electronics

MDPI

Article

A Deep Learning Model for Network Intrusion Detection with Imbalanced Data

Yanfang Fu [1], Yishuai Du [1], Zijian Cao [1], Qiang Li [1] and Wei Xiang [2,3,*]

[1] School of Computer Science and Engineering, Xi'an Technological University, Xi'an 710021, China; fuyanfang@xatu.edu.cn (Y.F.); duyishuai@st.xatu.edu.cn (Y.D.); caozijian@xatu.edu.cn (Z.C.); liqiang@st.xatu.edu.cn (Q.L.)
[2] School of Computing, Engineering and Mathematical Sciences, La Trobe University, Melbourne, VIC 3086, Australia
[3] College of Science and Engineering, James Cook University, Cairns, QLD 4878, Australia
* Correspondence: w.xiang@latrobe.edu.au

Abstract: With an increase in the number and types of network attacks, traditional firewalls and data encryption methods can no longer meet the needs of current network security. As a result, intrusion detection systems have been proposed to deal with network threats. The current mainstream intrusion detection algorithms are aided with machine learning but have problems of low detection rates and the need for extensive feature engineering. To address the issue of low detection accuracy, this paper proposes a model for traffic anomaly detection named a deep learning model for network intrusion detection (DLNID), which combines an attention mechanism and the bidirectional long short-term memory (Bi-LSTM) network, first extracting sequence features of data traffic through a convolutional neural network (CNN) network, then reassigning the weights of each channel through the attention mechanism, and finally using Bi-LSTM to learn the network of sequence features. In intrusion detection public data sets, there are serious imbalance data generally. To address data imbalance issues, this paper employs the method of adaptive synthetic sampling (ADASYN) for sample expansion of minority class samples, to eventually form a relatively symmetric dataset, and uses a modified stacked autoencoder for data dimensionality reduction with the objective of enhancing information fusion. DLNID is an end-to-end model, so it does not need to undergo the process of manual feature extraction. After being tested on the public benchmark dataset on network intrusion detection NSL-KDD, experimental results show that the accuracy and F1 score of this model are better than those of other comparison methods, reaching 90.73% and 89.65%, respectively.

Keywords: intrusion detection; Bi-LSTM; attention mechanism; NSL-KDD

Citation: Fu, Y.; Du, Y.; Cao, Z.; Li, Q.; Xiang, W. A Deep Learning Model for Network Intrusion Detection with Imbalanced Data. *Electronics* **2022**, *11*, 898. https://doi.org/10.3390/electronics11060898

Academic Editors: Jihoon Yang and Unsang Park

Received: 12 January 2022
Accepted: 3 March 2022
Published: 14 March 2022

1. Introduction

With the rapid development of computer and communications networks, Internet technology has provided more convenient services to people around the world than ever before. However, the number and types of cyberattacks (such as network viruses, malicious eavesdropping, malicious attacks, etc.), which increase year by year, are creating serious threats to people's information security and property safety. Therefore, information security and communications security has become crucial to both individuals and society as a whole [1,2]. Firewalls are widely deployed and used as basic means of security. However, due to the difficulty of human configuration and the lag for new types of attacks, it is no longer sufficient for units that need high security (e.g., government units, military bases, etc.) [3]. Therefore, network security researchers have proposed a new means to quickly identify and deal with anomalous networks intrusion detection systems (IDSs).

IDS is proven to be one of the efficient and promising approaches. It detects known threats and malicious activities by monitoring traffic data in computer systems, and alerts are issued when these threats are detected [4]. There are two types of monitoring for malicious

activities. One is signature-based detection, similar to antivirus software that requires comparison with previously collected attack features, while the other is anomaly-based detection, which requires comparison with normal traffic to make a judgment. In the KDD99 dataset, Stolfo et al. classified network attacks into four categories—namely, the denial-of-service attack (DoS), user-to-root attack (U2R), remote-to-local attack (R2L), and probe attack [5].

Nowadays, there are many researchers who advocate the combination of intrusion detection and machine learning (ML) technologies for the detection of network attacks by creating effective models. The authors in [6] propose the use of naive Bayes for the identification of anomalous networks and compare it with decision trees (another classical machine learning algorithm). The authors in [7] combine support vector machine (SVM) and the genetic algorithm to optimize the selection, parameters, and weights of SVM features, thus improving the accuracy of network attack identification. The authors in [8] improve the detection by constructing a multi-level random forest model to detect network anomalous behavior. The authors in [9] improve the existing k-nearest neighbor (KNN) classifier by combining K-MEANS clustering and KNN classifier with each other to improve the accuracy of detection. The authors in [10] propose a novel intrusion detection method that first decomposes the network data into smaller subsets by a C4.5 decision tree algorithm and then creates multiple SVM models for the subsets, which reduces the time complexity and improves the detection rate of unknown attacks. However, traditional machine learning methods usually emphasize feature engineering, which consumes considerable computational resources and usually only learns shallow features, leading to less satisfactory detection results. Many scholars have turned their attention to the current trend of deep learning, hoping to import network traffic data directly into the model to skip the feature selection step. In one study [11], the authors propose a model structure based on deep belief networks (DBNs) and probabilistic neural networks (PNNs) to reduce the dimensionality of the data using deep belief networks and then classify the data using a probabilistic neural network, which is superior to the traditional PNNs. The authors in [12] propose a convolutional neural network-based detection method by processing traffic data into image form, saving the process of designing features manually. In another study [13], the authors use RNN networks for Botnet anomaly detection, and the effectiveness of RNN networks on timing features is utilized to further improve the accuracy of classification. Table 1 gives a summary and summary of the relevant research.

