Abnormal Traffic Detection Based on Character-Level Convolutional Neural Networks

Runjie Liu, Bingjie Guo*, Chaoqiang Ji and Lei Shi

National Supercomputing Center in Zhengzhou, Zhengzhou University, Zhengzhou 450001, Henan, China

Abstract

The common preprocessing method for abnormal traffic detection is feature-level preprocessing, which is more complex and requires some a priori knowledge (number, meaning, and characteristics of features, etc.) to be considered. A character-level preprocessing-based abnormal traffic detection method is proposed. Starting from the fine-grained character level, the network traffic data is treated as a sequence of characters, and the character sequence is encoded as a vector using four character encodings of the character-level convolutional neural network, and the vectors of each data after character encoding are aggregated into a matrix and fed into the improved convolutional neural network. Experimental results show that the ASCII encoding model in this paper performs best. Compared with the Sparse encoding model, the verification time was reduced by 1.07s and the accuracy rate is increased by 3.21%; Compared with the feature-level pretreatment model, the pre-treatment was simplified without considering the prior knowledge of the data, and the accuracy rate is increased by 4.49%.

Keywords

CharCNN, Character encoding, Abnormal traffic detection

1.Introduction

In the 21st century, the network has become the largest and most comprehensive information center with the fastest delivery speed that mankind has ever had. The massive amount of traffic data generated by the network has security risks, and once it is maliciously attacked by hackers, it will bring serious harm to people's daily life and even national security. Anomaly detection is the focus of research in network security, in which network traffic anomaly detection can detect whether there is an attack in the network through the analysis of traffic data.

In recent years, deep learning has started to emerge, and good research results have been achieved in natural language processing (NLP) [1], image recognition [2], and speech recognition [3]. The nonlinear network structure formed by constructing more hidden layers through deep learning methods can learn features in data autonomously and still performs well in the application scenario of massive data. Therefore, many scholars have also applied deep learning in the field of cyber security. Convolutional neural networks (CNNs), as the main model of deep learning, are widely used in the detection of malicious codes, malicious HTTP requests, and malicious Web request [4]–[6].

2.Related Work

Deep learning is becoming increasingly popular in academic research due to its strong autonomous learning capability and is used in anomalous traffic detection. However, anomalous traffic detection data includes both continuous and discrete data, and the order of magnitude difference between different feature attributes in the data is large. To construct a reasonable dataset, it is usually necessary to pre-process the dataset. The data preprocessing stage usually contains two parts: character feature

ICBASE2022@3rd International Conference on Big Data & Artificial Intelligence & Software Engineering, October 21-23, 2022, Guangzhou, China

* Corresponding Email: hhhhyay@163.com (Bingjie Guo)

numericalization and numerical feature normalization. Numerical characterization refers to the mapping of data using one-hot encoding and other encoding methods; numerical feature normalization refers to the reduction of data values to the range of [0,1] using Min-Max, Z-score, and other normalization methods to eliminate the influence of size on feature attributes[7].

After research, according to the overall preprocessing process of abnormal traffic detection based on deep learning, it can be divided into feature level, hybrid level, and character level. Most of the preprocessing methods are used for feature-level preprocessing. Yin et al. [8] proposed a deep learning approach for intrusion detection based on recurrent neural networks (RNN-IDS), where the data were put into RNNs after feature-level preprocessing of the dataset, and the performances of the model in binary and multi-classification were investigated, as well as the impact of the number of neurons and different learning rates on the performances of the proposed model. Anju Krishnan et al. [9] proposed a one-dimensional convolutional neural network-based intrusion detection (1DCNN-IDS), where data after numerical and normalized feature-level preprocessing of character features were fed into a 1D-CNN, and experimental results showed superiority over traditional machine learning models. Based on the advantages of CNN in the image domain, Xiao et al. [10] transformed the data after feature-level preprocessing and dimensionality reduction using PCA or AE into grayscale images input to CNN for feature extraction and classification, and the experimental results were better than traditional machine learning algorithms. Feature-level preprocessing is performed only with prior knowledge of the a priori knowledge of the features in the dataset (number, meaning and characteristics of the features, etc.), and in a real network environment, where there are more possible values of character types, the character feature numericalization will be incomplete and the generalization is not good.

