of the models in the IoT environment. In addition, some methods ignore the fact that IoT traffic data has both spatial and temporal features [12], which leads to inadequate feature extraction by the models proposed in the methods and thus affects the detection performance. Moreover, some models cannot be processed for heterogeneous data in the IoT environment, which leads to a slight lack of model generalization [15], i.e., the models are not sufficiently adaptable to new data and have poor detection performance on new datasets. Finally, the limited amount of labeled data available for training deep learning models in the IoT environment seriously affects the detection performance of the classifier. In our work, we proposed an improved residual network structure, which consists of three residual modules, each containing a CONV-LSTM sub-network, and connects the module design to the attention module, which greatly reduces the complexity of the model structure. The evaluation results based on ToN_IoT with the UNSW-NB15 dataset show that our model has better performance performance than existing models. The contributions of our work are 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 paper, in terms of detection accuracy, is superior to some current state-of-the-art methods.
- Stronger generalization ability. With a certain amount of data, this paper improves the model's expressiveness by increasing the network width and optimizing the loss function to reach the global optimum.

The rest of this paper is organized as follows. Section 2 describes the intrusion detection-related work. Section 3 describes the mathematical modeling of the proposed model. Section 4 presents the experimental results and discussion. Finally, Section 5 summarizes the contributions and results of this study.

2. Related Work

In recent years, Hasan et al. [16] studied attack detection models in IoT sensors using machine learning methods. They accurately compared the performance of several machine learning techniques in predicting attacks and violations in IoT systems. A robust algorithm was developed for detecting IoT cyberattacks, particularly focusing on virtual environments. The proposed system demonstrated better detection accuracy compared to existing models. Ravi et al. [17] proposed DDoS attack mitigation and learning-driven detection in IoT through SDN-Cloud architecture. This approach aims at detecting DDoS attacks launched in IoT servers using malicious wireless IoT. Zhang et al. [18] proposed an effective method for classifying network traffic, which uses principal components analysis (PCA) to remove irrelevant features and Gaussian Parsimonious Bayes as a classifier. However, machine learning methods have certain limitations. Firstly, their performance is heavily dependent on the robustness of the feature engineering techniques employed. Secondly, their effectiveness diminishes when applied to large-scale and high-dimensional data. Lastly, their learning capability is insufficient to effectively handle unknown attacks in the IoT environment.

The most important advantage of deep learning over traditional machine learning is its superior performance on large datasets. The use of IoT systems usually generates a large amount of data, which is more complex and diverse, and contains a variety of intrusive behaviors. The ability of deep learning to automatically model complex feature sets from sample data makes deep learning more relevant in IoT security applications. For example, Mohamed et al. [19] proposed the use of DeepIFS integrated with gated recurrent units and a multi-headed attention mechanism to detect intrusions. Liu et al. [20] introduced a federated learning approach for collaborative and decentralized training on edge devices. They utilized LSTM to capture temporal representations and employed attention-enhanced convolutional neural networks to learn important spatial information. The study of Recurrent Neural Networks and their variants is important for improving the security of IoT systems, especially against time-series-based threats. For example, Yan et al. [21] used a variational autoencoder (VAE) as a network baseline and then learned potential temporal representations of the input time series for intrusion detection by RNNs. They also used FNN to parameterize the mean and variance of each time window to provide a non-smooth architecture that can operate in the absence of persistent noise. Gao et al. [22] introduced the LSTM-GaussianNB architecture for evaluating the probability of outliers in IoT data. These research works further demonstrate the suitability of recurrent neural networks as an IoT intrusion detection model. Therefore, some researchers have gone a step further and combined recurrent neural networks with other methods to achieve better detection in the field of IoT intrusion detection. For example, Parra et al. [23] proposed a distributed cloud-based approach based on CNN and LSTM to detect and mitigate phishing and botnet attacks on client devices, which outperformed a single LSTM model.

In the last two years of research work, Khan et al. [24] proposed an efficient model called XSRU-IoMT for the effective and timely detection of complex attack traffic in medical IoT. However, the authors tested it using only a single dataset, which was not sufficient to demonstrate the effectiveness of the model. Wu et al. [25] proposed a hierarchical CNN-RNN neural network LuNet, but the generalization ability of the model was poor. Later,

Wu et al. proposed a dense residual network Densely-ResNet [15] based on LuNet for secure detection of edge, cloud, and fog layers. Although the generalization ability of the model was improved, it was slightly inadequate and the structure of the model was too complex. Latif et al. [26] pointed out that their proposed dense random neural network (DnRaNN) could improve the generalization performance of the model, but there were not enough experiments to show that the generalization ability of the model was improved. Therefore, to further improve the detection accuracy of the model while maintaining good generalization ability, this paper proposes an intrusion detection model based on temporal convolutional residual modules, which consists of three residual modules, each containing a CONV-LSTM subnetwork, and connects the module design to the attention module, which greatly reduces the complexity of the model structure. The model can effectively learn the spatiotemporal representation of IoT data and obtain high accuracy while maintaining good generalization performance. All experiments are conducted on the ToN_IoT dataset and the UNSW-NB15 dataset.

