Hindawi
Security and Communication Networks
Volume 2021, Article ID 5593435, 9 pages
https://doi.org/10.1155/2021/5593435

Research Article

A Residual Learning-Based Network Intrusion Detection System

Jiarui Man[1,2] **and Guozi Sun**[1,2]

[1]*School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing 210023, China*
[2]*Key Laboratory of Urban Land Resources Monitoring and Simulation, MNR, Shenzhen 518000, China*

Correspondence should be addressed to Guozi Sun; sun@njupt.edu.cn

Received 9 January 2021; Revised 13 February 2021; Accepted 5 March 2021; Published 16 March 2021

Academic Editor: Weizhi Meng

Neural networks have been proved to perform well in network intrusion detection. In order to acquire better features of network traffic, more learning layers are necessarily required. However, according to the results of the previous research, adding layers to the neural networks might fail to improve the classification results. In fact, after the number of layers has reached a certain threshold, performance of the model tends to degrade. In this paper, we propose a network intrusion detection model based on residual learning. After transforming the UNSW-NB15 data set into images, deeper convolutional neural networks with residual blocks are built to learn more critical features. Instead of the cross-entropy loss function, the modified focal loss is calculated to address the class imbalance problem in the training set and identify minor attacks in the testing set. Batch normalization and global average pooling are used to avoid overfitting and enhance the model. Experimental results show that the proposed model can improve attack detection accuracy compared with existing models.

1. Introduction

With the continuing expanding network scale, network security confronts more sophisticated threats than ever before. Hence, network security issues are attracting increasing attention. Commonly used network security systems that discover suspicious attacks involve firewalls, intrusion detection systems (IDSs), and intrusion prevention systems (IPSs) [1]. Among them, the task of IDSs is to collect and identify abnormal behaviors in the network [2]. By analyzing captured data packets, IDSs can check legitimate network behaviors, detect the attacks, and report the attacks for further containment.

Conventional IDSs tend to get low detection rates and high false positive rates due to their reliance on patterns of known attacks. Researchers have applied artificial intelligence (AI) algorithms in the designing part of IDSs to provide better performances. The performance of an IDS is closely related to the selected classifier, while traditional machine learning algorithms tend to perform poorly in the scenarios when large amounts of network data packets are included. In recent years, deep learning has achieved outstanding results in multiple fields. The advantage of deep learning is that it can learn the hierarchical features from a large amount of data to improve model efficiency [3]. The application of deep learning can reduce costs of IDSs and strengthen the abilities to identify attacks.

Convolutional neural networks (CNNs) can extract deep and critical features from the given data. It is a general perspective that increasing the number of network layers can help learn better features; hence, the performance of model is improved. However, simply stacking more layers may fail these tasks. Furthermore, after the number of layers has reached a certain threshold, it may even lead to performance degradation. Residual learning is proposed to address the issue above. Residual is the error between the actual value and the estimated value, and residual learning is originally derived from the residual representation in image recognition [4]. Residual learning is realized by establishing a direct connection between the input and the output. CNNs based on residual learning have achieved outstanding results in image recognition [5]. They are easier to train and optimize than common CNNs. In network intrusion detection, it is also vital to build deeper networks to improve the

detection capabilities of IDSs. Because residual learning allows CNNs to be deeper, this paper introduced the concept of residual learning into IDSs.

UNSW-NB15, an imbalanced network intrusion detection data set, is selected to evaluate the model. In real-time network intrusion detection, the class imbalance problem seriously affects the classification results [6]. Prediction models that predict only the dominant classes fail to identify the minor classes. Resampling techniques are common solutions to class imbalance problems. However, resampling techniques have their disadvantages. Oversampling might disrupt the original data, and it takes more time to train the model when using oversampling techniques. Undersampling might cause the loss of vital information, affecting the classification capabilities. Focal loss was originally proposed to balance the loss between samples. We apply the modified focal loss function in the proposed model to enhance the abilities to detect minor classes without disrupting the training data.

The major contributions of the paper can be summarized as follows:

(1) Propose a deep learning-based intrusion detection model with a higher accuracy compared with other existing models

(2) Introduce residual learning into the model to address the network degradation problem, allowing the model to learn deeper features

(3) Use a modified focal loss function to deal with the class imbalance problem in the training set

This paper is organized as follows:

(1) The first chapter gives a brief overview of network intrusion detection and the motivation of the proposed methodology

(2) The second chapter introduces the related work

(3) The third chapter provides the methodology and implementation process in detail

(4) The fourth chapter carries out the experiments and analyzes the testing results

(5) The final chapter concludes the paper

2. Related Work

Data preprocessing is a key step in network intrusion detection. It can extract key features that have great influences on the classification results, effectively reducing the size of data and improving the efficiency of given classifiers. Zhang et al. [7] proposed an effective network traffic classification method, which used principal component analysis (PCA) to remove the irrelevant features and applied Gaussian Naive Bayes as the classifier. Kasongo et al. [8] applied a filter-based feature reduction technique on the UNSW-NB15 data set using the XGBoost algorithm and then implemented several algorithms to classify the data. Results demonstrated that the feature selection method increased the test accuracy. Sun et al. [9] proposed an improved Naive Bayesian learning method which took the influence of different features into account. It achieved a higher accuracy than traditional machine learning algorithms. It can be seen that the performance of traditional classifiers is excessively dependent on the extracted features. However, traditional machine learning algorithms are shallow learning algorithms which require feature engineering. To build the fittest model, optimization of parameters is also needed. The size of the data set also affects the efficiency of the models. These difficulties slow down the training process of traditional machine learning algorithms and affect the overall network security.