In terms of hybrid-level preprocessing, Min et al. [11] mixed feature-level and word-level using two modern NLP techniques: word embedding and Text-CNN, extracted salient features from the payload, and then performed a random forest algorithm on the combination of statistical features and payload features for classification. This method still requires prior knowledge of the statistical features in the dataset, which also complicates pre-processing.

In terms of character-level preprocessing, Lin et al. [12] proposed an intrusion detection method based on character-level convolutional networks (CharCNN). The traffic data was treated as a special text sequence, and each character of the data is first transformed into a character vector using one-hot coding, and the vectors are converged into a matrix and fed directly into the convolutional neural network model, and the final experimental results show the highest accuracy rate compared with the traditional machine learning model. This process simplifies preprocessing by eliminating the need to know the prior knowledge of the dataset, but the character vector formed using one-hot encoding increases the dimensionality, which leads to a very sparse phenomenon.

CharCNN was investigated due to the advantages of character-level preprocessing. Andrei Karpau et al. [13] introduced four encoding methods for CharCNN cited in NLP: Sparse, Sparse Group, ASCII, ASCII Group, and ASCII had a faster convergence speed. Joseph et al. [14] used the character encoding embedding method of UTF-8 to encode each character as a vector using a UTF-8 binary value. Experiments showed that training was faster and had higher performance, and it was easier to learn from character-level text.

Therefore, this paper proposes a character-level convolutional neural network-based anomalous traffic detection model (ATD-CharCNN). Applying the character-level convolutional neural network in the field of NLP to anomalous traffic detection, the network traffic data is treated as a special kind of character text from a more fine-grained character level. The main work of this paper is as follows：

(1) Four character encodings for the character-level convolutional neural network are introduced for character-level preprocessing of network traffic data; this method simplifies the preprocessing process without knowing the prior knowledge of the dataset features.

(2) Experiments are conducted on the public dataset NSL-KDD [15], and then the effectiveness of the four character encodings in abnormal traffic detection is verified and compared with the detection performance of the model with feature-level preprocessing.

3.Abnormal Traffic Detection Based on CharCNN

This paper does not use the mainstream preprocessing method but introduces the character-level convolutional neural network used in the natural language field for abnormal traffic detection, which simplifies the preprocessing steps and makes the model more generalizable. The overall process of the method is shown in Figure 1, which mainly includes three parts: (i) data preprocessing, the character encoding of traffic data as characters, convergence into vector matrix; (ii) construction of CNN model; (iii) Binary and multi-class training and testing using the NSL-KDD classic dataset.

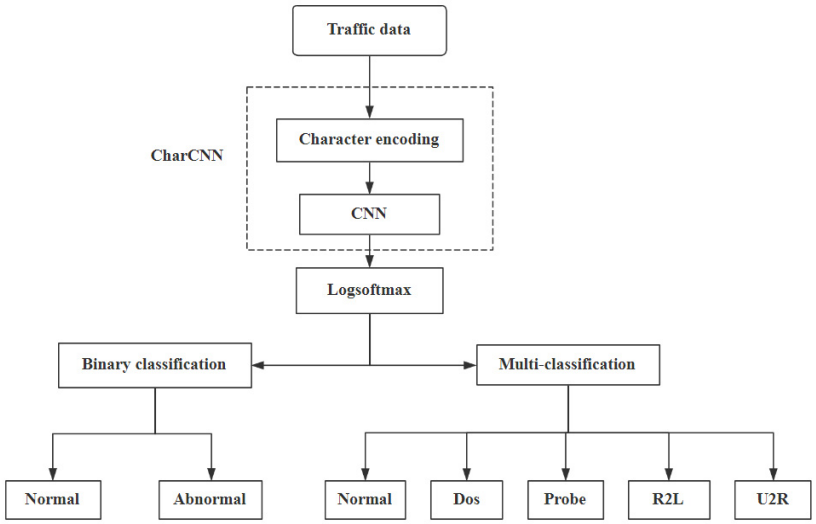

Figure 1. ATD-CharCNN overall flow chart.