3. Proposed Model

Model Overview

To achieve intrusion detection in the IoT environment, researchers have introduced convolutional neural networks [27]. However, CNNs are usually used for feature extraction in static environments and lack long-term relevance storage mechanisms, which are not suitable for modeling sequential data in IoT. Some researchers have applied RNNs to the field of IoT intrusion detection for the sequential relationship of traffic data in the IoT environment. RNNs integrate a temporal layer to capture sequential data, learning multifaceted changes through the hidden units of recurrent units. These hidden units are modified based on the data provided to the network and are continuously updated. RNNs are used for IoT security due to their efficient management of sequential data. The study of RNNs and their variants is important to improve the security of IoT systems, especially against time-series-based threats [14]. However, RNNs suffer from the gradient disappearance or explosion problem, where gradients can become too small or too large during training, leading to their unsatisfactory predictions for IoT intrusion detection.

This paper studies intrusion detection models based on the combination of deep learning and attention mechanisms [28] for intrusion detection from IoT traffic data. For the characteristics of data in IoT, this paper combines CNN and LSTM units, redesigns and improves the ResNet architecture [29], and introduces an attention mechanism for the extracted high-dimensional features to give weight information to different features to avoid the problem of failing to express important information due to the excessive dimensionality of the features.

The ResNet model proposed in this paper consists of three main pairs of modules. Each module pair contains a residual module and a ResBlock-CBAM module. A jump connection is used between each two module pairs, and after the summation operation is performed on the output of the last module pair, the output of the module pair is classified by the classification layer, which finally constitutes the main structure of the deep residual network model used in this paper. The overall architecture of the model is shown in Figure 1. This network model overcomes the disadvantages of traditional deep learning, such as gradient disappearance and gradient explosion, and has a strong generalization capability.

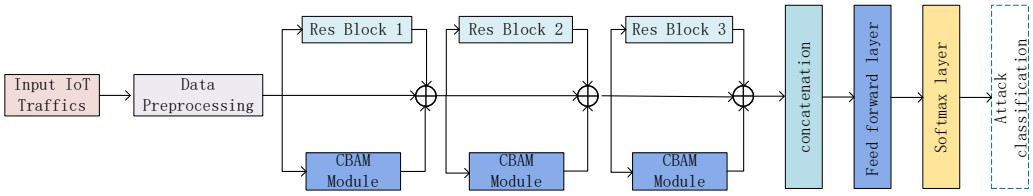


Figure 1. Model architecture diagram.

4. Residual Network Architecture

The specific structure design of the residual module proposed in this paper is shown in Figure 2. Each residual module consists of a CNN unit and an LSTM unit. The CNN unit mainly consists of three convolutional layers: the first convolutional layer has a convolutional kernel size of 1 × 1, which is used to reduce the dimensionality of the input feature map; the second convolutional layer has a convolutional kernel size of 3 × 3, which is the core part of the CNN structure and is used for feature extraction and nonlinear transformation; the third convolutional layer has a convolutional kernel size of 1 × 1, which is used to restore the dimensionality of the feature map to its original size. The proposed model extracts the features of the input data through the convolutional and pooling layers, and then the feature map is transformed and input to the LSTM unit.

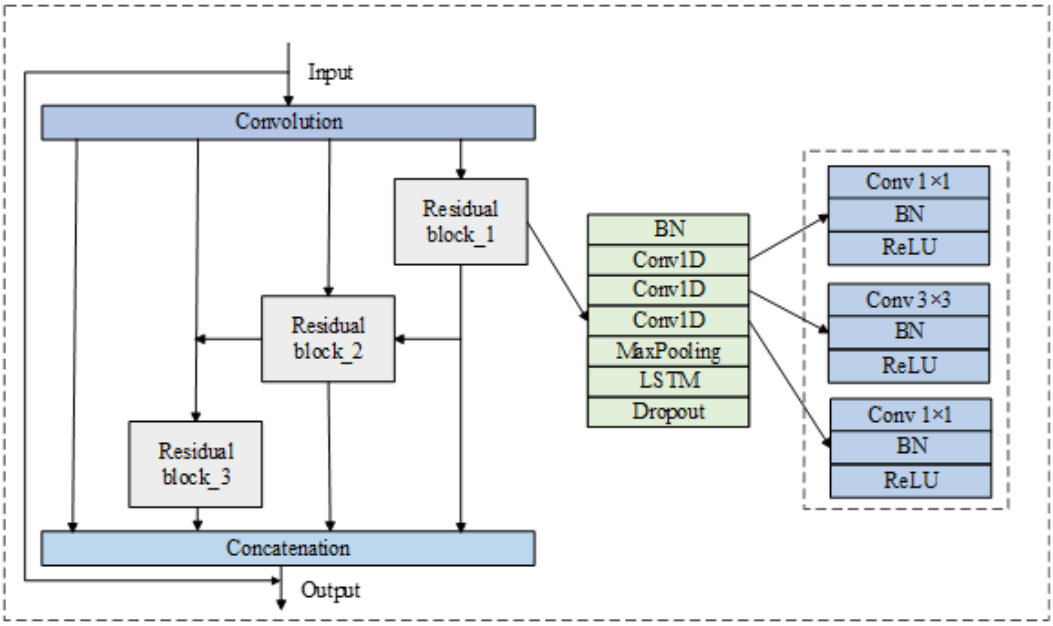


Figure 2. Residual network structure diagram.