Table 1. Summary of relevant research.

Author(s)	Year	Algorithm	Main Contribution	Field
Amor et al. [6]	2004	Naive Bayes	Proposed the use of naive Bayes for the identification of anomalous networks.	Machine Learning
Kim et al. [10]	2014	C4.5 decision tree and SVM	Proposed a novel intrusion detection method that first decomposes the network data into smaller subsets by C4.5 decision tree algorithm and then creates multiple SVM models for the subsets.	Machine Learning
Shapoorifard et al. [9]	2017	K-MEANS and KNN	Proposed a classifier combining K-MEANS clustering and KNN classifier to improve the accuracy of detection.	Machine Learning
Tao et al. [7]	2018	SVM and genetic algorithm	Proposed genetic algorithm to optimize the selection, parameters, and weights of SVM features.	Machine Learning
Jiadong et al. [8]	2019	Random forest	Proposed a multilevel random forest model to detect abnormal network behavior.	Machine Learning
Torres et al. [13]	2016	RNN	Proposed the use of RNN model to Botnet anomaly detection.	Deep Learning
Wang et al. [12]	2017	CNN	Proposed to use CNN to detect the network traffic data. Processed into the form of pictures.	Deep Learning
Zhao et al. [11]	2017	DBN and PNN	Proposed a model structure based on DBN and PNN to reduce the dimensionality of the data using DBN and then classify the data using PNN.	Deep Learning
Su et al. [14]	2020	CNN and LSTM	Proposed a model based on CNN and LSTM to detect each attack type.	Deep Learning

However, there is a problem of uneven distribution in network traffic data, and none of the above networks exploits the correlation between traffic features. In this paper, a DLNID model is proposed to solve the above remaining problems, using adaptive synthetic sampling (ADASYN) for data augmentation of unbalanced samples and a modified stacked autoencoder for data dimensionality reduction. To train and test the performance of the DLNID model, we take the NSL-KDD dataset for simulation testing. The following contributions are presented in this paper:

(1) A DLNID model combining attention mechanism and Bi-LSTM is proposed. This DLNID model can classify network traffic data accurately;
(2) To address the issue of imbalanced network data, ADASYN is used for data augmentation of the minority class of samples eventually making the distribution of the number of each sample type relatively symmetrical, allowing the model to learn adequately;
(3) An improved stacked autoencoder is proposed and used for data dimensionality reduction with the objective of enhancing information fusion.

The rest of this paper follows: Section 2 details the techniques and innovations used in this paper and presents a diagram of the model architecture of the DLNID model. Section 3 presents information about the NSL-KDD dataset used in this paper. Section 4 provides experimental results and analysis. In Section 5, we summarize our study and propose future research.

2. Technology

2.1. ADASYN

Adaptive synthetic sampling (ADASYN) [15] is an adaptive oversampling algorithm based on the minority class samples. Compared with other data expansion algorithms, it is characterized by the fact that it generates more instances in a special space with lower density and fewer instances in feature space with higher density. This feature has the advantage of adaptively shifting decision boundaries to difficult-to-learn samples, so ADASYN is more suitable than other data augmentation algorithms to handle network traffic with severe data imbalance. The algorithm is executed in the following steps:

Step 1: Calculate the number of samples to be synthesized as *G*, which can be expressed as

$$G = (n_b - n_s) \times \beta \tag{1}$$

where n_b represents the majority sample, n_s represents the minority samples, and $\beta \in (0, 1)$.

Step 2: For each minority sample, calculate K neighbors by the Euclidean distance and denote by r_i the proportion of majority class samples contained in the neighbors, which can be expressed as

$$r_i = k/K \tag{2}$$

where *K* represents the current number of neighbors, and *k* represents the majority class sample in the current neighbor.

Step 3: Calculate the number of samples that need to be synthesized for each minority sample according to *G* and synthesize the samples according to Equation (4), which can be expressed as

$$g = G \times r_i \tag{3}$$

$$Z_i = X_i + (X_{zi} - X_i) \times \lambda \tag{4}$$

where g represents the quantity to be synthesized, Z_i represents the synthesized new sample, X_i represents the current minority sample, and X_{Zi} represents a random minority sample among the *k* neighbors of X_i,$\lambda \in (0, 1)$.

2.2. Autoencoder

An autoencoder [16] is an unsupervised learning network architecture, in which the input and output dimensions are the same, and the number of nodes in the middle layer is generally less than the number of nodes on the left and right sides. Figure 1 illustrates a typical autoencoder consisting of two main components, i.e., the encoder and decoder. It

works by using deep learning techniques to find an efficient representation of the input data without losing information. In short, it compresses the original data by using the encoder to obtain a lower-dimensional representation, which is then reconstructed into the original data by the decoder. According to this working principle, we can use the trained encoder as a tool for data dimensionality reduction. Compared with the traditional principal component analysis (PCA) [17] data dimension reduction method, the autoencoder can achieve nonlinear changes, which facilitates the learning of more deep projection data information.