3.1.The binary classification and multi-classification of NSL-KDD

The distribution of the NSL-KDD dataset is shown in Table 1, and the network traffic in the dataset can be divided into two categories: Normal traffic and Abnormal traffic. NSL-KDD can also be divided into multi-classification in more detail: Normal traffic, Dos attack traffic, Probe attack traffic, R2L attack traffic, and U2R attack traffic.

From NSL-KDDTrain+, we can see that the amount of normal and abnormal data in the binary classification is more balanced, and the detection effect will be higher when the abnormal traffic detection model performs binary classification, but only abnormal traffic can be detected, and it is not possible to know more deeply which kind of abnormal traffic it is. In multi-classification, the abnormal traffic is classified more specifically, and from the overall perspective, the distribution of the five traffic data in the training set is unbalanced, and although specific categories of abnormal traffic can be detected, the detection rate is not high.

Table 1. Distribution of traffic in NSL-KDD

Binary classification	Multi-classification	NSL-KDDTrain+	NSL-KDDTest+
Normal	Normal	67343	9711
Abnormal	Dos	45927	7458
	Probe	11656	2421
	R2L	995	2754
	U2R	52	200
	Total	125973	22544

3.2.Character level preprocessing

In the classic traffic detection dataset NSL-KDD, there are numerical and character features.

However, for the neural network to better identify these features, the character features need to be numericalized before model training, and normalization is required to eliminate singular values. For example: in the NSL-KDD dataset, there are three character-type discrete features: 'protocol_type', 'service', and 'flag', which are first quantized and encoded, and converted into digital representations. In this way of preprocessing, the quantification and normalization of features can only be carried out according to the prior knowledge of the features of the dataset, including the number, meaning, and characteristics of features.

The character-level preprocessing method in this paper does not require such prior knowledge and only needs to process each character in the data. This model pre-processes traffic feature sequences with character length l_0. If the characters in a sequence are less than l_0, then spaces will be added to the header of the feature sequence to make it l_0 length; if there are more characters in a sequence than l_0, since the sequence header is often the basic feature of the traffic, it is not compatible with attacks. If the correlation is small, the characters beyond the head of the feature sequence are deleted [12].

The character table composed of all the characters that may appear in the statistical dataset is as follows. Table 2 contains the upper and lower case of 26 letters, ten numbers, and 3 punctuation marks that appear in the dataset.

Table 2. Character List.

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	1	2	3	4	5	6	7	8	9	,	.	_													

The characters that appear are encoded and mapped in four ways, as follows:

Sparse: The essence of Sparse encoding is one-hot encoding. All characters appearing in the dataset are first counted and numbered, and each character is encoded as a vector of zeros except for the number of character that appears, which is 1. All characters in this data set are 65 characters in Table 2, in which the uppercase letters are converted to lowercase letters to reduce sparsity, 39 characters are quantified as 1×39 vectors, e.g., the Sparse encoding of a is a vector containing 38 zeros: [1,0,0,0,0,0...,0,0]. For all characters except the character table, it is encoded as an all-zero vector with 39 zeros: [0,0,0,0,0,0...,0,0].

Sparse Group: This code adds 4 bits to identify the character type based on Sparse. Character types are lowercase letters, uppercase letters, numbers, and punctuation. For example, the Sparse Group encoding of the capital letter A is [1,0...,0,0,1,1,0,0], which is the Sparse encoding of a [1,0,0,0, 0...,0,0] adds [1,1,0,0] to identify as uppercase letters. This encoding provides more information about the characters and enhances the awareness of the model [13].

