



## Network Intrusion Traffic Detection Based on Feature Extraction

Xuecheng Yu<sup>1</sup>, Yan Huang<sup>2</sup>, Yu Zhang<sup>1</sup>, Mingyang Song<sup>1</sup> and Zhenhong Jia<sup>1,3,\*</sup>

<sup>1</sup>College of Information Science and Engineering, Xinjiang University, Urumqi, 830046, China

<sup>2</sup>Network Department, China Mobile Communications Group Xinjiang Co, Ltd. Urumqi, Urumqi, 830011, China

<sup>3</sup>Xinjiang Signal Detection and Processing Key Laboratory, Xinjiang University, Urumqi, 830000, China

\*Corresponding Author: Zhenhong Jia. Email: jzhh@xju.edu.cn

Received: 14 August 2023 Accepted: 08 November 2023 Published: 30 January 2024

### ABSTRACT

With the increasing dimensionality of network traffic, extracting effective traffic features and improving the identification accuracy of different intrusion traffic have become critical in intrusion detection systems (IDS). However, both unsupervised and semisupervised anomalous traffic detection methods suffer from the drawback of ignoring potential correlations between features, resulting in an analysis that is not an optimal set. Therefore, in order to extract more representative traffic features as well as to improve the accuracy of traffic identification, this paper proposes a feature dimensionality reduction method combining principal component analysis and Hotelling's T<sup>2</sup> and a multilayer convolutional bidirectional long short-term memory (MSC\_BiLSTM) classifier model for network traffic intrusion detection. This method reduces the parameters and redundancy of the model by feature extraction and extracts the dependent features between the data by a bidirectional long short-term memory (BiLSTM) network, which fully considers the influence between the before and after features. The network traffic is first characteristically downsampled by principal component analysis (PCA), and then the downsampled principal components are used as input to Hotelling's T<sup>2</sup> to compare the differences between groups. For datasets with outliers, Hotelling's T<sup>2</sup> can help identify the groups where the outliers are located and quantitatively measure the extent of the outliers. Finally, a multilayer convolutional neural network and a BiLSTM network are used to extract the spatial and temporal features of network traffic data. The empirical consequences exhibit that the suggested approach in this manuscript attains superior outcomes in precision, recall and F1-score juxtaposed with the prevailing techniques. The results show that the intrusion detection accuracy, precision, and F1-score of the proposed MSC\_BiLSTM model for the CIC-IDS 2017 dataset are 98.71%, 95.97%, and 90.22%.

### KEYWORDS

Network intrusion traffic detection; PCA; Hotelling's T<sup>2</sup>; BiLSTM

## 1 Introduction

Network traffic intrusion detection is mainly used for detecting network traffic anomalies and preventing malicious requests. With the gradual increase in network data traffic, the dimensionality of the traffic is increasing, which is causing some difficulties in traffic classification. While the traffic becomes more characteristic, we can analyze it in more detail. However, it also poses two problems.



This work is licensed under a Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The first problem is that the proliferation of relevant features leads to the generation of useless and redundant features and increasingly longer training times. The second issue is that not all features are valuable in describing the flow, which may also lead to bias in the classification step. Therefore, feature extraction holds an important position in network traffic intrusion and network traffic classification.

With the progression of machine learning, the exploration of intrusion detection is also evolving rapidly. Machine learning can be easily embedded in network intrusion detection systems. Modern network intrusion detection systems analyze network traffic based on a number of characteristics, such as the distribution of arrival times, packet sizes, specific TCP/IP flags, and the percentage of backward and forward flows. This information is very helpful for anomalous traffic detection, but in many cases, traditional signature-based identification techniques do not detect these features. The classification methods for feature extraction can be classified as supervised [1], unsupervised and semisupervised [2]. The supervised approach is to select a subset of features using the labeled data and measure the importance of the features by specific criteria. In contrast, unsupervised methods do not require knowledge of the intrinsic structure of the data and do not assume any a priori knowledge to select important features. Semisupervised learning is a hybrid-based approach that improves the results of feature extraction by representing some unlabeled features based on some labeled data.

In unsupervised and semisupervised approaches, Salo et al. [3] proposed a hybrid algorithm that combines information gain and principal component analysis for the analysis of irrelevant features in the dataset. Several integrated classifiers are used, which provide good performance in terms of accuracy and time. Pan et al. [4] introduced an unsupervised end-to-end deep learning system based on the Reuben Software Modeling Tool (RSMT) that automatically monitors and simultaneously describes the behavior of web applications at runtime. However, the results tested using a supervised type of classification method were poor. Yao et al. [5] proposed a multistage semisupervised intrusion detection system to improve the recall rate using the KDDcup99 dataset. This model shows a great improvement in recognition ability and accuracy in unknown cases; however, the flexibility of hyperparameter selection for this model is insufficient. A semisupervised feature grouping method based on the union of the linear correlation value and cuttlefish method was suggested by Mohammadi et al. [6]. The KDDcup99 dataset was used to evaluate the entire system, and despite the fact that the percentage of false positives was substantially lower, the training period was lengthy. A semisupervised anomaly detection system based on the least squares approach and a multivariate statistical network detection algorithm was proposed by Camacho et al. [7]. The system's experimental findings, which were based on actual traffic data, indicate that the approach has some potential applications. Dong et al. [8] proposed a semi-supervised Double Deep Q-Network (SSDDQN)-based optimization method for network abnormal traffic detection, it has lower model complexity, but it has almost no detection ability to detect the lowest number of U2R abnormal attack traffic. However, both semisupervised and unsupervised methods have the disadvantage of ignoring the correlation between features, which results in nonoptimal analysis.

In the supervised approach, to increase the precision of intrusion detection, Li et al. [9] developed the density peak nearest neighbor (DPNN) classifier based on the fundamental concepts of k-nearest neighbor (KNN) and density peak clustering (DPC). According to experimental findings, DPNN performs better than other classifiers in terms of average precision and effectiveness; however, it is unable to recognize U2R assaults. Xue et al. [10] demonstrated that KNN as an evaluation function can effectively deal with the difficult feature selection problem, and the algorithm is superior in terms of classification accuracy over other algorithms but is not applicable to the multiobjective feature domain. Gu et al. [11] employed support vector machines (SVMs) to construct intrusion detection models that are sturdy in terms of precision, training velocity, and false alarm frequency. Though, the

writers of this manuscript only probed the binary categorization instance of the intrusion detection dilemma. In addition, Kabir et al. [12] proposed an optimal assignment-based least squares support vector machine (OA-LSSVM), which can be used for both static and incremental data but has low accuracy in identifying intrusion traffic for the U2R category.

Both unsupervised and semisupervised methods suffer from the disadvantage of ignoring potential associations between features, resulting in an analysis that is not the optimal set. However, this can be crucial when dealing with traffic analysis, as we have to consider not only statistical-based features but also deterministic features. Alternatively, supervised methods can provide optimal results when the data is correctly marked. Additionally, intrusion detection usually occurs in controlled networks, where the type of traffic data passing through can be automatically marked with the help of a network analyzer. On the other hand, the dataset used for most intrusion detection model tests is the old KDDcup99 dataset.

Therefore, to solve the above problems, this paper proposes for the first time a new dimensionality reduction method and a bidirectional long and short-term memory model with multilayer convolution. In the dimensionality reduction method, first, the numerical and normalized dataset is dimensioned down using PCA, after which Hotelling's  $T^2$  test is used to further obtain more representative data, and then the data are input for training into the model presented in this paper. To evaluate the model proposed in this paper, the CIC-IDS2017 dataset is used, which contains benign and up-to-date common attacks, similar to real-world data (PCAPs). Compared to the literature, the main contributions of this paper are as follows:

- (1) We proposed a feature extraction model that combines PCA and Hotelling's  $T^2$ , which can select features that are more representative with better category differentiation and stronger correlation with category labels.
- (2) We proposed an improved algorithm for BiLSTM modeling. A two-layer convolutional network is used, and L2 regularization is used to effectively prevent overfitting of the model, followed by a flattened layer that flattens the two-dimensional tensor of the LSTM output into a one-dimensional vector for the fully connected layer. The detection accuracy is effectively improved.