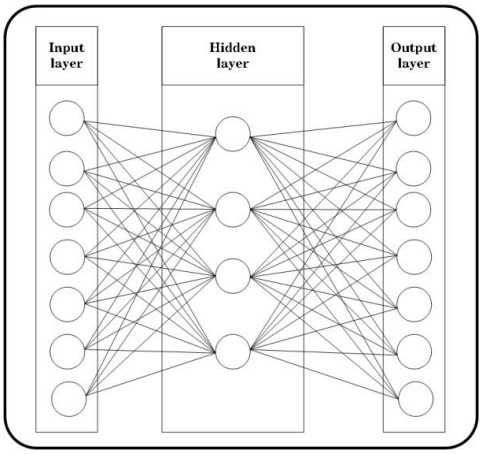

Figure 1. Autoencoder structure.

Although the autoencoder can achieve better data dimensionality reduction, compared with other dimensionality reduction methods, we aimed to propose an autoencoder that is able to perform dimensionality reduction and enhance data robustness to adapt to complex network scenarios. Dropout [18] enables each neuron to have the probability p to be discarded during network training iterations, and due to this mechanism, each neuron is not overly dependent on other neurons, thus reducing the phenomenon of overfitting and improving the generalization ability of the model to some extent. By combining the two ideas, a low-latitude representation is obtained by using dropout and stacked autoencoder after dimensionality reduction. Since each dimension has the probability of being discarded, the information set of each dimension is more comprehensive than that obtained by traditional autoencoder after dimensionality reduction, thus facilitating model learning. Based on the above ideas, we proposed a stacked encoder structure with increased dropout, as shown in Figure 2.

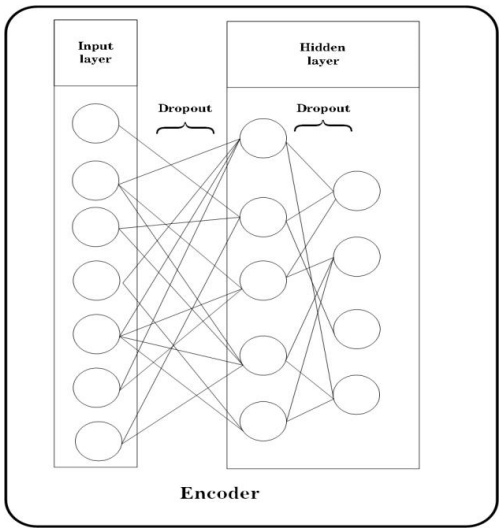

Figure 2. Improved stacked autoencoder.

2.3. Channel Attention

An attention mechanism was proposed based on the idea that people usually tend to focus more on some local regions of the image rather than the image as a whole when observing an image. At ImageNet 2017, the WMW team proposed a squeeze-and-excitation (SE) network based on the channel attention mechanism [19] and won the Image Classification challenge with a great advantage.

The convolutional block attention module (CBAM) [20] is improved on the basis of SE by adding a channel of Maxpool, and through a large number of experiments, the author of [20] proved that adding it can effectively improve the performance of the model classification. Based on these ideas, in this paper, the CBAM used in 3D image processing was applied to the intrusion detection model for 2D data, with modifications. As shown in Figure 3, the flow of the CBAM for 2D data processing is composed of two important phases, i.e., squeeze and excitation. In the squeeze phase, the traffic data are AvgPooling or Maxpooling, from a (c, w)-dimensional form to a (c, 1)-dimensional form, to obtain the global information of each channel. In the excitation phase, the compressed data are adaptively recalibrated by a multilayer perceptron (MLP) to return a weight matrix for each channel.

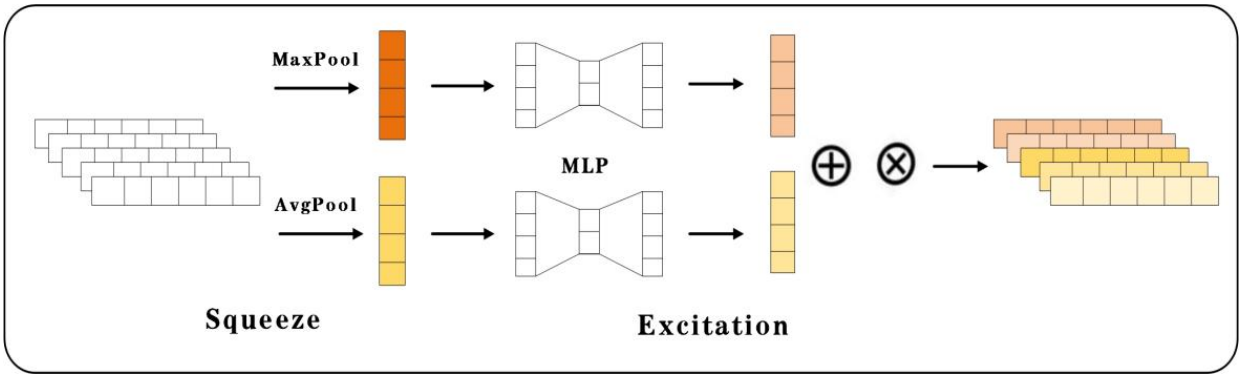

Figure 3. Convolutional block attention module (CBAM).