ASCII: The first two encodings contain a lot of zeros and high dimensionality, resulting in a very sparse phenomenon. ASCII encoding is a non-sparse encoding of an eight-bit binary based on the ASCII character set. Each character of the character table is encoded as a 1×8 vector according to the binary code value of ASCII, such as the ASCII encoding of A is [0,1,0,0,0,0,0,0,0,1].

ASCII Group: Like the Sparse Group encoding, it is a 1×12 vector with 4 bits added to the ASCII encoding to identify the character type. For example, the ASCII Group code of A is [0,1,1,0,0,0,0,0,1,1,1,1,0,0].

Table 3. Four character encoding methods.

Encoding	Size of the Alphabet	Vector Length	Encoding Examples
Sparse	40	39	'a': [1,0,0,0,0...,0,0] 'A': no upper case ',': [0,0,0,0...,1...,0] '1': [0,0,0,0...,1...,0]
Sparse Group	66	43	'a': [1,0...,0,0,1,0,0,0] 'A': [1, 0 ..., 0, 0, 1, 1, 0, 0] ',': [0...,1...,0,0,0,1,0] '1': [0...,1...,0,0,0,0,1]

ASCII	66	8	‘a’: [0,1,1,0,0,0,0,1] ‘A’: [0, 1, 0, 0, 0, 0, 0, 1] ‘,’: [0,0,1,0,1,1,0,0] ‘1’: [0,0,1,1,0,0,0,1]
ASCII Group	66	12	‘a’: [0,1,1,0,0,0,0,1,1,0,0,0] ‘A’: [0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 0, 0] ‘,’: [0,0,1,0,1,1,0,0,0,0,0,1] ‘1’: [0,0,1,1,0,0,0,1,0,0,1,0]

Taking ASCII encoding as an example, the traffic feature sequence of a traffic record will be finally converted into a matrix X of size $l_0 \times 8$. This matrix will be used as input to the convolutional neural network to train or test the model. The character length l_0 is set to 177, which is not only greater than the entire character length of any traffic feature sequence in the experimental dataset NSL-KDD used but also greater than the character length of most actual network traffic feature sequences [12].

Figure 2 shows the process of quantizing and assembling a matrix of ASCII encoding of a piece of traffic data and features. Among them, each block combination represents an ASCII-encoded vector corresponding to a character. For example, the eight-bit ASCII code values corresponding to "t", "c", and "p" are {01110100}, {01100011}, {01110000}, respectively. The white squares represent 0, and the black squares represent 1. All the ASCII-encoded vectors of the feature sequence of a flow are aggregated to form a feature matrix that can be used as CNN input.

By performing character-level preprocessing on traffic data, the preprocessing no longer needs to consider the meaning and content of features, and the process is simplified. In addition, in a real network, if a new feature value of a feature is encountered, especially a character feature, the processing process does not need to readjust the preprocessing module in the model.

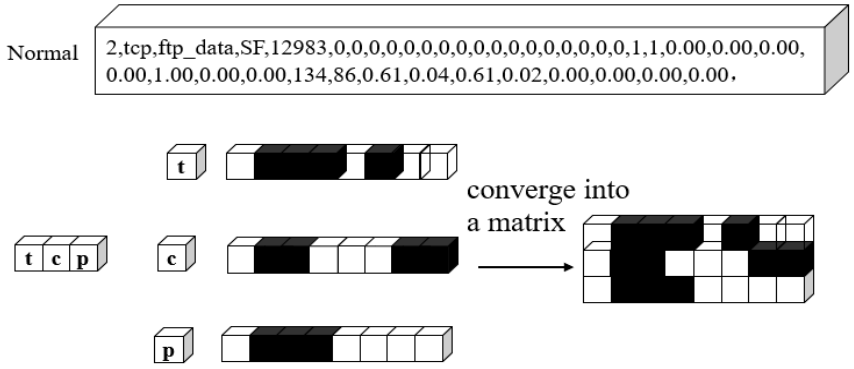

Figure 2. A piece of traffic data and ASCII encoding quantization process.