The rest of the paper is organized as follows: [Section 2](#) describes the work related to this paper, [Section 3](#) focuses on the methodology used in this paper and the structure of the model, [Section 4](#) focuses on the data preprocessing and the results of the data after feature extraction, [Section 5](#) discusses the experimental results and the results of evaluating the model proposed in this paper, and finally, in [Section 6](#), it summarizes the work of this paper and gives the direction of the possible improvements.

## 2 Related Work

Dimensionality reduction-based anomaly detection methods are usually based on the assumption of intercorrelation of normal data features and attempt to find the best description of the normal data subspace and then project the data into that best subspace, and those with large reconstruction errors will be identified as anomalous traffic [13]. Chen et al. [14] introduced PCA-based dimensionality reduction, which was verified to be useful in network intrusion detection, but PCA dimensionality reduction is a linear transformation that often fails to capture the nonlinear correlation between its features when confronted with complex and redundant network traffic data. Therefore, this paper uses

a combination of PCA and Hotelling's T<sup>2</sup> for feature extraction, and the extracted features have better discrimination.

Liu et al. [15] proposed the use of an independent component analysis (ICA) algorithm to extract features and eliminate redundant features so that the model has better feature learning ability and more accurate classification ability. Binbusayyis et al. [16] presented an unsupervised deep learning methodology for intrusion detection, integrating autoencoders (1D CAE) and a class of support vector machines (OCSVM) as classifiers in an IDS for the first time. Su et al. [17] proposed a traffic anomaly detection model called BAT. The BAT model combines BiLSTM and an attention mechanism to improve anomaly detection effectively. Nirmala et al. [18] proposed an intrusion detection method based on stacked expansive Convolutional Neural Networks (CNNs), and the proposed method can self-sufficiently learn the important features of a large amount of unlabeled raw traffic data.

Alsulami et al. [19] introduced an automatic labeling algorithm founded on self-augmentation and a consolidated categorization approach that can accurately and efficiently perceive and categorize network traffic assaults, but its running time is long. Saeed et al. [20] proposed a hybrid model for real-time group-based network intrusion detection for streaming data. They evaluated it on the NSL-KDD and KDDcup99 datasets and finally identified 16 features that have significant contributions to anomaly detection. Sabeel et al. [21] proposed a defense AI engine combining dual feature selection techniques and hyperparameter optimization of AI models to perform binary attack stream identification using the proposed system and trained and validated the AI models on the CIC-IDS2017 dataset. Zhang et al. [22] studied the improved LSTM intrusion detection algorithm model, used the particle swarm optimization algorithm to select features to reduce the feature dimension. They finally validated it on the KDDcup99 dataset. The obtained results showed that the improved LSTM network can effectively improve the accuracy and F1 value of intrusion detection. Yang et al. [23] used a deep convolutional generative adversarial network to extract features from the original data and then used an LSTM model to learn features of network intrusion behavior and validated it on the NSL-KDD dataset, achieving 99% accuracy. Halbouni et al. [24] constructed a hybrid model of CNN and LSTM, and the experimental results showed that the model has a high detection rate and accuracy but can exhibit lower performance in detection rate on unbalanced datasets.

Andresini et al. [25] proposed a hybrid deep learning network based on a self-encoder and LSTM for intrusion detection, which first uses a self-encoder for downscaling and feature extraction of network traffic data and then employs an LSTM network to process the order of network traffic data to achieve data classification. Although the above methods have achieved some results in abnormal traffic classification, they do not effectively utilize the difference information between abnormal and normal samples, resulting in low accuracy in recognizing abnormal data.

The research work in this paper differs from the above in that:

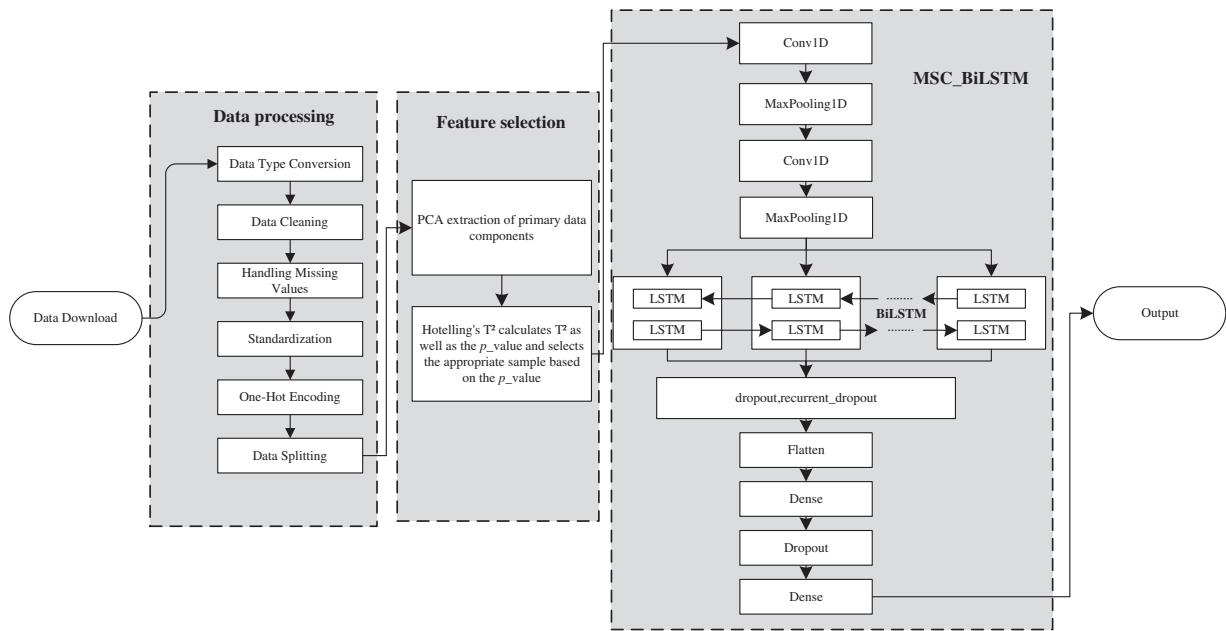
- (1) PCA is combined with Hotelling's T<sup>2</sup> for the first time for dimensionality reduction and feature extraction in intrusion traffic detection, which can select not only more representative features but also features with better differentiation.
- (2) We use an improved BiLSTM, which can make good use of the spatial features as well as temporal features between data and better exploit the different relationships between abnormal and normal samples.

### 3 Methods

#### 3.1 Overall Framework

**Fig. 1** depicts the main flow chart showing the process steps of the algorithm used in this paper. It contains four main sections:

- (1) First, the data part is filtered, and the dataset is divided into a training set and a test set.
- (2) Second, the most representative features are extracted from the high-dimensional data, replacing the initial high-dimensional features using a combination of PCA and Hotelling's  $T^2$ .
- (3) Then, the improved BiLSTM model is used to accurately classify the dataset after feature extraction.
- (4) Finally, the final classification result is obtained based on the output.



**Figure 1:** Flow chart of traffic intrusion detection

#### 3.2 Feature Extraction Based on PCA with Hotelling's $T^2$

There are various methods for extracting features, and this step is critical in classification problems because feature extraction eliminates redundant and irrelevant input features, improving the accuracy of the classifier while reducing the computation time for learning [26]. Principal component analysis is often used on data from diverse fields such as biology, medicine, psychology, chemistry, materials science, computer science, and deep learning [27]. One of the most iconic applications of PCA in the field of computer vision is face recognition, which in general data analysis can be considered a preprocessing step applied to the data before more sophisticated classification or learning methods are employed [28]. In such cases, PCA can be regarded as a feature selection or feature extraction process. The utilization of this method can reduce the size of the input data before employing more complex classification algorithms and can significantly reduce the computational cost of the method while minimizing the loss of accuracy [29].

Hotelling's  $T^2$  is one of the multivariate control charts used to detect multivariate outliers [30]. Hotelling's  $T^2$  index can be applied to identify normal variability in principal components (PCs). Hotelling's  $T^2$  measures the variation in the PCs and is calculated as shown in Eq. (1):

$$T^2 = x^T P \Lambda^{-1} P^T x \quad (1)$$

where  $x$  is the data vector,  $P$  is the transformation matrix obtained by PCA,  $\Lambda$  is the eigenvalue diagonal matrix,  $(-1)$  denotes the inverse matrix, and  $\hat{T}$  denotes the transpose of the matrix.