2.4. Bidirectional LSTM

Long short-term memory (LSTM) [21,22] introduces storage cells and cell states to overcome the long-term dependency problem that exists in recurrent neural networks (RNNs) [23]. The long-term dependency problem is a gradient explosion or gradient dispersion problem caused by multiple multiplications of matrices when RNNs compute the relationship of distant nodes. The following shows how the LSTM network is updated in one time step:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \quad (5)$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \quad (6)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \quad (7)$$

$$\widetilde{c}_t = \tan h(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \quad (8)$$

$$c_t = f_t c_{t-1} + i_t \widetilde{c}_t \quad (9)$$

$$h_t = o_t \tan h(c_t) \quad (10)$$

where i_t, f_t, and o_t represent the input gate, the forget gate, and the output gate, respectively. σ (sigmoid) and tanh represent two distinct activation functions, respectively. c_t represents the current cell state, c_{t-1} represents the previous cell state, and $\widetilde{c}_t$ represents the candidate memory cell. h_t represents the hidden state of the current cell, and h_{t-1} represents the hidden state of the previous cell.

The bidirectional LSTM (Bi-LSTM) network [24] improves its LSTM predecessor by adding backward hidden states $\overleftarrow{h}_t$ to the existing forward hidden states $\overrightarrow{h}_t$, allowing it to obtain a forward-looking capability similar to that of the hidden Markov model (HMM). The following shows how the Bi-LSTM network updates itself in one time step:

$$\overrightarrow{h}_t = \tan h\left(W_{h\overrightarrow{t}}x_t + W_{\overrightarrow{h}\overrightarrow{h}}\overrightarrow{h}_{t-1} + b_{\overrightarrow{h}}\right) \tag{11}$$

$$\overleftarrow{h}_t = \tan h\left(W_{h\overleftarrow{t}}x_t + W_{\overleftarrow{h}\overleftarrow{h}}\overleftarrow{h}_{t-1} + b_{\overleftarrow{h}}\right) \tag{12}$$

$$h_t = \overrightarrow{h}_t + \overleftarrow{h}_t \tag{13}$$

where h_t represents the hidden state of the current cell, h_{t-1} represents the hidden state of the previous cell, $\overrightarrow{h}_t$ represents the forward hidden state of the current cell, and $\overleftarrow{h}_t$ represents the reverse hidden state of the current cell.

For network traffic, Bi-LSTM can effectively utilize the temporal features present in the contextual information to improve the model training, and its structure is shown in Figure 4.

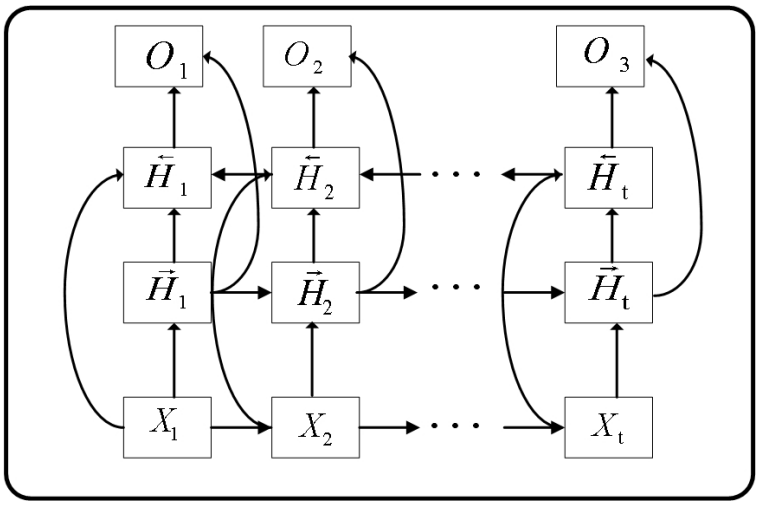

Figure 4. Bi-LSTM structure.

2.5. Network Architecture

As shown in Figure 5, the overall architecture of the DLNID model consists of seven parts, which are the input layer, encoder layer, multiple convolutional layer, attention layer, Bi-LSTM layer, fully connected layer, and the output layer. In the first layer, the model accepts the network traffic data from the dataset. In the encoder layer, the model uses the encoder part of the improved stacked autoencoder that has been trained to perform dimensionality reduction on the data. In the multiple convolutional layer, the model uses multiple convolutional operations to extract features from the downscaled data. In the attention layer, the model uses the CBAM to redistribute the weights of each channel and assign more important channels with higher weights. In the Bi-LSTM layer, the model extracts the feature information of each dimension and learns the relationship between the dimensions. In the fully connected layer and the output layer, the model passes the learned features onto the classifier and outputs the classification results. Algorithm 1 presents the training process of the DLNID model.