3.3. Structure of ATD-CharCNN model

Based on the CharCNN-IDS model [12], this paper improves the preprocessing method and neural network structure and improves the feature extraction ability and discrimination ability of the model. In the CharCNN model structure proposed by Zhang et al. [16], the convolutional neural network has 9 layers, including 6 convolutional layers and 3 fully connected layers. Compared with text classification in the field of natural language processing, the amount of data to convert the sequence of traffic data into text input is smaller (for example, l_0 in the natural language field is mostly 1014, and l_0 in the traffic data set in this paper is 177, much smaller than 1014). Therefore, in the ATD-CharCNN model, 3 convolution blocks and 3 fully connected blocks are enough to identify complex text patterns without consuming too many resources, and too many network layers will also cause overfitting. According to the experimental results, the convolutional neural network model in ATD-CharCNN is set as follows: the network contains a total of 6 layers, including 3 convolutional layers and 3 fully connected layers. The layout of the entire network model is shown in Table 4.

Table 4. Network structure layout table in ATD-CharCNN.

Type	Function	Feature amount	Kernel	Stride
Conv[1]	Convolution+ReLU	256	7	1
	Max pooling	256	3	3
Conv	Convolution+ReLU	256	7	1
	Max pooling	256	3	3
Conv	Convolution+ReLU	256	3	1
	Max pooling	256	3	3
FC[2]	Linear+ReLU+Dropout	1280	1024	-
FC	Linear+ReLU+Dropout	1024	1024	-
FC	Linear	1024	2/5	-

1. Conv: Convolutional layer
2. FC: Fully-connected layer

The traffic data is processed as special text, so the convolutional layers in the above table are all one-dimensional convolutions. To increase the nonlinear mapping learning ability of the model, ReLU is selected as the activation function for each layer in the network structure, and the maximum pooling layer is added between the convolutional layers to reduce feature redundancy and output local optimal features. Using the regularization method Dropout technology in the fully connected layer can avoid overfitting. The output of the last fully-connected layer uses Logsoftmax for binary and multi-classification. Logsoftmax is the logarithm of softmax, it can avoid solving the overflow and underflow problem, and can speed up the operation speed and improve the data stability. The mathematical expression is shown in formula (1):

$$L_i = \log \frac{e^{z_i}}{\sum_j^k e^{z_j}} \tag{1}$$

where z_i is the output value of category i in the fully connected layer, and k is the number of output nodes, that is, the total number of categories classified.

4.Experiment

The experiment of this model is trained and tested on the Jiutian·Bisheng platform. The graphics card is NVIDIA TESLA V100, and the video memory is 32GB. Based on the deep learning framework of Pytorch 1.8, the ATD-CharCNN model in this paper is implemented.

4.1. Performance metrics

The following metrics are used to evaluate the performance of ATD-CharCNN: Accuracy, Precision, Recall, F1-score, False Positive Rate (FPR), and AUC (Area Under Curve) values, as well as training and testing time. The larger the AUC value is, the better the classification performance is. The specific expressions are shown in formulas (2-6):

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \tag{2}$$

$$\text{Precision} = \frac{TP}{TP+FP} \tag{3}$$

$$\text{Recall} = \frac{TP}{TP+FN} \tag{4}$$

$$\text{F1-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision}+\text{Recall}} \tag{5}$$

$$FPR = \frac{FP}{FP+TN} \tag{6}$$

where TP is positive cases detected as positive, TN is negative cases detected as negative, FP is negative cases detected as positive, and FN is positive cases detected as negative. Accuracy, which is used to reflect the classifier's ability to determine various types of samples; Precision, the proportion of correct attack predictions to the attack predictions; Recall, which can be the proportion of correct attack samples among all attack samples, reflects the classifier's ability to detect network attacks; F1-score, a weighted average of precision and recall; and False Positive Rate (FPR), which refers to the proportion of misclassified normal traffic as abnormal traffic to all normal traffic.