Assuming that the input datum 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:

$$X = f(w_i \otimes x_i + b_i), \tag{1}$$

where $f(\cdot)$ is the activation function, w_i is the convolution kernel weight, and b_i is the corresponding bias. The rectified linear unit (ReLU) is used 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 λ and φ are learning parameters and x_i' is calculated as follows:

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

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

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

In these equations, μ_β represents the mean of the mini-batch, which is the average value of the β-th feature map element in a mini-batch; σ_β^2 represents the variance of the mini-batch, which is the variance of the β-th feature map element in a mini-batch; x_i represents the i-th element of the input feature map; x_i' represents the i-th element of the normalized input feature map; m represents the size of the mini-batch in batch normalization; that is, the number of elements in the input feature map; and ε represents a small constant in batch normalization used to prevent division by zero.

The second and third convolution operations are performed using 3 × 3 and 1 × 1 convolution 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:

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

where M^l denotes the size of the l pooling layer and $down(\cdot)$ denotes the downsampling function.

The maximum pooling layer is a common down-sampling technique in convolutional neural networks, which serves to reduce the feature dimension and improve the computational efficiency and generalization ability of the model. The specific roles of the maximum pooling layer are as follows: feature compression: the maximum pooling layer can reduce the size of the feature map to reduce the computation of the model, and it can retain the main features of the original data to avoid overfitting. Feature invariance: The maximum pooling layer can improve the feature invariance, i.e., the features can remain unchanged when the input changes slightly, thus improving the model's robustness and generalization ability. Feature selection: The maximum pooling layer can select the most important features in the data, i.e., keep the maximum value and ignore other values, thus improving the expressiveness and performance of the model. In summary, the maximum pooling layer can optimize the performance and efficiency of the convolutional neural network by compressing the feature map, improving the feature invariance, and selecting the most important features. The specific computational procedure is as follows:

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 follows:

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

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

The input vector 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 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 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.

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.

5. 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 section introduces the CBAM module [30] 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 Figures 4 and 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:

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

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

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.

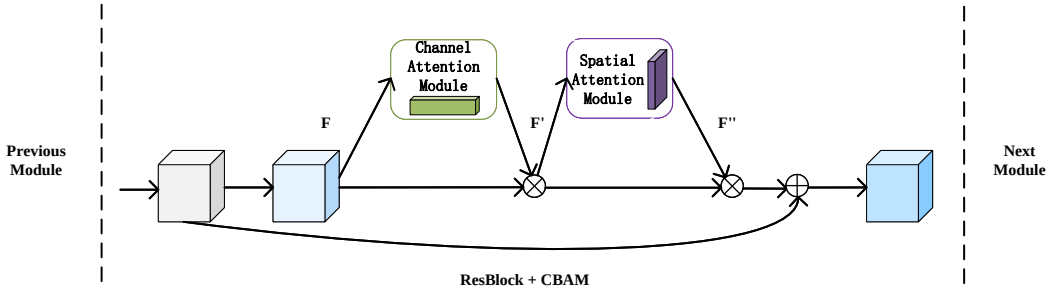


Figure 3. CBAM module diagram.

Since each channel of the feature map extracts some level of feature information, the channel attention mechanism feature information is more important. In order to compute channel attention efficiently, the channel attention module uses a spatial dimensionality method that compresses the input feature mapping, compared to using a single pooling method the channel attention module uses average pooling and maximum pooling methods and proves that the dual pooling method has stronger representational power. The specific computational procedure is shown as follows:

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

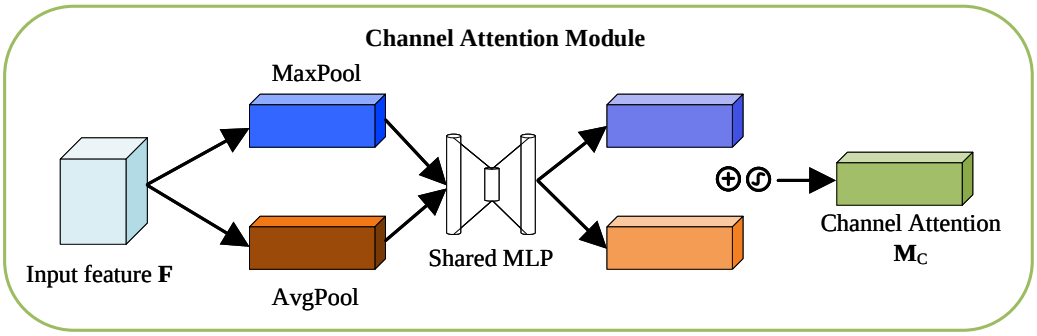


Figure 4. Channel attention module diagram.