Algorithm 1: DLNID Training

Input: NSL-KDD dataset
Output: Accuracy, Precision, Recall, F1 score
For data in the training set or test set; **do**
  One-hot encoding;
  If training set,
    ADASYN data augmentation;
  Normalization;
End.
For data in the training set or test set; **do**
  Use encoder for data dimensionality reduction;
  Perform multilayer convolution operations;
  Use CDAM to redistribute channel weights;
  Use Bi-LSTM to learn sequence information;
  Flatten the dimension;
  Send to the Fully connected layer and classify;
  End.
  Test model on NSL-KDDTest+;
  Obtain loss and update DLNID by Adam;
  Return accuracy, precision, recall, F1 score.

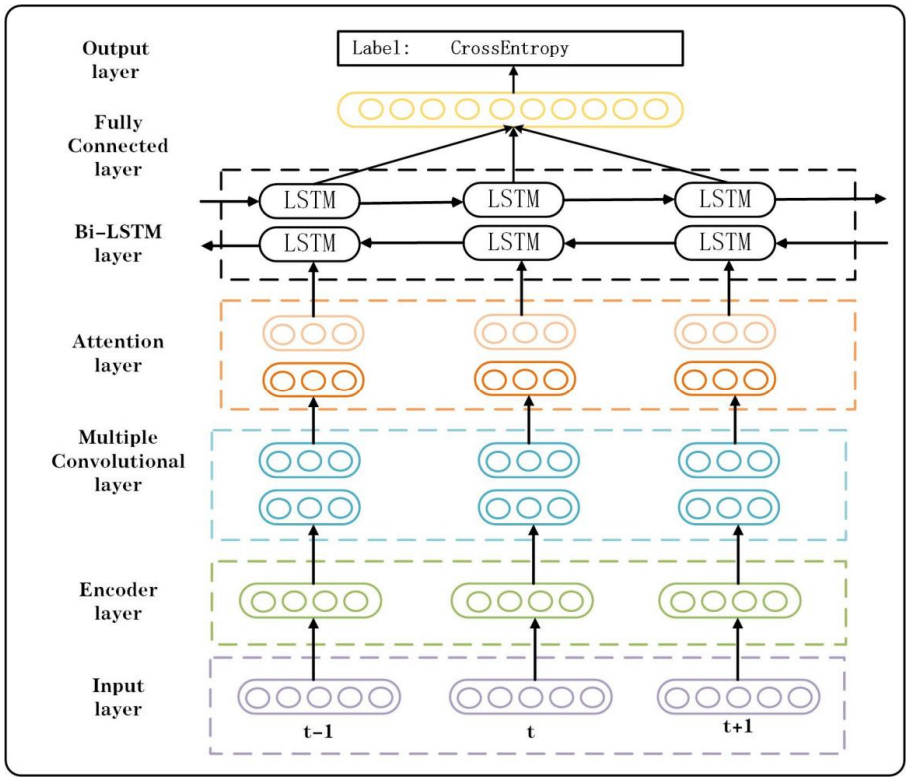

Figure 5. Overall structure of the model.

3. Datasets

3.1. Data Analysis

The experimental data in this paper adopt the NSL-KDD dataset [5], which is an improved version of the KDD99 dataset [25] that addresses the data redundancy problem present in the KDD99 dataset and is one of the benchmark datasets used to evaluate the performance of IDS. It consists of a training set (KDDTrain+), containing 125,973 traffic samples, and a test set (KDDTest+), containing 22,544 traffic samples. In order to restore the complex network situation in reality to a greater extent, there are only 19 attack types in the training set, and the other 17 attack types only exist in the testing set.

The NSL-KDD dataset has a total of 42 dimensional features, one of which is a classification label, and the rest are feature labels. For binary classification, the classification

labels are divided into two categories, i.e., normal and anomaly. For multiclassification, the classification labels are divided into five categories, i.e., normal, Dos, R2L, U2R, and probe.

3.2. Data Preprocessing

3.2.1. One-Hot Encoding

Since there are three non-numerical types of feature values, and the model can only accept numerical types, one-hot encoding was adopted to convert the three non-numerical features into numerical features. For example, the values of protocol_type are TCP, UDP and ICMP, and after encoding, the values become [1, 0, 0], [0, 1, 0], and [0, 0, 1], respectively. Finally, the dataset contains 122 dimensional data after encoding.

3.2.2. Data Augmentation

The number of U2R and R2L samples in the NSL-KDD test set is much higher than that in the training set, and only a small percentage of these samples are in the training set; therefore, the trained model has difficulty distinguishing these samples, so we used the aforementioned ADASYN algorithm to expand the data and expand the samples (such as U2R and R2L) that account for a smaller percentage of the original training set, balancing the percentage of the majority and minority samples. This can solve the imbalance problem in the network data to a certain extent and further boost the generalization ability of the model.

3.2.3. Normalization

A large gap between different dimensional feature data within the dataset can bring about problems such as slow model training and insignificant accuracy improvement; therefore, in order to tackle this issue, the MinMaxScaler [26] was adopted to map the data into the range of (0,1) as follows:

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{14}$$

where x_{max} is the maximum value, and x_{min} is the minimum value.

4. Results

In the following section, we detail the experimental settings and appraise the performance metrics of the model. In addition, we present two sets of ablation experiments to verify the reliability of the data augmentation and improve dimensionality reduction approaches proposed in Section 2. Finally, we compare the model with other papers.