4.2.Experimental Results and Analysis

4.2.1.Experimental data and parameter settings.

The experiments in this paper are to use four character encodings of CharCNN in abnormal traffic detection and to compare the performance. The experimental dataset is NSL-KDD, where the training set is NSL-KDDTrain+, and this dataset contains 125,973 data. The test set NSL-KDDTest+ has 22,544 data. The NSL-KDD dataset is introduced in 3.1. In the experiments of binary classification and multi-classification, the batch size of the training process (batch size) is 256, and the training process is optimized using the Adam optimizer with a loss function of cross-entropy function.

4.2.2.Performance in binary classification.

The two classifications in network traffic are divided into normal traffic and abnormal traffic. The NSL-KDDTrain+ dataset is used to train the ATD-CharCNN models with four character encodings. The loss curves and accuracy curves of the training are shown in Figure 3 and Figure 4 below. Table 5 shows the training time, test time, and experimental results at NSL-KDDTest+ for each character encoding.

From above Figure 3 and Figure 4, we can see that "Sparse" has less error and higher accuracy on the training set compared to "ASCII" and fits the training set better. From Table 5, it can be seen that "ASCII" outperforms "Sparse" in all four character encodings: shorter time, higher accuracy, lower false positive rate, and larger AUC value. The "Group" code with identifier type is better in terms of accuracy and AUC value, but it performs worse in terms of false positive rate, training time, and verification time. Combining the four graphs and the training set and test set accuracies in the table, we can see that "ASCII" has better generalization.

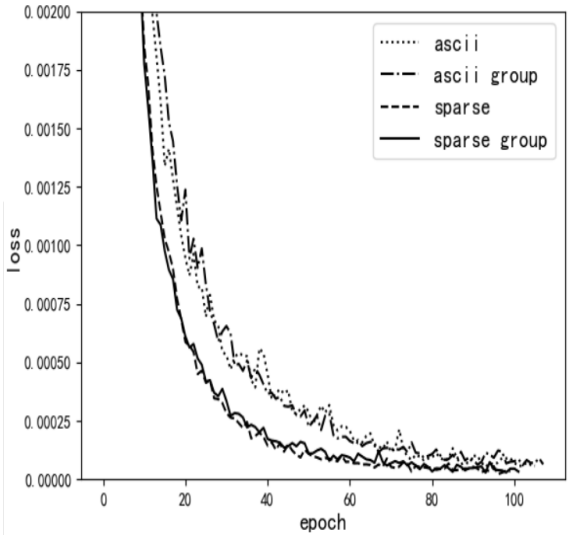

Figure 3. Loss curves.

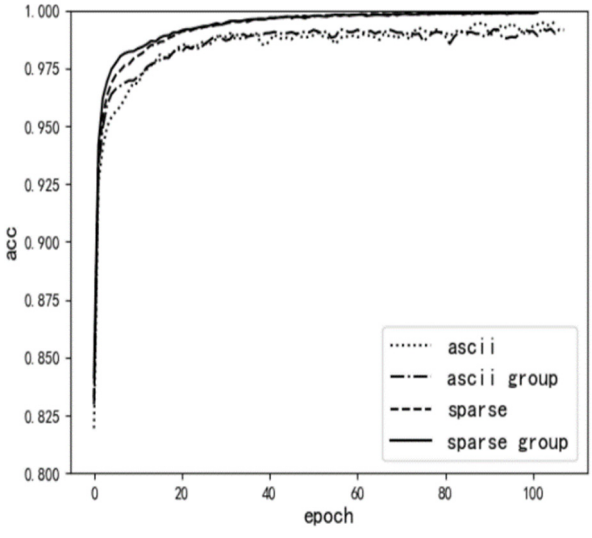

Figure 4. Accuracy curves.