In recent years, deep learning models have been gradually applied to intrusion detection to enhance the classification classifiers due to their high efficiency and easy implementation. Among deep learning models, CNNs have made great success in many fields [10–12], and researchers have applied CNNs in intrusion detection. Qian et al. [13] analyzed the network traffic with a CNN. In the training phase, rectified linear unit (ReLU) served as the activation function and adaptive moment estimation (Adam) algorithm was used to optimize the model. Lai et al. [14] also used a CNN as the intrusion detection model, achieving a higher accuracy rate than other deep learning models.

In the aspect of residual learning, the concept of Residual Network (ResNet) was proposed by He et al. [15] from Microsoft Research Institute to deal with the performance degradation problem as the number of layers grows. ResNets have outperformed common CNNs in image classification and object recognition [16–18]. Because of residual learning, the depth of ResNets is deeper than that of the traditional CNNs. In network intrusion detection, a deeper CNN can extract more critical features and get better classification results. Therefore, CNNs based on residual learning have been attempted in network intrusion detection. Wu et al. [19] proposed a deep neural network built upon residual blocks to discover malicious network behaviors, achieving a low false alarm rate. Chouhan et al. [20] proposed a multipath residual learning-based CNN architecture that was being evaluated on NSL-KDD data set, showing significant improvements over the previous research.

However, while residual learning can improve the overall performance of CNNs, in the practical aspect, it does not improve models' abilities to detect minor attacks due to lack of original training data. Classes in most modern network intrusion detection data sets are imbalanced. Therefore, most IDSs fail to provide better performances for attacks with fewer samples. Focal loss was proposed by He et al. [21] in 2017. Focal loss takes the different level of training difficulty of samples into consideration and focuses more on the difficult-to-train samples; therefore, it has been applied in many fields, such as object detection, imbalanced data classification, and so on [22–24]. To identify classes with fewer training samples more accurately, a modified focal loss function is used to replace cross-entropy loss function in the proposed model.

Choosing a suitable data set is vital for the building of IDSs. In recent years, most commonly used public data sets in network intrusion detection are KDD99 [25], NSL-KDD [26], and UNSW-NB15 [27]. In spite of being the most

popular data sets in network intrusion detection, KDD99 and NSL-KDD are out-of-date due to old and redundant data. Evaluating IDSs using KDD99 and NSL-KDD does not reflect satisfactory results due to their shortcomings. According to the previous research and statistical analysis, compared with the other two data sets, UNSW-NB15 has more complex feature sets, contains more modern normal traffic scenarios, covers richer types of attack traffic, and contains fewer incomplete samples. Also, most new cyberattacks are variants of these known attacks in the UNSW-NB15 data set. Therefore, UNSW-NB15 can more accurately reflect the characteristics of modern network traffic data and is more suitable for evaluating IDSs. Therefore, we choose UNSW-NB15 data set as the evaluation set for the proposed model.

In summary, with the powerful capabilities of automatic feature extraction, deep learning has been applied to network intrusion detection. However, how to build deeper networks without triggering the performance degradation problem and address the class imbalance problem in the training set are two major challenges. In this paper, a residual learning-based CNN is constructed to learn deeper features of network traffic, and the modified focal loss function is introduced into the proposed model to detect minor attacks.

3. Methodology

The proposed methodology consists of three parts: data preprocessing, model constructing, and model evaluation. First, network flows are converted into images. Then, CNNs with residual learning are constructed. Finally, trained models are tested and evaluated. The main structure is shown in Figure 1.

3.1. Data Set. As stated before, UNSW-NB15 is a network intrusion detection data set, which is processed and built through collecting different types of network connection data. This data set includes multiple types of contemporary attacks. Each flow of the data set contains 47 features, and the data set divides the network behaviours into nine categories of attacks plus the category of normal behaviours. These attacks can also be divided into 177 categories according to the environments that the specific attack depends on.

In this paper, part of the data set known as the UNSW-NB15 training set and UNSW-NB15 testing set are selected as the training data and testing data. They are data sets which are used for intrusion detection after redundant flows and features are processed. The distribution is shown in Table 1.

3.2. Data Preprocessing. Features in UNSW-NB15 contain numeric features and symbolic features; therefore, symbolic features should be digitized first. Then, processed features are normalized to obtain a standardized data set and converted into matrices.