4.1. Experimental Settings

In this study, all experiments were conducted in the hardware environment of Intel(R) Core(TM) i5-1035G1 CPU @ 1.00 GHz 1.19 GHz, with the operating system of windows 10, using Python 3.7, PyTorch 1.10, and the sklearn library for writing and simulating the model.

4.2. Performance Metrics

The confusion matrix was selected as the classification metric of the model predicted data. Additionally, accuracy (Acc), precision (Pre), recall (Rec), and F1 score (F1) were selected as the performance indicators for binary classification, while recall and false positive rate (FPR) were used as the performance indicators for multiclassification. The computation of each performance indicator is detailed as follows:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{15}$$

$$Pre = \frac{TP}{TP + FP} \tag{16}$$

$$\text{Rec} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{17}$$

$$\text{F1} = \frac{2 \times \text{Pre} \times \text{Rec}}{\text{Pre} + \text{Rec}} \tag{18}$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{19}$$

4.3. Result Analysis

The experiment studied the performance of the proposed network on normal, Dos, R2L, U2R, and probe for binary and multiclassification experiments, respectively. When the network parameters were chosen as shown in Table 2, the high accuracy and F1 score could be achieved on the KDDTest+ test set. Figures 6 and 7 show the experimental results using the confusion matrix. The experimental results show that most samples were classified correctly, which appear on the diagonal, indicating a better classification performance. However, the comparison between the two figures shows that the performance of the proposed model was somewhat degraded on the multiclassification experiments, compared with the binary classification experiments. Table 3 provides the false-positive and recall rates corresponding to different attacks under the multiclassification task; the aim was to achieve a lower false-positive rate and a higher recall rate in intrusion detection. From the analysis, it can be concluded that despite the data augmentation process, the U2R category was more likely to be misclassified because the U2R category in the test set was much larger than the others in the training set.

Table 2. Model parameters.

Type	Parameter
Encoder	-
Conv1d	5×5
BatchNorma1d	-
Maxpool1d	3×3
Conv1d	1×1
ChannelAttention	-
Bidirectional LSTM	-
Dropout	0.3
Fully connected (LeakyRelu)	32
Dropout	0.2
Fully connected ()	16
Loss function	CrossEntropy
Optimizer	Adam
Learning rate	0.0005

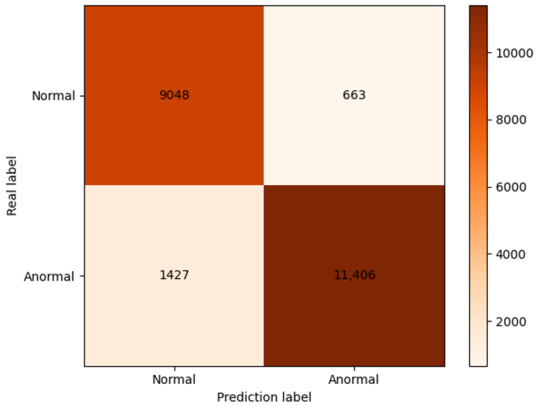

Figure 6. Confusion matrix (2 classes).

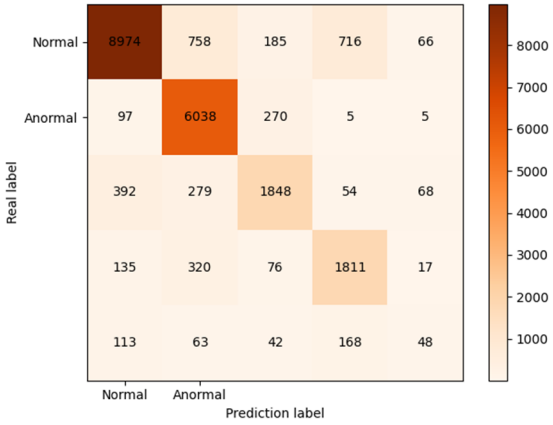

Figure 7. Confusion matrix (5 classes).

Table 3. Rec and FAR on the KDDTest+ (5 classes).

Type	Rec (%)	FPR (%)
Normal	92.14	13.44
Dos	80.96	2.47
Probe	76.33	3.94
R2L	65.76	2.77
U2R	24.00	1.73

4.3.1. Comparison of Data Enhancement Methods

Table 4 shows the experimental results of the transverse comparison test by selecting different data augmentation algorithms under the condition in which the network model was the same, and the dimensionality reduction method remained unchanged. Compared with the unprocessed data, each performance indicator of the proposed model underwent a large improvement. Compared with the SMOTE data augmentation method, the data augmentation method used in this paper also revealed a certain improvement in accuracy and F1 score, with an increase of 2.09% and 2.61%, respectively.

Table 4. Comparison of data enhancement methods.

Type	ACC (%)	Pre (%)	Rec (%)	F1 (%)
Not processed	80.01	70.71	91.63	79.82
SMOTE [27]	88.64	85.32	88.83	87.04
ADASYN	90.73	86.38	93.17	89.65

4.3.2. Dimensionality Reduction Comparison

Table 5 shows the experimental results of selecting different dimensionality reduction methods for horizontal comparison under the condition in which the model was the same, and the data augmentation method remained unchanged. Compared with the PCA, the performance of each model in this paper greatly improved. Compared with the autoencoder, the improved stacked autoencoder used in this paper also had some improvement in accuracy and F1 score, with an increase of 4.64% and 3.28%, respectively.