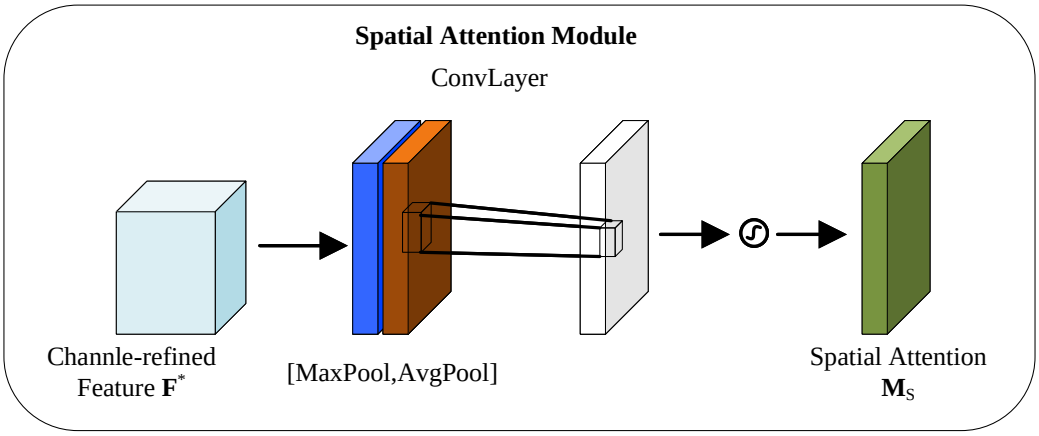


Figure 5. Spatial attention module diagram.

The specific structure of the spatial attention module is shown in Figure 5. The feature maps processed by the channel attention module first pass through two convolution layers to obtain the weight coefficients of each pixel point. Here, the weight coefficients are calculated by considering the response of each pixel point on different spatial scales, and the similarity between pixel points. Finally, the weight coefficients are applied to the feature map to emphasize the important pixel points and suppress the unimportant ones. This improves the generalization ability of the model. The spatial attention module focuses on which location information is meaningful, and it also complements the channel attention module, as shown below:

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

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:

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

where X is the input of the fully connected layer, w_j is the weight of the j-th class of features, and b_j is the bias term. The feedforward layer represents the captured spatiotemporal features as a linear representation suitable for predicting the final category labels using Softmax operations. In this process, each category is assigned a probability score, and

the category with the highest probability is considered 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):

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

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

To maximize feature reuse capability, this study uses jump connections to add the output of the previous parameter layer to its subsequent parameter layers to maintain local originality throughout the learning phase. In addition, each pair of jump-connected residual and attention modules will be connected to all subsequent pairs of residual and attention modules to maintain global originality throughout the learning phase. The proposed network structure has several significant advantages to improve its generalization performance.

6. Model Evaluation and Discussions

6.1. Dataset Description

Generalization ability is an important metric to assess the performance of a model. A good model should have strong generalization ability, meaning it can perform well on new datasets. Therefore, in this study, the model's generalization performance is tested using different datasets.

To evaluate the performance of the proposed model, we used ToN_IoT [13], a real dataset created from a large-scale IoT system developed by the Cyber IoT Laboratory at ADFA, New South Wales, Australia. This dataset contains normal category data and nine categories of attack data, including password, scan, ransomware, backdoor, denial of service, distributed denial of service, MITM, injection, and XSS attacks. The total number of data instances in this dataset is shown in Table 1. We trained our model using the ToN_IoT dataset to ensure it learns from a comprehensive set of attacks and normal behavior data.

Table 1. ToN_IoT dataset record description.

Record Type	Number of Records
Backdoor	35,000
DDoS	25,000
DoS	20,000
Injection	35,000
MITM	1043
Password	35,000
Ransomware	16,030
Scanning	3973
XSS	6116
Normal	245,000

To test the generalization ability of our model, we then used the UNSW-NB15 [31] dataset. The raw network packets for the UNSW-NB15 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. The basic information of this dataset is shown in Table 2.

Table 2. UNSW-NB15 dataset record description.