Table 5. The metrics of ATD-CharCNN in NSL-KDD.

Encoding	Training time(s)	Testing time(s)	Accuracy(%)	FAR(%)	AUC
ASCII	3620.96	2.50	85.61	4.42	0.8682
ASCII Group	3938.76	2.54	86.29	5.13	0.8733
Sparse	3987.17	3.57	82.40	7.25	0.8366
Sparse Group	4231.98	3.66	82.91	7.83	0.8404

From above Figure 3 and Figure 4, we can see that "Sparse" has less error and higher accuracy on the training set compared to "ASCII" and fits the training set better. From Table 5, it can be seen that "ASCII" outperforms "Sparse" in all four character encodings: shorter time, higher accuracy, lower false positive rate, and larger AUC value. The "Group" code with identifier type is better in terms of accuracy and AUC value, but it performs worse in terms of false positive rate, training time, and verification time. Combining the four graphs and the training set and test set accuracies in the table, we can see that "ASCII" has better generalization.

ASCII is an 8-bit binary non-sparse encoding, it is less memory intensive and takes less time to train than the Sparse encoding which contains 38 zeros in 39 bits. And ASCII is an international set of character encoding containing upper and lower case English characters, numbers, and punctuation marks. For data sets where all characters are in the ASCII character table, ASCII encoding is a more time- and memory-efficient encoding with a higher accuracy rate. In the field of NLP, it is better to distinguish between upper and lower case English characters than a model with only lower case English characters. From the accuracy of the Sparse Group encoding and ASCII encoding in Table 5, the encoding that artificially sets the identity character type does not have good detection performance for encodings that contain these character types on its own.

4.2.3.Performance in multi-classification.

It can be seen from Section 3.1 that the multi-classification in the network traffic data in the NSL-KDD dataset is divided into five classifications. They are normal traffic, Dos attack traffic, Probe attack traffic, R2L attack traffic, and U2R attack traffic. There are few samples of the latter two abnormal flows. The Recall values and Accuracy values of various character encoding multi-classifications of ATD-CharCNN are shown in Table 6.

Table 6. Results of Recall and Accuracy values (%).

Encoding	Dos	Probe	R2L	U2R	Accuracy
ASCII	88.75	62.95	2.29	0	77.08
ASCII Group	92.05	60.63	2.80	0	76.89
Sparse	81.12	62.12	0.62	0	75.29
Sparse Group	80.63	66.42	1.20	0	74.75

From the table, the recall values are larger for the attack types with large sample sizes in the dataset, and the detection rate of ATD-CharCNN for Dos attack traffic and Probe attack traffic is very high, and the low detection rates for the latter two are due to the small sample size, and the model learns less valid information when performing learning. From the overall Accuracy values, the ATD-CharCNN model with ASCII encoding has the best detection performance in multi-classification, and the ASCII Group encoding with four more identifying characters is not very useful in multi-classification, and the ASCII character set itself already identifies the character type.

4.2.4.Comparison: ATD-CharCNN and CNN-IDS.

To verify the effectiveness of the character-level preprocessing proposed in this paper in abnormal traffic detection, the detection accuracy of the convolutional neural network model with feature-level

preprocessing for intrusion detection is listed, and the detection accuracy of different models and the detection accuracy of the model proposed in this paper are compared, and the results are shown in Table 7.

Table 7. Metrics of ATD-CharCNN and CNN-IDS (%).

Model	CNN-IDS (GoogleNet) [12]	CNN-IDS(1D-CNN)	Proposed method
Accuracy	77.04	81.12	85.61
Precision	91.66	89.69	95.89
Recall	65.64	75.52	78.06
F1-score	76.50	82.00	86.06
FPR	7.89	11.47	4.42

As can be seen from Table 7, the metrics value of the proposed character-level preprocessing is better than the feature-level preprocessing, indicating that the proposed method has better detection performance. Character-level preprocessing does not require numerical characterization by knowing the a priori knowledge of the dataset, nor does it require normalization of the singular values, which simplifies the pre-processing process and improves the detection performance.