Table 5. Dimensionality reduction comparison.

Type	ACC (%)	Pre (%)	Rec (%)	F1 (%)
PCA	85.29	83.45	82.14	82.79
Autoencoder	86.09	85.08	82.12	83.57
Improved stacked autoencoder	90.73	86.38	93.17	89.65

4.3.3. Model Comparison

Figure 8 compares the proposed DLNID model with other reference models in terms of accuracy, and it can be seen that the accuracy of DLNID is higher than other models. Table 6 compares the proposed model and other network models in terms of various performance metrics, from which it can be seen that the proposed DLNID model outperforms its comparison peers in terms of Accuracy and F1 score, reaching 90.73% and 89.65% on the KDDTest+ dataset, respectively. Compared with the traditional machine learning methods such as GAR-Forest or NB Tree, the proposed method required no manual feature extraction and improves the accuracy rate. Compared with the autoencoder, the improved stacked autoencoder proposed in this paper enhanced the information set after dimensionality reduction and achieved better classification results. Compared with the CNN, the proposed model used CNN to first extract feature information and then reassign the weights of channels by using the attention mechanism, and finally, learn the relationship between features in the network traffic by Bi-LSTM, thereby achieving an improved classification performance.

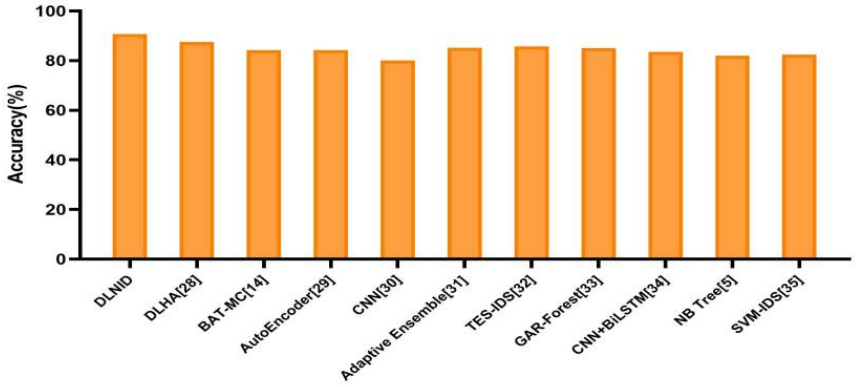

Figure 8. Accuracy comparison.

Table 6. Overall performance comparison.

Type	ACC (%)	Pre (%)	Rec (%)	F1 (%)
DLNID	90.73	86.38	93.17	89.65
DLHA [28]	87.55	88.16	90.14	89.19
BAT-MC [14]	84.25	-	-	-
Autoencoder [29]	84.24	87.00	80.37	81.98
CNN [30]	80.13	-	-	-
Adaptive Ensemble [31]	85.20	86.50	86.50	85.20
TES-IDS [32]	85.79	88.00	86.80	87.39
GAR-Forest [33]	85.06	87.50	85.10	85.10
CNN+BiLSTM [34]	83.58	85.82	84.49	85.14
NB Tree [5]	82.02	-	-	-
SVM-IDS [35]	82.37	-	-	-

5. Conclusions

To address the issue of data imbalance in network data and low detection accuracy, we proposed an ADASYN oversampling algorithm as the data augmentation method to tackle the network intrusion data imbalance problem, the stacked autoencoder with increased dropout structure as the data downscaling method, to improve the generalization ability of the model, and the network structure by combining the channel attention mechanism with the bidirectional LSTM network. The accuracy and F1 score of the proposed network model reached 90.73% and 89.65% on the KDDTest+ test set, respectively. Compared with other reference network models, the proposed DLNID model offered a better classification performance. The proposed network model is considered useful for the current development

of network intrusion detection. In the future, we plan to apply the DLNID model to an actual, combined network capture module to implement an online intrusion detection model.

Author Contributions: Methodology, Y.F. and Y.D.; funding acquisition, W.X.; investigation, Y.F., Y.D., W.X. and Q.L.; resources, Z.C. and W.X.; validation, Z.C. and Q.L.; writing—original draft preparation, Y.D.; writing—review and editing, Y.F. All authors have read and agreed to the published version of the manuscript.

Funding: The work of Y.F., Y.D., Z.C. and Q.L. is supported, in part, by Shannxi S&T under Grant 2021KW-07 and Shaanxi Education under Fund 19jk0414.