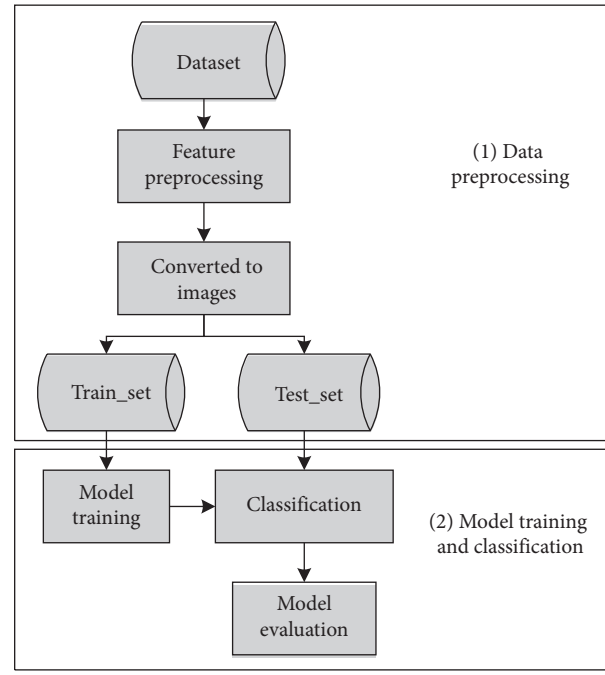

Figure 1: Model structure.

Table 1: Distribution of UNSW-NB15 training set and testing.

Type	Train_set	Test_set	Label
Analysis	2000	677	0
Backdoors	1746	583	1
DoS	12264	4089	2
Exploits	33393	11132	3
Fuzzers	18185	6062	4
Generic	40000	18871	5
Normal	56000	37000	6
Recon	10492	3496	7
Shellcode	1133	378	8
Worms	130	44	9

One-hot encoding is used to map symbolic features into numerical features, and labels are mapped into digits using label encoding. The specific implementations are as follows:

(1) *One-Hot Encoding*. One-hot encoding mainly uses a state register of size X to encode a character, and each character will have an independent register bit

(2) *Label Encoding*. Labels in the UNSW-NB15 data set are divided into 10 categories. Coding rules are shown in Table 1

This paper uses Min-Max normalization. The main function of Min-Max normalization is to unify the feature values in the interval of [0,1]:

$$x^* = \frac{x - x_{\min}}{x_{\max} - x_{\min}}, \tag{1}$$

where x^* is the normalized eigenvalue, x is the original eigenvalue, $x_{\min}$ is the minimum eigenvalue, and $x_{\max}$ is the maximum eigenvalue. After numerical normalization, each flow of the new set contains 196 features, so the data are

converted into 14 ∗ 14 matrices; then, the matrices are changed into black and white images.

3.3. Network Construction. Figure 2 shows the overall structure and the parameters of the CNN model. The proposed model extracts the features of input data by the convolution layers and pooling layers. Feature maps are then input into a global average pooling layer. Finally, the model classifies the sample data with a softmax layer.

3.3.1. Convolution and Pooling. Convolution layers are the core parts of CNN models. Convolution layers in proposed model extracted spatial features of given data and produced a feature map as the output. ReLU is often used as the activation function:

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

and the function of pooling layers is to reduce the size of feature maps.

3.3.2. Batch Normalization. In the training part of CNNs, with the change of the parameters of the previous layer, the input distribution of the next layer will change correspondingly, making it more difficult to train deeper neural networks. Batch normalization, in the training process of CNNs, makes the input of each layer maintain the same distribution and provides with the solution to the difficulty of network training, thus effectively improving the training speed of networks and avoiding overfitting. Input data are all divided into batches, for instance, parameter "batch_size" is set to 128; therefore, 128 pieces of data are input as a batch at a time. Batch normalization layers are to normalize each batch so that the distribution of data remains unchanged. Suppose we have a batch of inputs:

$$x = \{x_1, x_2, \ldots, x_n\}. \tag{3}$$

The output of batch normalization is computed by

$$y_i = \lambda * x_i' + \varphi, \tag{4}$$

where λ and φ are learned parameters and x_i' is calculated through

$$\mu_\beta = \frac{1}{m}\sum_{i=1}^{m} x_i;$$

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

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

In this paper, batch normalization layers are placed after convolution layers and before the activation functions.

Layer	Parameters
Convolution	3 ∗ 3 ∗ 64, stride 1
Maxpooling	3 ∗ 3, stride 2
Residual block_1	Conv 1 ∗ 1 ∗ 64, stride 1 Conv 3 ∗ 3 ∗ 64, stride 1 Conv 1 ∗ 1 ∗ 256, stride 1
Residual block_2	Conv 1 ∗ 1 ∗ 128, stride 1 Conv 3 ∗ 3 ∗ 128, stride 1 Conv 1 ∗ 1 ∗ 512, stride 1
Residual block_3	Conv 1 ∗ 1 ∗ 256, stride 1 Conv 3 ∗ 3 ∗ 256, stride 1 Conv 1 ∗ 1 ∗ 1024, stride 1

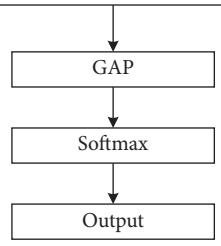

Figure 2: CNN model.