Record Type	Number of Records
Analysis	2000
Backdoor	1746
DoS	12,264
Exploits	33,393
Fuzzers	18,184
Generic	40,000
Reconnaissance	10,491
Shellcode	1133
Worms	130
Normal	56,000

We use several metrics to evaluate the performance of the proposed model, i.e., *accuracy*, *precision*, *recall*, and *F*1-*score* because they are widely used to evaluate deep learning algorithms. The formula is as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{17}$$

$$Precision = \frac{TP}{TP + FP} \tag{18}$$

$$Recall = \frac{TP}{TP + FN} \tag{19}$$

$$F1\text{-}score = \frac{2TP}{2TP + FP + FN} \tag{20}$$

where TP is true positive, representing the number of positive samples correctly identified as positive, FP is false positive, representing the number of negative samples incorrectly identified as positive, TN is true negative, representing the number of negative samples correctly detected as negative, and FN is false negative, representing the number of positive samples incorrectly identified as negative.

6.2. Hyperparameter Settings

Hyperparameters are parameters that are artificially adjusted before or during training. To ensure the best performance of the proposed deep learning model, optimal hyperparameters are determined through extensive experimentation. The hyperparameters involved in the experimental process include learning rate, epoch, and batch size. The details are shown in Table 3.

- Learning rate (LR): A critical hyperparameter in deep learning that regulates the network model's learning progress. In this paper, we set three learning rate values to obtain the best performance.
- Batch size: This parameter indicates the number of samples selected for one training. The batch size affects the memory usage, as well as the optimization and speed of the model. The batch size cannot be set too large or too small, so we set the range of batch size to 16, 32, 64, 128, 256, 512, 1024.
- Epoch: The epoch represents the process of training all the training samples once. Too many epochs can lead to overfitting, while too few epochs may result in suboptimal training parameters. In this study, the number of epochs is set to 100.

We divided the ToN_IoT dataset into a training set and a test set by the ratio of 80% and 20%, respectively. Initially, we set the learning rate to 0.01, epochs fixed to 100 times, and batch size range set to 16, 32, 64, 128, 256, 512, and 1024. We conducted experiments at different batch sizes and recorded the results for the above learning rates in Table 3. A detailed performance comparison of these parameters is shown in Figure 6a. The best-

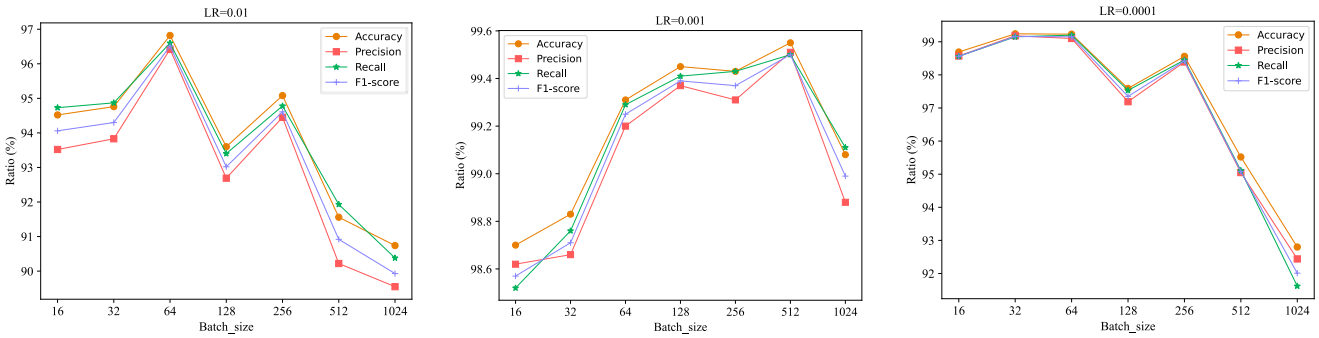
tested accuracy of the model is 96.82% for a learning rate of 0.01 and a batch size of 64. The lowest value of accuracy for this learning rate is 90.74% for a batch size of 1024. For other batch sizes, the accuracy and other performance scores are above 90%, demonstrating the model's robustness to changes in batch size.

Table 3. 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

In the second stage, we fixed the learning rate at 0.001 and the epoch remained fixed at 100 times, while the batch sizes ranged from 16, 32, 64, 128, 256, 512, and 1024. By keeping the learning rate and epoch fixed, we conducted experiments at different batch sizes and recorded the results in Table 3. A detailed performance comparison of these parameters is shown in Figure 6b. The best test accuracy is 99.55% when the batch size is 512. At batch sizes of 64, 128, 256, and 1024, the accuracy of the model was greater than 99%. At lower batch sizes of 16 and 32, the accuracy and other performance scores decreased but were still close to 99%. This indicates that a learning rate of 0.001 with a batch size of 512 provides the optimal balance between training stability and performance.

Finally, we set the learning rate to 0.0001, and the other parameters were kept the same as those in the first two stages. The detailed performance comparison of these parameters is shown in Figure 6c. The best test accuracy is 99.24% when the batch size is 32. For batch sizes of 16, 64, and 256, the accuracy of the model is above 98%. For other batch sizes, the accuracy and other performance scores decreased but the results were above 90%. The comparative experimental results of the above performance parameters show that the proposed model achieves the best results when the learning rate is 0.001, epochs are 100, and the batch size is 512. This combination of parameters consistently provides high accuracy and stable performance across different configurations, demonstrating the robustness and generalizability of the proposed model.