In the case of normal data obeying a multivariate normal distribution,  $T^2$  is related to the distribution of the statistic  $F$ , which can be expressed by Eq. (2):

$$\frac{N - l}{l(N - 1)} T^2 \sim F_{1,N-l} \quad (2)$$

where  $N$  is the number of data points and  $l$  is the dimensionality of the data. If the number  $N$  of data points is large, the  $T^2$  index can be well approximated by a  $\chi^2$  cardinal distribution with  $l$  degrees of freedom, i.e.,  $T^2 \leq \chi_l^2$ .

After feature extraction using PCA, the principal components extracted by PCA can be used as input variables for Hotelling's  $T^2$ . First, the number of samples (`n_samples`) and the number of features (`n_features`) of the training data are obtained, and the mean and covariance matrix (`cov`) of the training data are calculated. Hotelling's  $T^2$  statistic and the corresponding  $p_{values}$  are calculated by iterating through each sample in a loop.

Hotelling's  $T^2$  statistic ( $T^2$ ) was calculated as shown in Eq. (3):

$$T^2 = n_{features} * (X_{pca}[i] - m) \cdot C^{-1} \cdot (X_{pca}[i] - m)^T \quad (3)$$

where  $X_{pca}[i]$  is the PCA-transformed feature vector of the  $i$ th sample,  $m$  is the mean vector of the training data, and  $C$  is the covariance matrix of the training data.

Next,  $p_{values}$  were calculated using the cumulative distribution function (CDF) of the chi-square distribution based on Hotelling's  $T^2$  statistic and degrees of freedom (number of features).  $p_{values}$  indicate the probability that the observed statistic ( $T^2$ ) is greater than or equal to the observed statistic under the null hypothesis (that the sample data conform to a normal distribution). The calculation formula is shown in Eq. (4):

$$p_{values}[i] = 1 - \text{chi2.cdf}(T^2[i], n_{features}) \quad (4)$$

where `chi2.cdf()` is the cumulative distribution function of the chi-square distribution,  $T^2[i]$  is Hotelling's  $T^2$  statistic for the  $i$ th sample, and  $n_{features}$  is the number of features.

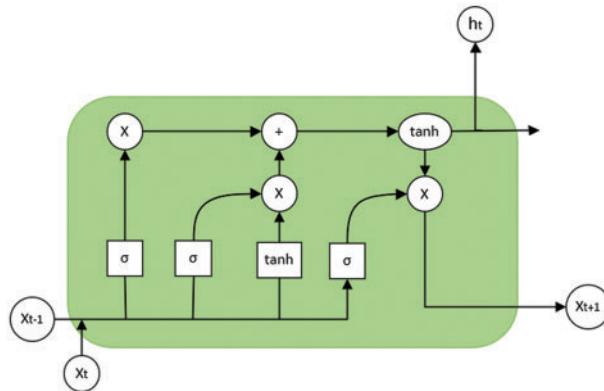
Finally, according to the threshold of  $p_{values}$ , the samples with  $p_{values}$  less than the threshold are selected as the training data.

Overall, the feature extraction model, combining PCA and Hotelling's  $T^2$ , initially employs PCA to extract the main components of the data. The remaining principal components may contain noise or irrelevant information. The aim of Hotelling's  $T^2$  is to detect observations in the principal component space that deviate from the normal data distribution. Hotelling's  $T^2$  statistic and the chi-square distribution were used to assess the degree of outliers of each sample in the feature space by calculating  $p_{values}$  to determine whether the samples were significantly different from the normal samples. Based on the results of  $p_{values}$ , the samples that are significantly different are filtered out, thus improving the

performance and generalization of the model. Detecting outliers improves the quality of the dataset and reduces sensitivity to noise, which improves subsequent analysis or modeling.

### 3.3 Classification Based on MSC\_BiLSTM

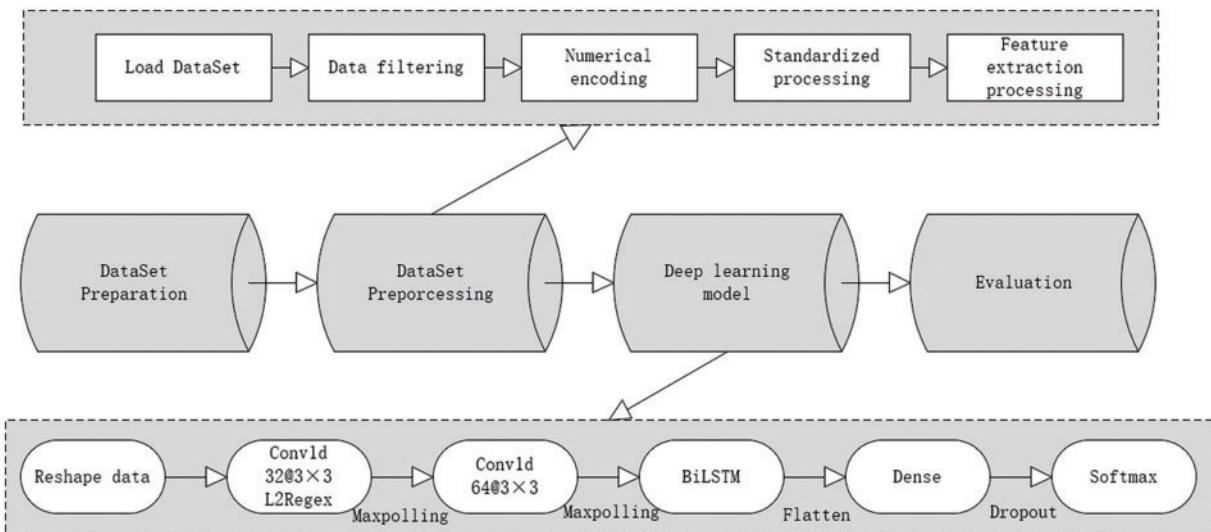
The basic unit of LSTM is the storage unit, which is a linear module with constant weight recursive connections. Multiplicative ingress and egress gate components modulate a steady error flow, ensuring that disturbances induced by irrelevant inputs or storage contents do not sway them. The ingress and egress gates ascertain which errors they must rescale or obtain. Employing conventional techniques, the error signal tends to evaporate because errors are gauged founded on the magnitude of the weights. Ingress and egress gates configure a network, and the network's architecture is established by the operator. Gates can glean entries from other units and utilize these entries to conclude whether to retain information or not [31]. Fig. 2 shows the most typical LSTM unit.



**Figure 2:** LSTM base unit

Therefore, we propose a method called the MSC\_BiLSTM new model for processing sequential data. In contrast to the BiLSTM network, this model contains a convolutional neural network (Conv1D) and BiLSTM layers that learn the dependencies between different time steps in the sequential data to better capture the long-term dependencies. Additionally, to avoid model overfitting, we employ the L2 regularization technique and a dropout layer in the model. L2 regularization penalizes the weight parameters in the model, thereby preventing the model from being overly dependent on specific weight values. The dropout layer randomly discards the outputs of some neurons, preventing overfitting.

To convert the multidimensional input data into one-dimensional vectors, a flattened layer is also included in the model so that the multidimensional input data can be converted into one-dimensional vectors for fully connected operations in the dense layer. The last layer uses the softmax activation function, which transform the model's output into a probability distribution of the categories for multi-categorization tasks. With the combination of these layers, the MSC\_BiLSTM model is better equipped to handle sequence data, thus enhancing the model's performance and generalization ability. The framework diagram of the proposed model is shown in Fig. 3.



**Figure 3:** Overall framework of model

The improved modeling algorithm is shown in [Table 1](#).

**Table 1:** Improved BiLSTM processes

---

#### Algorithm 1 Improved BiLSTM

---

**Input** X\_train\_selected, y\_train\_selected

**Step 1** Reshape the input data to (n\_components,1);

**Step 2** Use  $32 \times 3 \times 3$  convolution kernels to extract features of the input sequence, use the ReLU activation function for nonlinear transformation, and use L2 regularization to prevent overfitting

**Step 3** Pooling of convolutional layers on the output

**Step 4** Use  $64 \times 3 \times 3$  convolution kernels are used to further extract features

**Step 5** Same as the Step 3

**Step 6** Use BiLSTM networks to capture long-term dependencies in sequences, then dropout and loop dropout to prevent overfitting.