3.3.3. Residual Learning. Compared with common CNNs such as the LeNet-5 [28] and the AlexNet [29], ResNets introduced residual learning into the constructing of CNNs. The depth of a CNN has a great influence on the final classification results, so deeper networks are often constructed. However, as the network depth increases, the phenomenon of gradient explosion might occur, and the performance of the network will degrade. According to the previous experimental results, simply adding convolution layers and pooling layers to the network does not improve the accuracy of the network but leads to the deterioration of network performance. In this paper, residual learning is used to address the issue above. Residual refers to the residual difference between the local input and output:

$$f(x) = g(x) - x. \tag{6}$$

In contrast to identity mapping, the learning goal of residual learning is 0, that is, to reduce the difference between the input and the output, allowing the original input to be directly connected to one certain network layer, so that the network can learn the residual. Residual learning is realized by a fast shortcut connection between the input and output of a block. It not only avoids adding additional parameters and computations to the network, but also effectively trains the parameters in the network and guarantees the performance while the network can learn deeper features. Two blocks used in this paper are shown in Figure 3. In the construction of plain models, the normal block exhibited in (a) is used, while the residual block shown in (b) is used to construct residual networks.

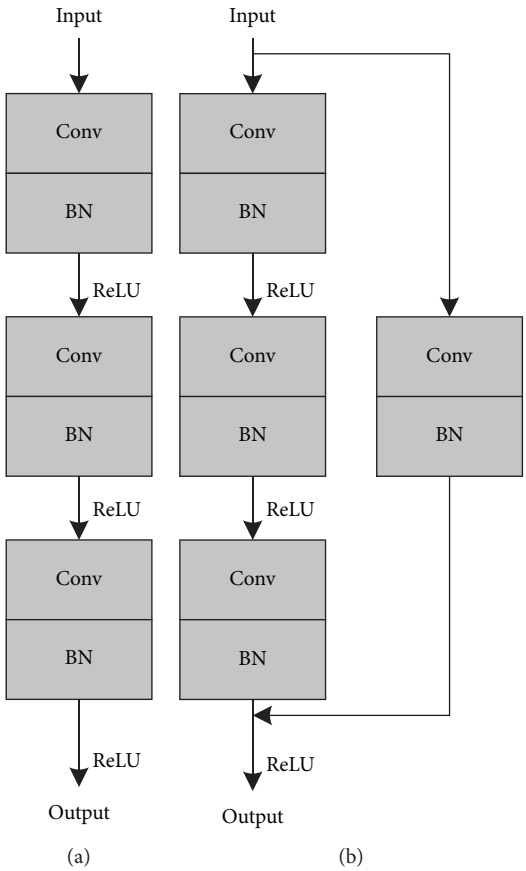

Figure 3: Normal block and residual block. (a) Normal block. (b) Residual block.

3.3.4. Global Average Pooling. At the end of CNN models, flatten layers are often adopted to flatten the data processed by the previous layers into a one-dimensional vector. The output size is gradually reduced through full connection layers, and the final output is obtained through an activation function. Since every node in flatten layers and full connection layers is connected to each other, too many parameters may lead to overfitting. A global average pooling layer is an average pooling layer without filter size. It averages the entire feature map. Using a global average pooling layer can reduce the count of calculating parameters and accordingly reduce the possibility of overfitting. In this paper, a global average pooling layer is used to replace the flatten layer. The principle of global average pooling is shown in Figure 4.

3.3.5. Softmax and Loss Function. Finally, the probability distribution of each label is calculated through the softmax layer:

$$S_j = \frac{e^{a_j}}{\sum_{k=1}^{N} e^{a_k}}, \tag{7}$$

where N denotes the total count of classes.a_j denotes the j_{th} input of softmax layer. Cross-entropy loss function is defined as

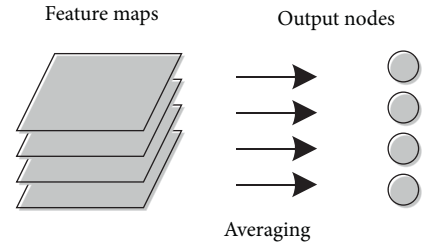

Figure 4: Global average pooling.

$$L = -\sum_{k=1}^{N} y_k \log S_k, \tag{8}$$

where y_k denotes the probability that tested sample belongs to class k.