(**a**) LR = 0.01 performance. (**b**) LR = 0.001 performance. (**c**) LR = 0.0001 performance.

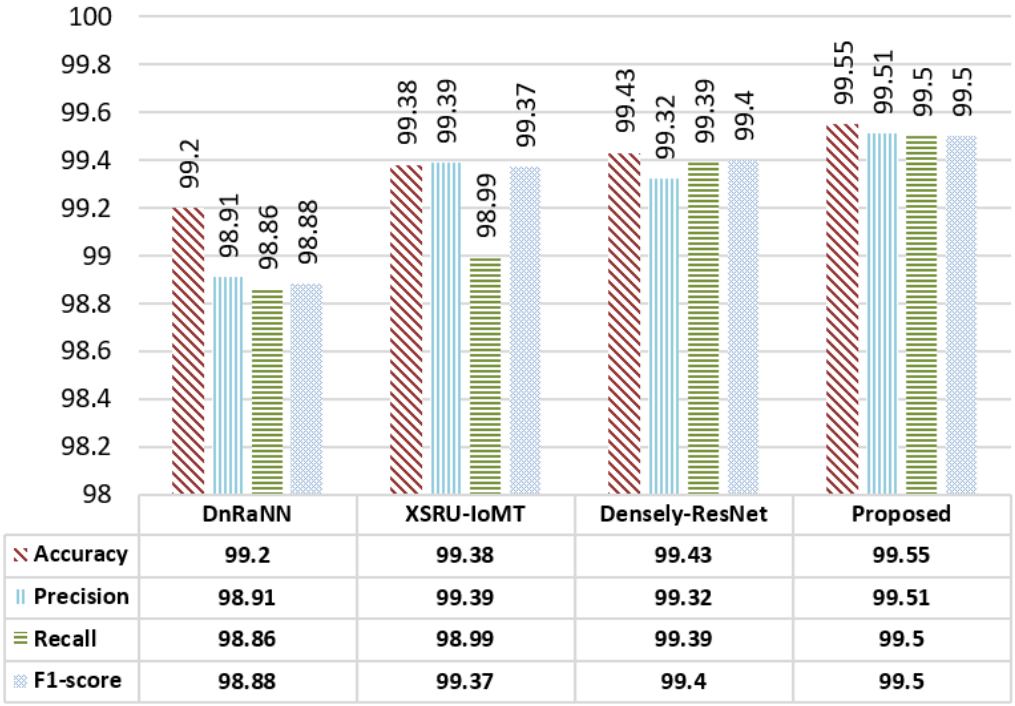
Figure 6. Performance at different learning rates.

Finally, we set the learning rate to 0.0001, and the other parameters were kept the same as those in the first two stages. The detailed performance comparison of these parameters is shown in Figure 6c. The best test accuracy is 99.24% when the batch size is 32. For batch sizes of 16, 64, and 256, the accuracy of the model is above 98%. For other batch sizes, the accuracy and other performance scores decreased, but the results were above 90%.

The comparative experimental results of the above performance parameters show that the proposed model achieves the best results when the learning rate is 0.001, epochs are 100, and batch size is 512.

6.3. Comparison with State-of-the-Art Methods

To further analyze the effectiveness of the proposed model, we compared the proposed model with some of the best current methods and evaluated the performance differences between the proposed model and other methods. The result is shown in Figure 7. As can be seen from the figure, the proposed model has the best performance on the ToN_IoT dataset, outperforming existing models in terms of accuracy, precision, recall, and F1-score. Specifically, the proposed model achieved an increase in accuracy from 99.20% with DnRaNN to 99.55%, an increase of 0.35%. Compared to XSRU-IoMT, the accuracy increased from 99.38% to 99.55%, which is an increase of 0.17%. Additionally, the accuracy improved from 99.43% with Densely-ResNet to 99.55%, an increase of 0.12%. These results verify the validity of our proposed model.



	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

Figure 7. Performance comparison with other models.

In addition, we compared the generalization ability of the model with several deep learning methods on the UNSW-NB15 dataset, as shown in Table 4. Our proposed LSTM-ResNet model achieved accuracy, precision, recall, and F1-score values of 89.23%, 88.83%, 87.77%, and 88.25%, respectively. This represents a significant improvement in performance metrics compared to other models. For instance, the accuracy of our model is 15.3% higher than that of the Densely-ResNet model, which is the second-best performing model in our comparison.

The severe class imbalance problem in the UNSW-NB15 benchmark often leads to poor model generalization performance. However, our proposed model demonstrates robust performance despite this challenge. As shown in Table 4, the LSTM-ResNet model

not only achieves the highest accuracy but also excels in precision, recall, and F1-score compared to LSTM, LuNet, Densely-ResNet, and CNN models. This indicates that our model has superior generalization capability and can effectively handle imbalanced data. These results highlight the robustness and reliability of our method in diverse scenarios, confirming its effectiveness in real-world applications.