**Step 7** Compresses the input sequence into a one-dimensional vector

**Step 8** A fully connected layer of 128 neurons was used to further extract features

**Step 9** Dropout operations are performed on the output of the fully connected layer to reduce overfitting

**Step 10** Use the Dense layer of the SoftMax activation function to output the probability value for each category

---

**Output** num\_classes

---

The main flow of the MSC\_BiLSTM model is shown in [Table 1](#), as well as the parameter settings for each layer after adjusting the model in [Table 2](#). It employs two layers of convolutional structure and one layer of bi-directional LSTM structure. This design allows the model to learn both deep spatial features through the convolutional layers and better temporal features through the bi-directional

LSTM layer. From [Table 2](#), we can see that the input dimension is (128, 22, 1), where 128 means that each batch contains 128 samples, each sample has 22 features and each feature has 1 channel. The convolution layer parameter is given as (32, 3, 3) where 32 filters are present in layer 1 with size (3, 3). In addition, the first layer incorporates L2 regularization, which can be a good solution to the overfitting problem. In the following sections, we will evaluate the performance of this classifier on the CIC-IDS 2017 dataset.

**Table 2:** MSC\_BiLSTM model parameter settings

| Layer | Name                 | Parameters | Dimensions    |
|-------|----------------------|------------|---------------|
| 0     | Input                | —          | (128, 22, 1)  |
| 1     | Convolution          | (32, 3, 3) | (128, 20, 32) |
|       | Pooling              | (2, 2)     | (128, 10, 32) |
|       | Activation (relu)    | —          | —             |
|       | Regularizer (L2)     | 0.001      | —             |
| 2     | Convolution          | (64, 3, 3) | (128, 8, 64)  |
|       | Pooling              | (2, 2)     | (128, 4, 64)  |
|       | Activation (relu)    | —          | —             |
| 3     | BiLSTM               | 64         | (128, 4, 128) |
|       | Dropout              | 0.2        | —             |
|       | Recent_dropout       | 0.2        | —             |
| 4     | Flatten              | —          | (128, 512)    |
| 5     | Dense                | 128        | (128, 128)    |
|       | Activation (relu)    | —          | —             |
| 6     | Dropout              | 0.2        | —             |
| 7     | Dense                | 15         | (128, 15)     |
|       | Activation (softmax) | —          | —             |

## 4 Data Preprocessing and Feature Extraction

In this section, the dataset used is presented along with the preprocessing of the data and the main features extracted.

### 4.1 Data Filtering

There are many datasets used for traffic intrusion detection systems, such as KDDcup99, NSL-KDD, and CIC-IDS2017. However, based on the evaluation of existing datasets, it has been shown that the majority of the datasets are obsolete and undependable. Certain of these datasets are deficient in traffic diversity and volume, while others do not encompass a variety of known attacks, and some lack feature sets and metadata [32]. To overcome these shortcomings and meet the needs of modern attacks and network traffic, Sharafaldin et al. [33] at the Canadian Institute for Cybersecurity developed a dataset, CIC-IDS2017, which contains benign and up-to-date common attacks, similar to real-world data (PCAPs).

The dataset contains 80 multidimensional features, plus a total of 15 categories of normal traffic. Traffic data are collected from Monday to Friday. Monday is all normal traffic. Tuesday to Friday includes normal traffic and various types of intrusion traffic such as brute force File Transfer Protocol (FTP), brute force Secure Shell Protocol (SSH), Disk Operating System (DoS), Heartbleed, Web attacks, infiltration, botnets, and Distributed Denial of Service (DDoS). The details are shown in [Table 3](#).

**Table 3:** CIC-IDS2017 dataset

|   | Description      | Amount  |    | Description              | Amount |
|---|------------------|---------|----|--------------------------|--------|
| 0 | BENIGN           | 2273097 | 8  | Heartbleed               | 11     |
| 1 | Bot              | 1966    | 9  | Infiltration             | 36     |
| 2 | DDoS             | 128027  | 10 | PortScan                 | 158930 |
| 3 | DoS GoldenEye    | 10293   | 11 | SSH-Patator              | 5897   |
| 4 | DoS Hulk         | 231073  | 12 | Web Attack-Brute Force   | 1507   |
| 5 | DoS Slowhttptest | 5499    | 13 | Web Attack-Sql Injection | 21     |
| 6 | DoS slowloris    | 5796    | 14 | Web Attack-XSS           | 652    |
| 7 | FTP-Patator      | 7983    |    |                          |        |

## 4.2 Data Preprocessing

Data preprocessing, which includes encoding, standardization and normalization of discontinuous variables, is an important stage in determining classification performance. Although this stage plays an important role in preprocessing, little attention has been given to it [\[34\]](#). As stated earlier, the CIC-IDS2017-based dataset comprises over 80 features, but some of these features play a minimal role in classification. Therefore, in addition to numerical processing the nonnumerical data in the dataset, as well as normalization and standardization it, this paper also performs feature dimensionality reduction and extraction on the dataset.

### 4.2.1 Numerical Encoding

For the numerical processing of data, this paper uses the label encoder coding technique, which converts categorical variables into numerical variables. It maps each different categorical variable label to an integer value. Specifically, for each categorical variable label  $l$ , the label encoder maps it to an integer value  $v$ . This mapping can be represented as  $v = f(l)$ , where  $f$  is a function that maps categorical variable labels to integer values. A dictionary or hash table can be used to implement this function.

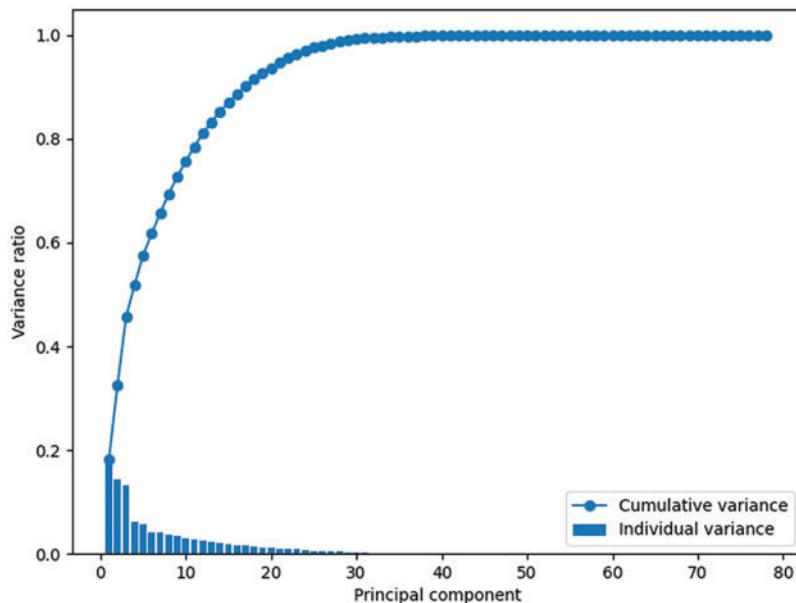
This approach enables the conversion of categorical variable labels into corresponding integer values through the label encoder, facilitating the processing of these categorical variables by machine learning algorithms. It should be noted that when using label encoder, care should be taken not to interpret the integer values of the categorical variables as having any natural mathematical meaning, as these values are simply a way of encoding.

#### 4.2.2 Standardized Processing

Prior to the normalization process, it is necessary to handle infinite values and null values within the dataset. The approach employed for addressing infinite and null values involves mean replacement. For standardization, the `StandardScaler` function is utilized. This technique performs feature scaling, normalizing each attribute in the data to follow a standard Gaussian distribution with a mean of 0 and a standard deviation of 1. Specifically, for each feature, `StandardScaler` calculates their mean and standard deviation, then subtracts the feature values from their mean and divides them by their standard deviation, ensuring that the values of each feature conform to a standard normal distribution.