With the obvious class imbalance problem in the training set, preventing loss function from optimising one category while suppressing other categories is important. To increase the classification accuracy for minor classes, we need to make the model pay more attention to them during training. Resampling is one of the most common methods to deal with imbalanced data. Among resampling methods, undersampling may cause the loss of vital information while oversampling may add new information to disrupt the data and greatly increase training time. Compared with cross-entropy loss function, focal loss aims to solve the class imbalance problem so that if the number of samples that are easy to train is large, contribution of certain samples to the total loss is small. In other words, focal loss function focuses on minor samples. In our multilabel classification, focal loss is defined as

$$FL_{\text{loss}} = -a_t (1 - p_t)^{\gamma} \log (p_t), \tag{9}$$

where $(1 - p_t)^{\gamma}$ is a modulating factor that reduces loss contribution from easy samples.p_t was calculated through

$$p_t = \begin{cases} p, & y = 1 \\ 1 - p, & \text{otherwise}, \end{cases} \tag{10}$$

where $p \in [0, 1]$ represents the category prediction probability and y is the label value. As $p_t \longrightarrow 1$, $(1 - p_t)^{\gamma}$ goes to 0 and the weights of samples that are easy to train to the loss are reduced. And a_t is a weighting factor that can be used to scale the minor classes separately. In this paper, we introduce the multilabel focal loss where γ was set to 2 and a_t was calculated through

$$a_t = 1 - \frac{\text{num}_t}{\text{total\_cnt}}, \tag{11}$$

where num_t denotes the number of samples belonging to class t and total_cnt denotes the total number of samples in the training data.

4. Experiments

4.1. Experimental Environments. Experimental environments of this paper are shown in Table 2.

Table 2: Experimental environment.

Environment	Value
OS	Windows
CPU	i7-7700 HQ
Memory	DDR4 8 GB
Language	Python
SDE	Keras
Tool	Anaconda

4.2. Evaluation Metrics. Accuracy, precision, recall, and F1-measure are adopted as evaluation metrics.

(1) Accuracy (Acc): the ratio of the number of correctly classified samples to the total number of samples tested.

(2) Precision (P): the ratio of correctly classified positive samples to the total number of positive samples.

(3) Recall (R): the ratio of accurately identified positive samples to the total number of positive samples in the testing set.

(4) F1-measure (F1): the weighted average of precision and recall.

$$\begin{aligned} \text{accuracy} &= \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}; \\ \text{precision} &= \frac{\text{TP}}{\text{TP} + \text{FP}}; \\ \text{recall} &= \frac{\text{TP}}{\text{TP} + \text{FN}}; \\ F1 &= \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}}. \end{aligned} \tag{12}$$

True Positive (TP) denotes the number of positive samples correctly classified as positive; True Negative (TN) denotes the number of negative samples correctly classified as negative; False Positive (FP) denotes the number of negative samples misclassified as positive; and False Negative (FN) denotes the number of positive samples misclassified as negative. The confusion matrix is shown in Table 3.

4.3. Experimental Performance Evaluation. In the training phase, 6 CNN models are constructed, including 3 plain models and 3 residual models. Model P_n (R_n) consists of n normal blocks (residual blocks) shown in Section 3.3.3. Each model is trained by the processed training set for 100 epochs. The learning rate is set to 0.01. And after calculating the loss, an optimizer is needed to update the network weights. Adam is selected as the optimizer. Performances of 6 models are evaluated by calculating the model accuracy and the weighted average of precision, recall, and F1-measure. We choose weighted average to evaluate the overall performance, because compared with other average methods like the micro average and the macro average, the weighted average method takes the number of samples belonging to each class into consideration, so its results are more convincing to reflect the performance of the model. The weighted average is defined as

Table 3: Confusion matrix.

Actual class	Predicted class	
	Positive	Negative
Positive	TP	FN
Negative	FP	TN

$$W_M = \sum_{i=1}^{k} n_k M_k, \tag{13}$$

where k denotes the total amount of classes, n_k denotes the number of testing samples in class k, and M_k is the testing result of metric M on class k.

We record the training loss of the above 6 models every 20 epochs to examine the effects of residual learning on CNNs. It can be seen from Figure 5, by utilizing residual learning in the blocks, training loss is greatly reduced. By adding residual blocks in the CNN, we achieve lower training loss, indicating that residual learning can address the network degradation problem. Also, as the figure demonstrates, the training loss at the very beginning is quite large, but as the training process progresses, the loss value continues to decrease. When the training epoch reaches 20, the training loss tends to decrease at a slower rate.

According to the comparison results from Table 4, the overall performance of CNNs has been significantly improved with residual blocks added into the plain models, indicating that we can build deeper CNNs with residual learning. Results can also demonstrate that with the increasing number of network layers, residual networks can achieve better performances than shallow residual networks on the whole. The model with 3 residual blocks (R3) achieves the highest overall classification accuracy of 88.695%. R3 (RLF-CNN) will be further compared with the state-of-the-art classification algorithms.