Moreover, the attention mechanism incorporated in our model helps it focus on important features, further enhancing its performance and robustness. This makes the LSTM-ResNet model particularly effective in identifying and classifying network intrusions, as evidenced by its leading performance across multiple evaluation metrics.

Table 4. Comparison of model generalization performance on the UNSW-NB15 dataset.

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**

7. Conclusions

In this paper, we propose 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, which is 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 achieves state-of-the-art detection accuracy on the UNSW-NB15 benchmarks while maintaining a low false positive rate. The evaluation results confirm the effectiveness and stability of the proposed model, and it is recommended for use in other intrusion detection tasks in the future.

Author Contributions: Conceptualization, B.C. and Y.C.; methodology, B.C. and Y.C.; investigation, B.C. and Y.C.; writing—original draft preparation, B.C., Y.C., K.L. and Z.Y.; writing—review and editing, B.C., Y.C., K.L. and Z.Y.; visualization, Y.C. All authors have read and agreed to the published version of the manuscript.

Funding: This paper is supported by the National Natural Science Foundation of China (61962042), the Natural Science Foundation of Inner Mongolia (2022MS06020), the Central Government Guides Local Science and Technology Development Fund (2022ZY0064), the University Youth Science and Technology Talent Development Project (Innovation Group Development Plan) of Inner Mongolia A. R. of China (grant no. NMGIRT2318), and the Fund for Supporting the Reform and Development of Local Universities (Disciplinary Construction).

Data Availability Statement: The datasets used in this study are publicly available at the following links: https://research.unsw.edu.au/projects/unsw-nb15-dataset and https://research.unsw.edu.au/projects/toniot-datasets, accessed on 1 May 2024.

Acknowledgments: The authors would like to extend their sincere appreciation to the reviewers for their invaluable feedback and insightful suggestions.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Bertino, E.; Islam, N. Botnets and internet of things security. *Computer* **2017**, *50*, 76–79. [CrossRef]
2. Kolias, C.; Kambourakis, G.; Stavrou, A.; Voas, J. DDoS in the IoT: Mirai and other Botnets. *Computer* **2017**, *50*, 80–84. [CrossRef]
3. Abomhara, M.; Koien, G.M. Cyber security and the internet of things: Vulnerabilities, threats, intruders and attacks. *J. Cyber Secur. Mobil.* **2015**, *4*, 65–88. [CrossRef]