#### 4.3 Feature Extraction Processing

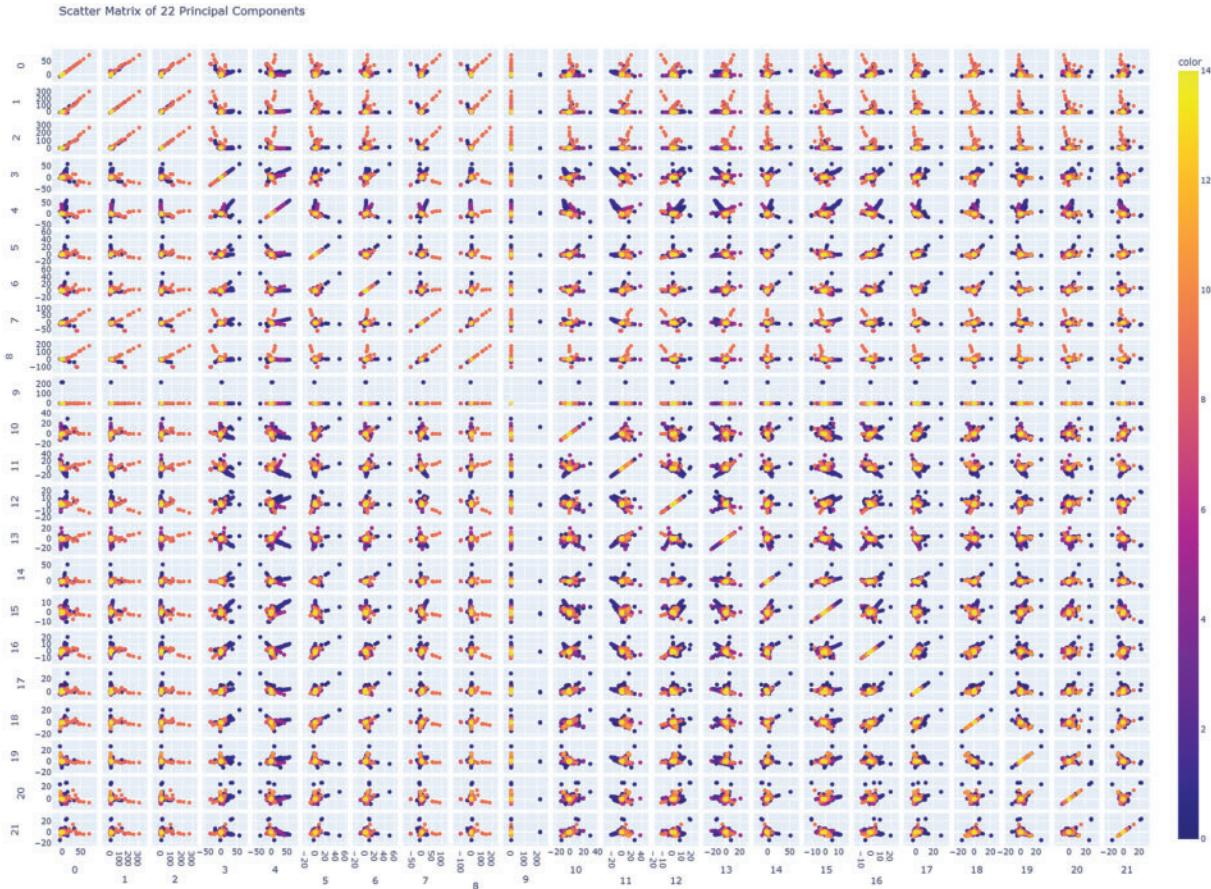
The above processed dataset was subjected to feature extraction and discrimination using PCA and Hotelling's  $T^2$ . The data are first processed using PCA, i.e., the number of dimensions  $d$  of the data is obtained first, and the mean vector of each dimension is calculated. After that, the covariance matrix as well as the eigenvectors and eigenvalues are calculated for the whole dataset. Finally, by selecting the first  $n$  larger eigenvalues, a  $d * n$  matrix is obtained, and the new matrix obtained is the new sample space. Then, Hotelling's  $T^2$  mentioned in [Section 2](#) is used to discriminate, and finally, the  $K$ -dimensional dataset is derived. In this paper, for the CIC-IDS2017 dataset, the features whose principal component contribution ratio is 95% are extracted. After processing, the dataset is downscaled to 22 dimensions from the original 80+ dimensional data, with the principal component ratio of each dimension shown in [Fig. 4](#).



**Figure 4:** Number of components accounting for 95% of variance: 22

The relationship between each principal component is shown in [Fig. 5](#), where each cell shows a scatterplot between the two principal components and a histogram or density plot of each principal component is shown on the diagonal. This chart has 22 rows and 22 columns, with each row and column representing a principal component. In each cell, you can see the distribution of data points on the plane corresponding to the two principal components. On the diagonal, one can see the distribution

of data for each principal component itself. From Fig. 5, we can clearly see the range of the main distribution of each principal component, as well as the linear and proportional relationship between different principal components.



**Figure 5:** Plot of relationships between principal components

## 5 Experimental Results and Discussion

The classification method employed in this paper utilizes the proposed MSC\_BiLSTM model. The preprocessed dataset is randomly extracted at a ratio of 80% for the training set and 20% for the test set. In addition, the confusion matrix is utilized to gauge the effectiveness of the intrusion detection system, and the accuracy and error rate measures the accuracy of the intrusion classification. The accuracy metrics in network anomaly detection are classified into two categories: True Positive (TP) signifies the quantity of examples in which the genuine positives are correctly positive and anticipated as positive; and True Negative (TN) denotes the number of instances where the actual negatives are accurately forecasted as negative and the number of specimens anticipated to be negative. Additionally, false detections are categorized into two categories: False Positive (FP) denotes the quantity of instances that are truly negative but forecasted as positive, while False Negative (FN) signifies the number of examples that are genuinely positive but anticipated to be negative.

There are two ways to algorithmize the accuracy rate in multiclassification tasks. One is the macro average, which signifies that each class possesses identical weight when computing the mean, resulting in the arithmetic average of the metrics for each class. The other is the micro average, which involves computing multiclassification indicators by allocating identical significance to each exemplar across all classes and integrating all specimens together to calculate each metric. If the number of examples in each class is approximately equivalent, then there is negligible discrepancy between macro averaging and micro averaging. However, if the number of illustrations in each class diverges substantially, micro averaging is used when focusing on classes with an abundant quantity of exemplars, while macro averaging is used when focusing on classes with a limited number of specimens.

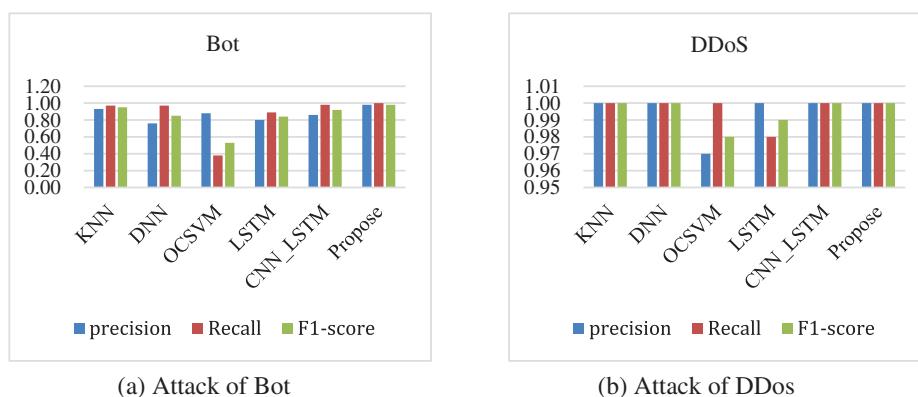
Recall denotes the ratio of the genuine positive specimens anticipated as positive to the total authentic positive examples. It mirrors the model's capability to identify positive specimens; the superior the recall, the stronger the model's aptitude to recognize positive examples. The F1-score can be construed as the harmonized mean of precision and recall, constituting an amalgamation of precision and recall; the superior the F1-score, the more robust the model.

### 5.1 Comparative Analysis of Experiments

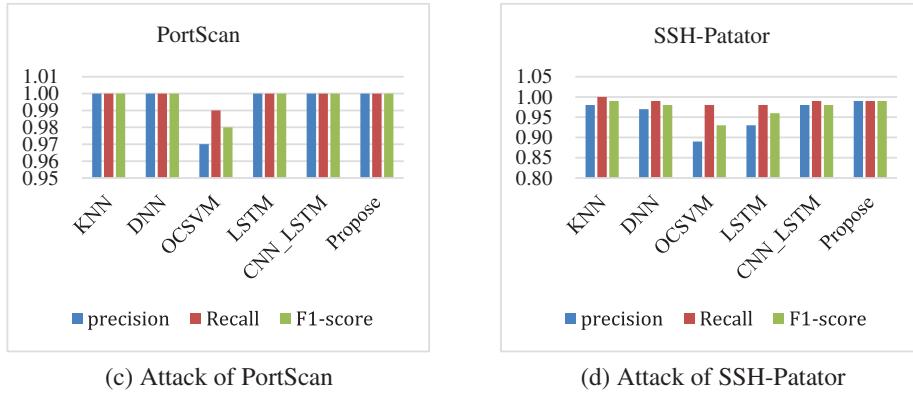
All experiments were performed on a Windows 10 PC with an Intel (R) Core (TM) i5-9400 CPU @ 2.90 GHz and 16.0 GB of memory. The deep learning model uses the TensorFlow, pandas, and Keras libraries. We have evaluated our model using fine-grained classification, as shown in [Table 2](#). The normal traffic is BENIGN, and the remainder is intrusion traffic.