In order to evaluate the abilities of proposed method to detect attacks like Shellcode and Worm in network intrusion detection, we conduct several experiments and compared the recall value of each class with Multilayer Perceptron (MLP) and Long Short-Term Memory (LSTM), LeNet-5, AlexNet, and CNN with simple cross-entropy loss function (RLC-CNN). Support Vector Machine (SVM) and Random Forest (RF) are commonly used machine learning algorithms in network intrusion detection [30, 31]. We select SVM and RF to compare their classification results with those of deep learning algorithms. The number of training epochs of deep learning models was also set to 100. Table 5 and Figure 6 demonstrate the recall values of all models on each class. Figure 6 also shows the overall accuracy of each model. It can be seen from Table 5 that the performance of deep neural networks is significantly better than the classic machine learning algorithms. Classic machine learning algorithms need manually designed features of network traffic before the training phase, while deep learning algorithms automatically extract features.

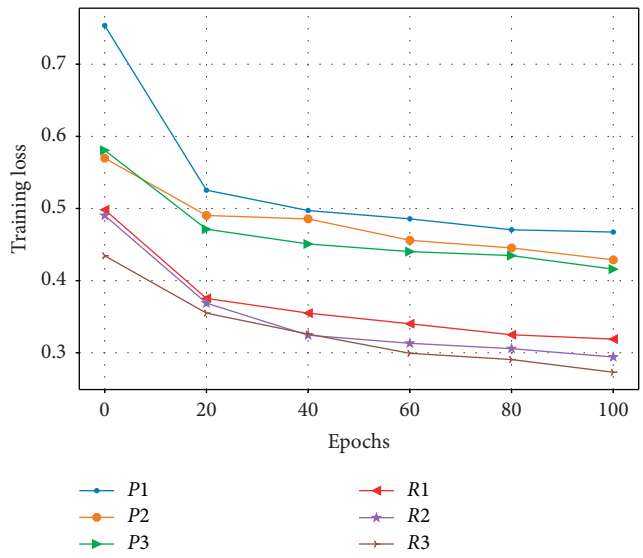

Figure 5: Training loss.

Table 4: Classification results of 6 models.

Model	Testing metrics			
	Acc	P_W	R_W	$F1_W$
P1	0.849	0.861	0.843	0.825
P2	0.869	0.852	0.873	0.859
P3	0.864	0.853	0.868	0.844
R1	0.872	0.864	0.875	0.851
R2	0.881	0.869	0.884	0.865
R3	0.887	0.875	0.893	0.879

Table 5: Comparison of results of recall with other classifiers.

Model	Testing metric R									
	Analysis	Backdoors	DoS	Exploits	Fuzzers	Generic	Normal	Recon	Shellcode	Worms
RF	0	0	0.012	0.839	0.741	0.630	0.999	0.754	0.111	0
SVM	0.062	0.033	0.183	0.669	0.730	0.763	0.997	0.645	0.206	0.068
MLP	0.007	0.039	0.076	0.825	0.679	0.891	0.999	0.661	0.508	0.227
LSTM	0	0	0.008	0.853	0.793	0.747	0.998	0.683	0	0.045
LeNet-5	0	0	0	0.852	0.689	0.901	0.996	0.787	0.553	0
AlexNet	0	0.002	0.057	0.833	0.725	0.915	0.999	0.711	0.352	0.114
RLC-CNN	0	0.026	0.077	0.875	0.711	0.926	0.998	0.805	0.370	0
RLF-CNN	**0.435**	**0.482**	**0.311**	**0.860**	**0.732**	**0.918**	**0.993**	**0.797**	**0.712**	**0.818**

Among deep learning algorithms, we are able to build deeper networks to learn more critical features of network traffic due to the residual blocks. Table 5 and Figure 6 demonstrate that our model achieves better results than other deep learning algorithms. With residual learning, CNNs can provide better performances in network intrusion detection.

In terms of minor classes, all the other models perform poorly due to the class imbalance problems in the training set. Our model utilizes focal loss to address the issue above. Although in some dominant classes, RLF-CNN's performance slightly weakens due to their reduced weights in the loss function, RLF-CNN outperforms other classifiers in the classification of minor classes with higher recall values, indicating that focal loss is more suitable in classifying imbalanced data sets and enhancing the detecting capabilities.

To prove our model's ability to detect normal flows and attacks, we compare it with other algorithms using metrics including True Positive and True Negative. The testing results are shown in Table 6.

Among these models, RLC-CCNN is the improved version of RLC-CNN possessing the same class weights as the ones used in the focal loss of RLF-CNN. SMOTE-RF [32] is an algorithm of Random Forest combined with SMOTE. Pelican [19] and S-ResNet [1] are improved