5.Conclusion and future work

An ATD-CharCNN model with character-level preprocessing is proposed and experimentally validated on the dataset NSL-KDD. The effectiveness of character-level preprocessing in anomalous traffic detection is shown, where ASCII encoding has the best detection performance in binary and multi-classification with the shortest time and high detection rate. Compared with the feature-level preprocessing model, the model in this paper greatly simplifies the preprocessing process and the accuracy rate is increased by 4.49%, improving the detection performance. The model has a low detection rate for minority attacks in five classifications, and in future work, the imbalance of data will be further reduced to improve the detection rate for minority attacks.

6.References

[1] Z. Rezaei, B. Eslami, M. -A. Amini, et al. Hierarchical Three-module Method of Text Classification in Web Big Data[C]//2020 6th International Conference on Web Research (ICWR), 2020:58-65.

[2] Li H and Li G M. Research on Facial Expression Recognition Based on LBP and Deep Learning[C]//2019 International Conference on Robots & Intelligent System (ICRIS), 2019:94-97.

[3] Qu F K, Liu J L, Li W C, et al. Design of Intelligent Classification Trash Bin Based on Speech Recognition[C]//7th International Conference on Electronic Technology and Information Science, 2022:1-5.

[4] Fu Y X, Lu T L, Ma Z L. One-Hot based CNN malicious code detection technology[J]. Computer Applications and Software, 2020,37(1):304-308+333.

[5] Ito Michiaki, Iyatomi, Hitoshi. Web Application Firewall using Character-level Convolutional Neural Network[C]//IEEE 14th International Colloquium on Signal Processing and its Applications (CSPA).2018:103-106.

[6] Rafael Brinhosa, Marcos A, et al. Comparison between LSTM and CLCNN in detecting malicious requests in web attacks[C]//in Proceedings of the 21st Brazilian Symposium on Information and Computational Systems Security, Belém, 2021:113-126.

[7] Jian S J, Lu Z G, Du D, et al. A Survey of Network Intrusion Detection Technology[J]. Journal of Cyber Security, 2020,5(4):96-122.

[8] Yin C L, Zhu Y F, Fei J L, et al. A deep learning approach for intrusion detection using recurrent neural networks［J］. IEEE Access,2017(5):21954-21961.

[9] Anju Krishnan and S. T. Mithra. A Modified 1D-CNN Based Network Intrusion Detection System[J]. IJRESM, 2021,4(6): 291–294.

[10] Xiao Y, Xing C, Zhang T, et al. An intrusion detection model based on feature reduction and convolutional neural networks[J]. IEEE Access, 2019,7:42210-42219.
[11] Min E X, Long J, Liu Q, et al. TR-IDS: Anomaly-based intrusion detection through text-convolutional neural network and random forest[J]. Security and Communication Networks, 2018:1-9.
[12] Lin S Z, Shi Y, Xue Z. Character-level intrusion detection based on convolutional neural networks[C]//International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, 2018:1-8.
[13] Andrei Karpau, Markus Hofmann. Input encodings for Character Level Convolutional Neural Networks[J]. Digitale Welt, 2020,4(1):26-31.
[14] Joseph D. Prusa, Taghi M. Khoshgoftaar. Designing a Better Data Representation for Deep Neural Networks and Text Classification[C]//2016 IEEE 17th International Conference on Information Reuse and Integration (IRI) (2016): 411-416.
[15] University of New Brunswick. NSL-KDD dataset [DS/OL]. https://www.unb.ca/cic/datasets/nsl.html.
[16] Zhang X, Zhao J, Le C Y. Character-level convolutional networks for text classification[C]//Advances in neural information processing systems. 2015:649-657.