4. Thakkar, A.; Lohiya, R. A review on machine learning and deep learning perspectives of IDS for IoT: Recent updates, security issues, and challenges. *Arch. Comput. Methods Eng.* **2021**, *28*, 3211–3243. [CrossRef]
5. Al-Garadi, M.A.; Mohamed, A.; Al-Ali, A.K.; Du, X.; Ali, I.; Guizani, M. A survey of machine and deep learning methods for internet of things (IoT) security. *IEEE Commun. Surv. Tutorials* **2020**, *22*, 1646–1685. [CrossRef]
6. Babu, M.R.; Veena, K.N. A survey on attack detection methods for IoT using machine learning and deep learning. In Proceedings of the 2021 3rd International Conference on Signal Processing and Communication (ICPSC), Coimbatore, India, 13–14 May 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 625–630.
7. Denning, D.E. An intrusion-detection model. *IEEE Trans. Softw. Eng.* **1987**, *SE-13*, 222–232. [CrossRef]
8. Chaabouni, N.; Mosbah, M.; Zemmari, A.; Sauvignac, C.; Faruki, P. Network intrusion detection for IoT security based on learning techniques. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 2671–2701. [CrossRef]
9. Alsoufi, M.A.; Razak, S.; Siraj, M.M.; Nafea, I.; Ghaleb, F.A.; Saeed, F.; Nasser, M. Anomaly-based intrusion detection systems in IoT using deep learning: A systematic literature review. *Appl. Sci.* **2021**, *11*, 8383. [CrossRef]
10. Selvapandian, D.; Santhosh, R. Deep learning approach for intrusion detection in IoT-multi cloud environment. *Autom. Softw. Eng.* **2021**, *28*, 19. [CrossRef]
11. Khoa, T.V.; Saputra, Y.M.; Hoang, D.T.; Trung, N.L.; Nguyen, D.; Ha, N.V.; Dutkiewicz, E. Collaborative learning model for cyberattack detection systems in IoT industry 4.0. In Proceedings of the 2020 IEEE Wireless Communications and Networking Conference (WCNC), Seoul, Republic of Korea, 25–28 May 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–6.
12. Haider, A.; Adnan Khan, M.; Rehman, A.; Rahman, M.; Kim, S.H. A real-time sequential deep extreme learning machine cybersecurity intrusion detection system. *Comput. Mater. Contin.* **2021**, *66*, 1785–1798. [CrossRef]
13. Booij, T.M.; Chiscop, I.; Meeuwissen, E.; Moustafa, N.; Den Hartog, F.T. ToN_IoT: The role of heterogeneity and the need for standardization of features and attack types in IoT network intrusion data sets. *IEEE Internet Things J.* **2021**, *9*, 485–496. [CrossRef]
14. Tsimenidis, S.; Lagkas, T.; Rantos, K. Deep learning in IoT intrusion detection. *J. Netw. Syst. Manag.* **2022**, *30*, 8. [CrossRef]
15. Wu, P.; Moustafa, N.; Yang, S.; Guo, H. Densely connected residual network for attack recognition. In Proceedings of the 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), Guangzhou, China, 29 December 2020–1 January 2021; IEEE: Piscataway, NJ, USA, 2020; pp. 233–242.
16. Hasan, M.; Islam, M.M.; Zarif, M.I.I.; Hashem, M.M.A. Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches. *Internet Things* **2019**, *7*, 100059. [CrossRef]
17. Ravi, N.; Shalinie, S.M. Learning-driven detection and mitigation of DDoS attack in IoT via SDN-cloud architecture. *IEEE Internet Things J.* **2020**, *7*, 3559–3570. [CrossRef]
18. Zhang, B.; Liu, Z.; Jia, Y.; Ren, J.; Zhao, X. Network intrusion detection method based on PCA and Bayes algorithm. *Secur. Commun. Netw.* **2018**, *2018*, 1914980. [CrossRef]
19. Abdel-Basset, M.; Chang, V.; Hawash, H.; Chakrabortty, R.K.; Ryan, M. Deep-IFS: Intrusion detection approach for industrial internet of things traffic in fog environment. *IEEE Trans. Ind. Inform.* **2020**, *17*, 7704–7715. [CrossRef]
20. Liu, Y.; Garg, S.; Nie, J.; Zhang, Y.; Xiong, Z.; Kang, J.; Hossain, M.S. Deep anomaly detection for time-series data in industrial IoT: A communication-efficient on-device federated learning approach. *IEEE Internet Things J.* **2020**, *8*, 6348–6358. [CrossRef]
21. Li, L.; Yan, J.; Wang, H.; Jin, Y. Anomaly detection of time series with smoothness-inducing sequential variational auto-encoder. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *32*, 1177–1191. [CrossRef] [PubMed]
22. Gao, J.; Gan, L.; Buschendorf, F.; Zhang, L.; Liu, H.; Li, P.; Dong, X.; Lu, T. Omni SCADA intrusion detection using deep learning algorithms. *IEEE Internet Things J.* **2020**, *8*, 951–961. [CrossRef]
23. Parra, G.D.L.T.; Rad, P.; Choo, K.K.R.; Beebe, N. Detecting internet of things attacks using distributed deep learning. *J. Netw. Comput. Appl.* **2020**, *163*, 102662.
24. Khan, I.A.; Moustafa, N.; Razzak, I.; Tanveer, M.; Pi, D.; Pan, Y.; Ali, B.S. XSRU-IoMT: Explainable simple recurrent units for threat detection in internet of medical things networks. *Future Gener. Comput. Syst.* **2020**, *127*, 181–193. [CrossRef]
25. Wu, P.; Guo, H. LuNET: A deep neural network for network intrusion detection. In Proceedings of the 2019 IEEE Symposium Series on Computational Intelligence (SSCI), Xiamen, China, 6–9 December 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 617–624.
26. Latif, S.; e Huma, Z.; Jamal, S.S.; Ahmed, F.; Ahmad, J.; Zahid, A.; Dashtipour, K.; Aftab, M.U.; Ahmad, M.; Abbasi, Q.H. Intrusion detection framework for the internet of things using a dense random neural network. *IEEE Trans. Ind. Inform.* **2021**, *18*, 6435–6444. [CrossRef]
27. Ferrag, M.A.; Maglaras, L.; Moschoyiannis, S.; Janicke, H. Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. *J. Inf. Secur. Appl.* **2020**, *50*, 102419. [CrossRef]
28. Guo, M.H.; Xu, T.; Liu, J.J.; Liu, Z.N.; Jiang, P.T.; Mu, T.J.; Zhang, S.H.; Martin, R.; Cheng, M.M.; Hu, S.M. Attention mechanisms in computer vision: A survey. *Comput. Vis. Media* **2022**, *8*, 331–368. [CrossRef]
29. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 770–778.

30. Woo, S.; Park, J.; Lee, J.Y.; Kweon, I.S. CBAM: Convolutional block attention module. In Proceedings of the 2018 European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; Springer: Berlin/Heidelberg, Germany, 2018; pp. 3–19.
31. Moustafa, N.; Slay, J. UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In Proceedings of the 2015 Military Communications and Information Systems Conference (MilCIS), Canberra, Australia, 10–12 November 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 1–6.