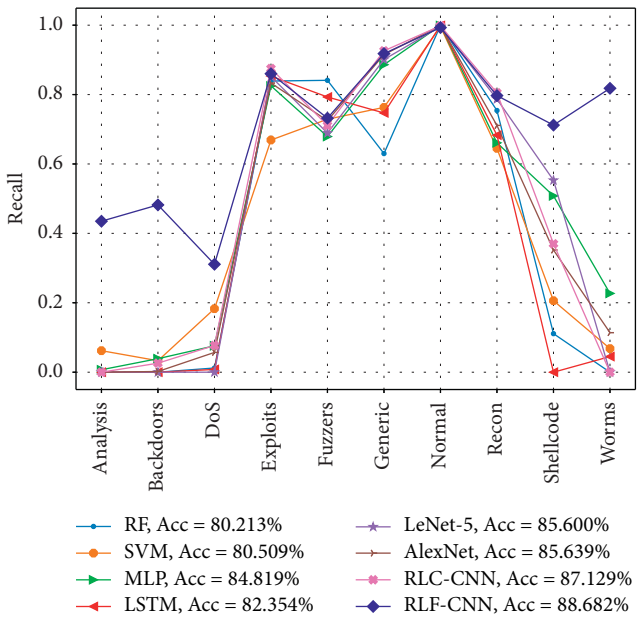

Figure 6: Recall and accuracy of classifiers.

Table 6: Comparison with other detection methods on UNSW-NB15 testing set.

Model	Testing metrics			
	TN	TP	Acc	R_W
RLF-CNN	36758	36278	0.887	0.893
RLC-CCNN	36996	35429	0.879	0.881
SMOTE-RF [32]	36952	32286	0.841	0.826
Pelican [19]	36850	34928	0.872	0.859
S-ResNet [1]	36928	31427	0.830	0.842

residual networks which have faster convergence velocity and better testing results compared to other deep learning algorithms. As shown in Table 6, all the models above can identify over 99.3% of all 37000 normal samples correctly. But compared with other contemporary algorithms for network intrusion detection, RLF-CNN can identify more attacks correctly, given that most of the attacks in the data set are minor samples, showing higher attack detection rates.

Compared with SMOTE-RF, our model detects more attacks while it avoids generating new data. SMOTE-RF generates over 300000 training samples, consuming a lot of time and memory. Also, tradition machine learning algorithms lack the abilities to acquire data features automatically; therefore, with the absence of feature engineering techniques, SMOTE-RF is inferior to others in detecting attacks. Compared with other residual networks, our model got better results in the detection of attacks. It can also be seen that our model outperforms RLC-CCNN. Focal loss enables the model to focus on samples that are harder to learn, and testing results indicate that the focal loss can learn complex samples more efficiently and is superior to class weights in the training phase.

5. Conclusions

In this paper, a network intrusion detection method based on residual learning and focal loss has been proposed. Experimental results show that models with residual learning are easier to train, achieving lower loss values on the training data and higher accuracy rates on the testing data. Compared with other deep learning algorithms, RLF-CNN has achieved better performance in terms of several metrics due to residual learning. And our model uses a modified focal loss function to deal with the class imbalance problem existing in the training data. Also, the proposed model shows better results than a CNN with the same class weights. Despite outstanding results, this study has its potential limitations. Although our model has outperformed other deep learning algorithms in the detection of minor attacks with the focal loss, its performance to detect some dominant classes has weakened due to reduced weights. Therefore, how to improve the model's performance on minor classes without affecting its abilities to detect dominant classes is an important issue that needs to be addressed in the future. Also, UNSW-NB15 data set only contains a few types of attacks; due to the low tolerance for errors in IDSs, we will combine other data sets to cover various types of attacks in the future. Last but not least, due to limited computing resources, deeper neural networks with more residual blocks and normal blocks cannot be tested. So, with more powerful resources in the future, we will continue to perform more experiments and maybe get better results when it comes to detecting network attacks.

Data Availability

The processed UNSW-NB15 data sets used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publishing of this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (no. 61906099) and the Open Fund of Key Laboratory of Urban Land Resources Monitoring and Simulation, Ministry of Natural Resources (no. KF-2019-04-065).

References

[1] Y. Xiao and X. Xiao, "An intrusion detection system based on a simplified residual network," *Information*, vol. 10, no. 11, p. 356, 2019.

[2] F. Safara, A. Souri, and M. Serrizadeh, "Improved intrusion detection method for communication networks using association rule mining and artificial neural networks," *IET Communications*, vol. 14, no. 7, pp. 1192–1197, 2020.

[3] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[4] H. Jégou, F. Perronnin, M. Douze, J. Sanchez, P. Perez, and C. Schmid, "Aggregating local image descriptors into compact codes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 9, pp. 1704–1716, 2012.

[5] Z. Chen, Y. Zhou, and Z. Huang, "Auto-creation of effective neural network architecture by evolutionary algorithm and ResNet for image classification," in *Proceedings of the 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pp. 3895–3900, IEEE, Bari, Italy, October 2019.

[6] S. Rodda and U. S. R. Erothi, "Class imbalance problem in the network intrusion detection systems," in *Proceedings of the 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, pp. 2685–2688, IEEE, Chennai, India, March 2016.

[7] B. Zhang, Z. Liu, Y. Jia et al., "Network intrusion detection method based on PCA and Bayes algorithm," *Security and Communication Networks*, vol. 2018, Article ID 1914980, 11 pages, 2018.