In this section, the OCSVM [16], K-Nearest Neighbor (KNN) [20], Deep Neural Networks (DNN) [21], LSTM [23], and CNN\_LSTM [24] models are used in our comparison experiments.

[Fig. 6](#) gives the identification results of each model for several common intrusion traffic types including Bot, DDoS, PortScan, and SSH. [Fig. 6a](#) shows the comparison graph of the detection results of different models for Bot, in which we can see that the model in this paper achieves a certain advantage in terms of precision, recall, and F1-score, while the OCSVM performs the worst. From [Fig. 6b](#), we can see that KNN, DNN, CNN\_LSTM and the model in this paper achieve better recognition results in the detection of DDoS attacks. [Fig. 6c](#) shows the detection results for PortScan, and it can be seen that only OCSVM is slightly less effective in detecting it. [Fig. 6d](#) is a comparison chart of the results of detecting SSH, and it can be seen that the model in this paper is more stable compared with other models.

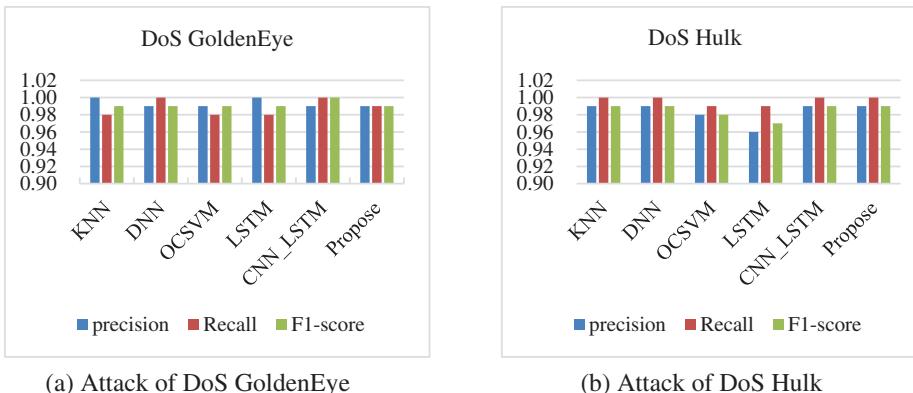


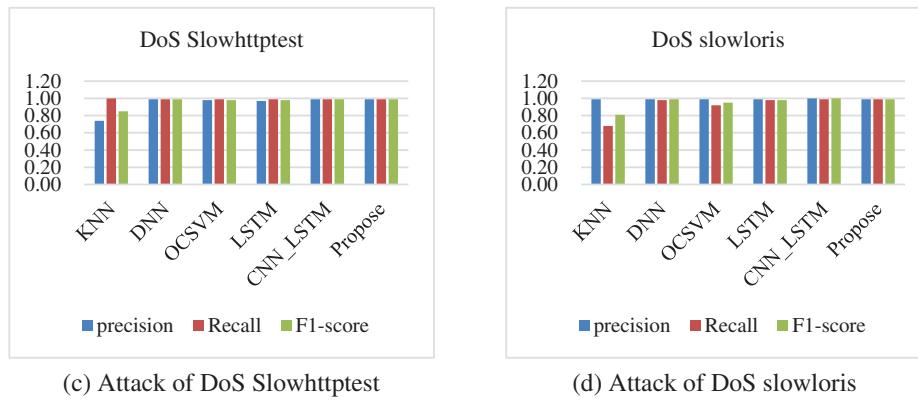
**Figure 6:** (Continued)

**Figure 6:** Different intrusion traffic detection results

**Fig. 7** shows the detection results of different models for different kinds of intrusion traffic of DoS. **Fig. 7a** shows the graph of detection results for DoS GoldenEye, and we can see that the model proposed in this paper is more balanced compared to other models. **Fig. 7b** shows the graph of the recognition results for DoS Hulk, and it can be seen that except for the LSTM model, the precision, recall, and F1-score of the other models is all above 0.98. For the identification and detection of DoS Slowhttptest (**Fig. 7c**) and DoS slowloris (**Fig. 7d**), it can be seen that only KNN's detection results are slightly lower than the other models, i.e., the models in this paper, as well as DNN, OCSVM, LSTM, and CNN\_LSTM, have an accuracy of more than 0.99 for the identification of these two types of anomalous traffic.

In **Fig. 8**, we can see the detection results of different models for web attacks and FTP attacks. In **Fig. 8a**, we can see that the detection of brute force in web attacks has considerable detection results, in which the model in this paper reaches 0.96 in recall, but in **Figs. 8b** and **8c**, we can see that the detection of SQL injection and XSS in web attacks is not very effective, as OCSVM, LSTM, and CNN\_LSTM cannot recognize these two types of anomalous traffic at all, and our proposed model can detect them, but the detection effect is also not satisfactory. In **Fig. 8d**, we can see the detection results of different models for the anomalous traffic of the FTP type, and we can see that except for the LSTM model with a slightly lower detection recall and F1-score, all the other models can achieve better results, i.e., the detection accuracies are all above 0.99.

**Figure 7:** (Continued)

**Figure 7:** Comparison of the detection results of each model for various types of DoS attacks**Figure 8:** Comparison of the detection results of each model for various types of web attacks and FTP attacks

The identification results of different models for each class of traffic accuracy are shown in [Table 4](#). The numbers 0–14 in [Table 4](#) correspond to each traffic class in [Table 2](#). That is, 0: BENIGN, 1: Bot, 2: DDoS, 3: DoS GoldenEye, 4: DoS Hulk, 5: DoS Slowhttptest, 6: DoS slowloris, 7: FTP-Patator, 8: Heartbleed, 9: Infiltration, 10: PortScan, 11: SSH-Patator, 12: Web Attack-Brute Force, 13: Web

Attack-Sql Injection, 14: Web Attack-XSS. In [Table 4](#), we can discern that the proposed model in this manuscript surpasses the other models in the majority of the traffic categories, and there is still some traffic whose recognition accuracy is lower than other models, but for each category of traffic recognition the accuracy is above 99%.

**Table 4:** Comparison of the accuracy of different models

|    | KNN    | DNN           | OCSVM  | LSTM          | CNN_LSTM      | Propose       |
|----|--------|---------------|--------|---------------|---------------|---------------|
| 0  | 0.9025 | <b>0.9957</b> | 0.8704 | 0.9952        | 0.9926        | 0.9910        |
| 1  | 0.9925 | <b>0.9987</b> | 0.9893 | 0.9983        | 0.9967        | 0.9960        |
| 2  | 0.9742 | 0.9996        | 0.9809 | 0.9998        | 0.9995        | <b>0.9998</b> |
| 3  | 0.9706 | 0.9990        | 0.6821 | 0.9993        | 0.9990        | <b>0.9996</b> |
| 4  | 0.9644 | 0.9990        | 0.8887 | 0.9982        | 0.9982        | <b>0.9990</b> |
| 5  | 0.9557 | 0.9988        | 0.9837 | 0.9991        | 0.9985        | <b>0.9992</b> |
| 6  | 0.9487 | 0.9989        | 0.9806 | <b>0.9995</b> | 0.9991        | 0.9990        |
| 7  | 0.9781 | 0.9992        | 0.9408 | 0.9997        | <b>0.9997</b> | 0.9993        |
| 8  | 0.9999 | 1.0000        | 1.0000 | 1.0000        | 1.0000        | 1.0000        |
| 9  | 0.9995 | 0.9998        | 0.9996 | 0.9999        | 0.9998        | <b>0.9998</b> |
| 10 | 0.9970 | 0.9994        | 0.9958 | <b>0.9997</b> | 0.9996        | 0.9855        |
| 11 | 0.9482 | <b>0.9986</b> | 0.9505 | 0.9986        | 0.9984        | 0.9920        |
| 12 | 0.9637 | 0.9936        | 0.9905 | 0.9931        | 0.9931        | <b>0.9940</b> |
| 13 | 0.9997 | 0.9998        | 0.9997 | 0.9996        | 0.9998        | <b>0.9998</b> |
| 14 | 0.9942 | 0.9949        | 0.9946 | 0.9944        | 0.9944        | <b>0.9953</b> |

Finally, in [Table 5](#), we give the average results of the evaluation metrics for all the model test results. From [Table 5](#), we can clearly see the advantages of this paper's model compared to other models. The precision is improved by approximately 3%–16% compared to the other models, the recall is improved by 1%–10%, and the F1-score is improved by approximately 4%–13%, which shows that the model proposed in this paper is more stable and applicable for the identification of abnormal traffic.

**Table 5:** Comparison of evaluation indicators of the overall model

|           | KNN    | DNN    | OCSVM  | LSTM   | CNN_LSTM | Propose       |
|-----------|--------|--------|--------|--------|----------|---------------|
| Accuracy  | 0.9720 | 0.9829 | 0.9545 | 0.9531 | 0.9855   | <b>0.9871</b> |
| Precision | 0.8705 | 0.9276 | 0.7940 | 0.8788 | 0.8966   | <b>0.9597</b> |
| Recall    | 0.8153 | 0.8468 | 0.7543 | 0.8286 | 0.8279   | <b>0.8513</b> |
| F1-score  | 0.8420 | 0.8853 | 0.7736 | 0.8529 | 0.8599   | <b>0.9022</b> |

## 5.2 Ablation Experiment Analysis

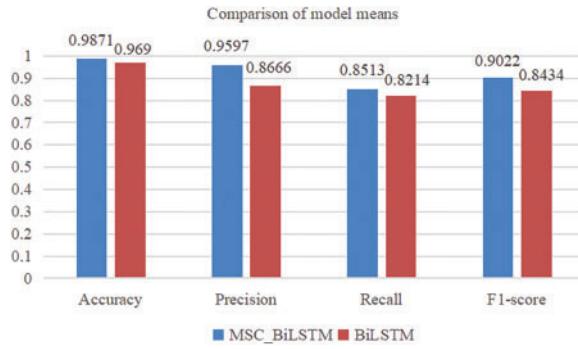
In addition, we performed ablation experiments to compare the improved model in this paper with the unimproved BiLSTM, and the comparison results are shown in [Table 6](#).

**Table 6:** Comparison of ablation experiments

|    | Accuracy       |        | Precision      |             | Recall         |             | F1-score       |             |
|----|----------------|--------|----------------|-------------|----------------|-------------|----------------|-------------|
|    | MSC_BiL<br>STM | BiLSTM | MSC_BiL<br>STM | BiLSTM      | MSC_BiL<br>STM | BiLSTM      | MSC_BiL<br>STM | BiLSTM      |
| 0  | <b>0.9910</b>  | 0.9529 | <b>0.99</b>    | 0.94        | <b>0.97</b>    | 0.93        | <b>0.98</b>    | 0.94        |
| 1  | <b>0.9960</b>  | 0.9923 | <b>0.98</b>    | 0.76        | <b>1.00</b>    | 0.76        | <b>0.98</b>    | 0.76        |
| 2  | <b>0.9998</b>  | 0.9954 | <b>1.00</b>    | 0.97        | 1.00           | 1.00        | <b>1.00</b>    | 0.99        |
| 3  | <b>0.9996</b>  | 0.9992 | 0.99           | 0.99        | 0.99           | <b>1.00</b> | 0.99           | <b>1.00</b> |
| 4  | <b>0.9990</b>  | 0.9982 | 0.99           | 0.99        | <b>1.00</b>    | 0.99        | 0.99           | 0.99        |
| 5  | <b>0.9992</b>  | 0.9987 | 0.99           | 0.99        | 0.99           | 0.99        | 0.99           | 0.99        |
| 6  | <b>0.9990</b>  | 0.9985 | 0.99           | 0.99        | <b>0.99</b>    | 0.98        | 0.99           | 0.99        |
| 7  | <b>0.9993</b>  | 0.9985 | <b>1.00</b>    | 0.99        | 0.99           | 0.99        | 0.99           | 0.99        |
| 8  | 1.0000         | 1.0000 | 1.00           | 1.00        | 1.00           | 1.00        | 1.00           | 1.00        |
| 9  | <b>0.9998</b>  | 0.9995 | 0.83           | <b>1.00</b> | 0.83           | 0.78        | 0.83           | 0.88        |
| 10 | 0.9955         | 0.9673 | 1.00           | 1.00        | <b>1.00</b>    | 0.97        | <b>1.00</b>    | 0.99        |
| 11 | 0.9920         | 0.9979 | <b>0.99</b>    | 0.98        | 0.99           | 0.99        | <b>0.99</b>    | 0.98        |
| 12 | <b>0.9940</b>  | 0.9923 | <b>0.71</b>    | 0.65        | <b>0.96</b>    | 0.92        | <b>0.82</b>    | 0.76        |
| 13 | <b>0.9998</b>  | 0.9998 | <b>0.40</b>    | 0.00        | <b>0.67</b>    | 0.00        | <b>0.50</b>    | 0.00        |
| 14 | <b>0.9953</b>  | 0.9941 | <b>1.00</b>    | 0.75        | <b>0.50</b>    | 0.02        | <b>0.67</b>    | 0.05        |

From [Table 6](#), we can clearly see that for most types of traffic recognition, both in terms of precision and recall, our model outperforms the unimproved BiLSTM model.

[Fig. 9](#) gives a comparison chart of the combined mean values of the evaluation metrics of this paper's model and the BiLSTM model for each class of traffic identification results, and the improvement in the effectiveness of this paper's model can be clearly seen in the chart.

**Figure 9:** Comparison of model means

In [Fig. 9](#), we can see that the accuracy of the model proposed in this paper is 98.71%, which is an improvement of 1.81% over the unimproved BiLSTM model, and the corresponding precision is

95.97%, the recall is 85.13%, and the F1 value is 90.22%, which is an improvement of 9.31%, 2.99%, and 5.88%, respectively.

## 6 Conclusion

In this manuscript, we propose a feature extraction approach that combines PCA and Hotelling's T<sup>2</sup> for intrusion detection of network traffic. The high-dimensional traffic data is first downsampled, of which main features are extracted using the integrated PCA and Hotelling's statistics. These extracted features replace the original data traffic, facilitating subsequent intrusion detection. This method results in better category differentiation and more relevant and representative features for the category labels.

Next, we employ a neural network prototype containing Conv1D and BiLSTM layers to detect and classify the data. The Conv1D layer captures the sequential patterns in the data, while the BiLSTM layer handles the long-term dependencies. To prevent overfitting, we incorporate L2 regularization and dropout layers. In the final stage of the framework, a softmax activation function transforms the model's output into a probability distribution for multi-class classification.

During training, a flattened layer is utilized to convert the multidimensional input data into one-dimensional vectors for fully connected operations in the dense layer. This combination of layers enables the framework to effectively learn information from sequential data and enhance its generalization capability, making it more adept at solving real-world problems.

After applying these processing steps, the intrusion detection results surpass those achieved by existing machine learning models, exhibiting improved precision, recall, and F1-scores. However, there is still room for improvement in detecting specific types of anomalous traffic. The limited data available contributes to the low accuracy in recognizing individual traffic. Therefore, future work could focus on balancing the dataset to improve recognition results for traffic categories with lower accuracy.

**Acknowledgement:** We thank our families and colleagues who provided us with moral support.

**Funding Statement:** This work was supported by Tianshan Talent Training Project-Xinjiang Science and Technology Innovation Team Program (2023TSYCTD).

**Author Contributions:** Study conception and design: Xuecheng Yu; data collection: Xuecheng Yu, Yu Zhang; analysis and interpretation of results: Xuecheng Yu, Mingyang Song, Zhenhong Jia; draft manuscript preparation: Mingyang Song, Yan Huang, Zhenhong Jia. All authors reviewed the results and approved the final version of the manuscript.

**Availability of Data and Materials:** The dataset used in this paper is the public dataset CIC-IDS2017 dataset, accessed as follows: <https://www.unb.ca/cic/datasets/ids-2017.html>.