[8] M. Kasongo and Y. Sun, "Performance analysis of intrusion detection systems using a feature selection method on the UNSW-NB15 dataset," *Journal of Big Data*, vol. 7, no. 1, p. 105, 2020.

[9] C. Sun, J. Xing, Q. Yang, and D. Han, "Intrusion detection methods based on improved Naive Bayesian," *Microcomputer and its Applications*, vol. 2017, no. 1, pp. 8–10, 2017.

[10] J. Wang, X. Chen, L. Cao et al., "Individual rubber tree segmentation based on ground-based LiDAR data and faster R-CNN of deep learning," *Forests*, vol. 10, no. 9, p. 793, 2019.

[11] J. Pons and X. Serra, "Randomly weighted CNNs for (music) audio classification," in *Proceedings of the ICASSP 2019—2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 336–340, IEEE, Brighton, UK, May 2019.

[12] Y. Peng, M. Liao, H. Deng et al., "CNN-SVM: a classification method for fruit fly image with the complex background," *IET Cyber-Physical Systems: Theory & Applications*, vol. 5, no. 2, pp. 181–185, 2020.

[13] T. Qian, Y. Wang, M. Zhang et al., "Intrusion detection method based on deep neural network," *Journal of Huazhong University of Science & Technology*, vol. 46, no. 1, pp. 6–10, 2018.

[14] Y. Lai, J. Zhang, Z. Liu, and M. Alazab, "Industrial anomaly detection and attack classification method based on convolutional neural network," *Security and Communication Networks*, vol. 2019, Article ID 8124254, 11 pages, 2019.

[15] K. He, X. Zhang, S. Ren et al., "Deep residual learning for image recognition," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, IEEE, Las Vegas, NV, USA, June 2016.

[16] M. Feng, H. Lu, and Y. Yu, "Residual learning for salient object detection," *IEEE Transactions on Image Processing*, vol. 29, pp. 4696–4708, 2020.

[17] P. McAllister, H. Zheng, R. Bond, and A. Moorhead, "Combining deep residual neural network features with supervised machine learning algorithms to classify diverse food image datasets," *Computers in Biology and Medicine*, vol. 95, pp. 217–233, 2018.

[18] T. Li, H. Song, K. Zhang, and Q. Liu, "Recurrent reverse attention guided residual learning for saliency object detection," *Neurocomputing*, vol. 389, pp. 170–178, 2020.

[19] P. Wu, H. Guo, and N. Moustafa, "Pelican: a deep residual network for network intrusion detection," in *Proceedings of the 2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pp. 55–62, IEEE, Valencia, Spain, July 2020.

[20] N. Chouhan, A. Khan, and H. Khan, "Network anomaly detection using channel boosted and residual learning based deep convolutional neural network," *Applied Soft Computing Journal*, vol. 83, 2019.

[21] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, pp. 318–327, 2020.

[22] S.-H. Choi and S. H. Jung, "Stable Acquisition of fine-grained segments using batch normalization and focal loss with L1 regularization in U-Net structure," *International Journal of Fuzzy Logic and Intelligent Systems*, vol. 20, no. 1, pp. 59–68, 2020.

[23] C. Wang, C. Deng, and S. Wang, "Imbalance-XGBoost: leveraging weighted and focal losses for binary label-imbalanced classification with XGBoost," *Pattern Recognition Letters*, vol. 136, pp. 190–197, 2020.

[24] Y. Zhao, F. Lin, S. Liu, Z. Hu, H. Li, and Y. Bai, "Constrained-focal-loss based deep learning for segmentation of spores," *IEEE Access*, vol. 7, pp. 165029–165038, 2019.

[25] V. Abhishek and R. Virender, "Statistical analysis of CIDDS-001 dataset for network intrusion detection systems using distance-based machine learning," *Procedia Computer Science*, vol. 125, pp. 709–716, 2018.

[26] C. Sarika and K. Nishtha, ""Analysis of KDD-Cup'99, NSL-KDD and UNSW-NB15 datasets using deep learning in IoT," *Procedia Computer Science*, vol. 167, pp. 1561–1573, 2020.

[27] N. Moustafa and J. Slay, "The evaluation of Network Anomaly Detection Systems: statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set," *Information Security Journal: A Global Perspective*, vol. 25, no. 1–3, pp. 18–31, 2016.

[28] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[29] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[30] L. Liu, P. Wang, J. Lin, and L. Liu, "Intrusion detection of imbalanced network traffic based on machine learning and deep learning," *IEEE Access*, vol. 9, pp. 7550–7563, 2021.

[31] Y. Chang, W. Li, and Z. Yang, "Network intrusion detection based on random forest and support vector machine," in *Proceedings of the 2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, pp. 635–638, IEEE, Guangzhou, China, July 2017.

[32] X. Tan, S. Su, Z. Huang et al., "Wireless sensor networks intrusion detection based on SMOTE and the random forest algorithm," *Sensors*, vol. 19, no. 1, p. 203, 2019.