**Conflicts of Interest:** The authors declare that they have no conflicts of interest to report regarding the present study.

## References

- [1] B. Hallajian, H. Motameni and E. Akbari, "Ensemble feature selection using distance-based supervised and unsupervised methods in binary classification," *Expert Systems with Applications*, vol. 200, no. 3, pp. 116794, 2022.

- [2] J. C. Ang, A. Mirzal, H. Haron and H. N. Abdull Hamed, "Supervised, unsupervised, and semi-supervised feature selection: A review on gene selection," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 13, no. 4, pp. 971–989, 2016.
- [3] F. Salo, A. B. Nassif and A. Essex, "Dimensionality reduction with IG-PCA and ensemble classifier for network intrusion detection," *Computer Networks*, vol. 148, no. 5, pp. 164–175, 2019.
- [4] Y. Pan, F. Sun, Z. Teng, J. White, D. C. Schmidt *et al.*, "Detecting web attacks with end-to-end deep learning," *Journal of Internet Services and Applications*, vol. 10, no. 6, pp. 1–22, 2019.
- [5] H. Yao, D. Fu, P. Zhang, M. Li and Y. Liu, "MSML: A novel multi-level semi-supervised machine learning framework for intrusion detection system," *IEEE Internet of Things Journal*, vol. 6, no. 7, pp. 1949–1959, 2018.
- [6] S. Mohammadi, H. Mirvaziri, M. Ghazizadeh-Ahsaee and H. Karimipour, "Cyber intrusion detection by combined feature selection algorithm," *Journal of Information Security and Applications*, vol. 44, no. 8, pp. 80–88, 2019.
- [7] J. Camacho, G. Maciá-Fernández, N. M. Fuentes-García and E. Saccenti, "Semi-supervised multivariate statistical network monitoring for learning security threats," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 9, pp. 2179–2189, 2019.
- [8] S. Dong, Y. Xia and T. Peng, "Network abnormal traffic detection model based on semi-supervised deep reinforcement learning," *IEEE Transactions on Network and Service Management*, vol. 18, no. 10, pp. 4197–4212, 2021.
- [9] L. Li, H. Zhang, H. Peng and Y. Yang, "Nearest neighbors-based density peaks approach to intrusion detection," *Chaos, Solitons & Fractals*, vol. 110, no. 11, pp. 33–40, 2018.
- [10] Y. Xue, W. Jia, X. Zhao and W. Pang, "An evolutionary computation based feature selection method for intrusion detection," *Security and Communication Networks*, vol. 2018, no. 12, pp. 1–10, 2018.
- [11] J. Gu, L. Wang, H. Wang and S. Wang, "A novel approach to intrusion detection using SVM ensemble with feature augmentation," *Computers & Security*, vol. 86, no. 13, pp. 53–62, 2019.
- [12] E. Kabir, J. Hu, H. Wang and G. Zhuo, "A novel statistical technique for intrusion detection systems," *Future Generation Computer Systems*, vol. 79, no. 14, pp. 303–318, 2018.
- [13] S. K. Gautam and H. Om, "Computational neural network regression model for host based intrusion detection system," *Perspectives in Science*, vol. 8, no. 15, pp. 93–95, 2016.
- [14] Z. Chen, C. K. Yeo, B. S. Lee and C. T. Lau, "Autoencoder-based network anomaly detection," in *2018 Wireless Telecommunications Symp. (WTS)*, New York, NY, USA, pp. 1–5, 2018.
- [15] J. H. Liu, S. P. Mao and X. M. Fu, "Intrusion detection model based on ICA algorithm and deep neural network," *Netinfo Security*, vol. 19, no. 17, pp. 1–10, 2019.
- [16] A. Binbusayyis and T. Vaiyapuri, "Unsupervised deep learning approach for network intrusion detection combining convolutional autoencoder and one-class SVM," *Applied Intelligence*, vol. 51, no. 18, pp. 7094–7108, 2021.
- [17] T. Su, H. Sun, J. Zhu, S. Wang and Y. Li, "BAT: Deep learning methods on network intrusion detection using NSL-KDD dataset," *IEEE Access*, vol. 8, no. 19, pp. 29575–29585, 2020.
- [18] P. Nirmala, T. Manimegalai, J. R. Arunkumar, S. Imala, G. Vinoth Rajkumar *et al.*, "A mechanism for detecting the intruder in the network through a stacking dilated CNN model," *Wireless Communications and Mobile Computing*, vol. 2022, no. 20, pp. 1–13, 2022.
- [19] B. Alsulami, A. Almalawi and A. Fahad, "Toward an efficient automatic self-augmentation labeling tool for intrusion detection based on a semi-supervised approach," *Applied Sciences*, vol. 12, no. 21, pp. 7189, 2022.
- [20] M. M. Saeed, "A real-time adaptive network intrusion detection for streaming data: A hybrid approach," *Neural Computing and Applications*, vol. 34, no. 22, pp. 6227–6240, 2022.
- [21] U. Sabeel, S. S. Heydari, K. Elgazzar and K. EL-Khatib, "Building an intrusion detection system to detect atypical cyberattack flows," *IEEE Access*, vol. 9, no. 23, pp. 94352–94370, 2021.
- [22] L. Zhang, H. Yan and Q. Zhu, "An improved LSTM network intrusion detection method," in *2020 IEEE 6th Int. Conf. on Computer and Communications (ICCC)*, Chengdu, China, pp. 1765–1769, 2020.

- [23] J. Yang, T. Li, G. Liang, W. He and Y. Zhao, "A simple recurrent unit model based intrusion detection system with DCGAN," *IEEE Access*, vol. 7, no. 25, pp. 83286–83296, 2019.
- [24] A. Halbouni, T. S. Gunawan, M. H. Habaebi, M. Halbouni, M. Kartiwi *et al.*, "CNN-LSTM: Hybrid deep neural network for network intrusion detection system," *IEEE Access*, vol. 10, no. 26, pp. 99837–99849, 2022.
- [25] G. Andresini, A. Appice, N. Di Mauro, C. Loglisci and D. Malerba, "Exploiting the auto-encoder residual error for intrusion detection," in *2019 IEEE European Symp. on Security and Privacy Workshops (EuroS&PW)*, San Francisco, CA, USA, pp. 281–290, 2019.
- [26] S. Ramakrishnan and S. Devaraju, "Attack's feature selection-based network intrusion detection system using fuzzy control language," *International Journal of Fuzzy Systems*, vol. 19, no. 28, pp. 316–328, 2016.
- [27] F. L. Gewers, G. R. Ferreira, H. F. de Arruda, F. N. Silva, C. H. Comin *et al.*, "Principal component analysis: A natural approach to data exploration," *ACM Computing Surveys*, vol. 54, no. 29, pp. 1–34, 2021.
- [28] S. Sukhvinder, "Poly logarithmic naive bayes intrusion detection system using linear stable PCA feature extraction," *Wireless Personal Communications*, vol. 125, no. 30, pp. 3117–3132, 2022.
- [29] Z. Xia, Y. Chen and C. Xu, "Multiview PCA: A methodology of feature extraction and dimension reduction for high-order data," *IEEE Transactions on Cybernetics*, vol. 52, no. 31, pp. 11068–11080, 2022.
- [30] D. K. Shepherd and S. E. Rigdon, "Are principal components with estimated parameters feasible in multivariate process control?," *Quality and Reliability Engineering International*, vol. 38, no. 32, pp. 4141–4156, 2022.
- [31] C. Avci, B. Tekinerdogan and C. Catal, "Analyzing the performance of long short-term memory architectures for malware detection models," *Concurrency and Computation: Practice and Experience*, vol. 33, no. 33, pp. 1–1, 2023.
- [32] A. Thakkar and R. Lohiya, "A review of the advancement in intrusion detection datasets," *Procedia Computer Science*, vol. 167, no. 34, pp. 636–645, 2020.
- [33] I. Sharafaldin, A. H. Lashkari and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," *Proceedings of the 4th International Conference on Information Systems Security and Privacy*, vol. 1, no. 35, pp. 108–116, 2018.
- [34] T. T. Jui, N. Hoq, S. Majumdar and S. Hossain, "Feature reduction through data preprocessing for intrusion detection in IoT networks," in *2021 Third IEEE Int. Conf. on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, New York, NY, USA, pp. 41–50, 2021.