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

References

1. Patel, A.; Qassim, Q.; Wills, C. A survey of intrusion detection and prevention systems. *Inf. Manag. Comput. Secur.* **2010**, *18*, 277–290. [CrossRef]
2. Khraisat, A.; Gondal, I.; Vamplew, P.; Kamruzzaman, J. Survey of intrusion detection systems: Techniques, datasets and challenges. *Cybersecurity* **2019**, *2*, 20. [CrossRef]
3. Yuan, L.; Chen, H.; Mai, J.; Chuah, C.N.; Su, Z.; Mohapatra, P. Fireman: A toolkit for firewall modeling and analysis. In Proceedings of the 2006 IEEE Symposium on Security and Privacy (S&P'06), Berkeley/Oakland, CA, USA, 21–24 May 2006; IEEE: Manhattan, NY, USA, 2006; pp. 15–213.
4. Musa, U.S.; Chhabra, M.; Ali, A.; Kaur, M. Intrusion detection system using machine learning techniques: A review. In Proceedings of the 2020 International Conference on Smart Electronics and Communication (ICOSEC), Trichy, India, 10–12 September 2020; IEEE: Manhattan, NY, USA, 2020; pp. 149–155.
5. Tavallaee, M.; Bagheri, E.; Lu, W.; Ghorbani, A.A. A detailed analysis of the KDD CUP 99 data set. In Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, Ottawa, ON, Canada, 8–10 July 2009; IEEE: Manhattan, NY, USA, 2009; pp. 1–6.
6. Amor, N.B.; Benferhat, S.; Elouedi, Z. Naive bayes vs. decision trees in intrusion detection systems. In Proceedings of the Proceedings of the 2004 ACM Symposium on Applied Computing, Nicosia, Cyprus, 14–17 March 2004; Association for Computing Machinery: New York, NY, USA, 2004; pp. 420–424.
7. Tao, P.; Sun, Z.; Sun, Z. An improved intrusion detection algorithm based on GA and SVM. *IEEE Access* **2018**, *6*, 13624–13631. [CrossRef]
8. Jiadong, R.; Xinqian, L.; Qian, W.; Haitao, H.; Xiaolin, Z. A multi-level intrusion detection method based on KNN outlier detection and random forests. *J. Comput. Res. Dev.* **2019**, *56*, 566.
9. Shapoorifard, H.; Shamsinejad, P. Intrusion detection using a novel hybrid method incorporating an improved KNN. *Int. J. Comput. Appl.* **2017**, *173*, 5–9. [CrossRef]
10. Kim, G.; Lee, S.; Kim, S. A novel hybrid intrusion detection method integrating anomaly detection with misuse detection. *Expert Syst. Appl.* **2014**, *41*, 1690–1700. [CrossRef]
11. Zhao, G.; Zhang, C.; Zheng, L. Intrusion detection using deep belief network and probabilistic neural network. In Proceedings of the 2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC), Guangzhou, China, 21–24 July 2017; IEEE: Manhattan, NY, USA, 2017; Volume 1, pp. 639–642.
12. Wang, W.; Zhu, M.; Zeng, X.; Ye, X.; Sheng, Y. Malware traffic classification using convolutional neural network for representation learning. In Proceedings of the 2017 International Conference on Information Networking (ICOIN), Da Nang, Vietnam, 11–13 January 2017; IEEE: Manhattan, NY, USA, 2017; pp. 712–717.
13. Torres, P.; Catania, C.; Garcia, S.; Garino, C.G. An analysis of recurrent neural networks for botnet detection behavior. In Proceedings of the 2016 IEEE Biennial Congress of Argentina (ARGENCON), Buenos Aires, Argentina, 15–17 June 2016; IEEE: Manhattan, NY, USA, 2016; pp. 1–6.
14. Su, T.; Sun, H.; Zhu, J.; Wang, S.; Li, Y. BAT: Deep learning methods on network intrusion detection using NSL-KDD dataset. *IEEE Access* **2020**, *8*, 29575–29585. [CrossRef]
15. He, H.; Bai, Y.; Garcia, E.A.; Li, S. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In Proceedings of the 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), Hong Kong, China, 1–8 June 2008.
16. Meng, Q.; Catchpoole, D.; Skillicom, D.; Kennedy, P.J. Relational autoencoder for feature extraction. In Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, USA, 14–19 May 2017; IEEE: Manhattan, NY, USA, 2017; pp. 364–371.
17. Roweis, S. EM algorithms for PCA and SPCA. *Adv. Neural Inf. Processing Syst.* **1998**, *10*, 626–632.
18. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.

19. Jie, H.; Li, S.; Gang, S. Squeeze-and-excitation networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018.
20. Woo, S.; Park, J.; Lee, J.Y.; Kweon, I.S. Cbam: Convolutional block attention module. In Proceedings of the European conference on computer vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 3–19.
21. Greff, K.; Srivastava, R.K.; Koutník, J.; Steunebrink, B.R.; Schmidhuber, J. LSTM: A search space odyssey. *IEEE Trans. Neural Netw. Learn. Syst.* **2017**, *28*, 2222–2232. [CrossRef] [PubMed]
22. Gui, Z.; Sun, Y.; Yang, L.; Peng, D.; Li, F.; Wu, H.; Guo, C.; Guo, W.; Gong, J. LSI-LSTM: An attention-aware LSTM for real-time driving destination prediction by considering location semantics and location importance of trajectory points. *Neurocomputing* **2021**, *440*, 72–88. [CrossRef]
23. Lin, T.; Horne, B.G.; Tino, P.; Giles, C.L. Learning long-term dependencies in NARX recurrent neural networks. *IEEE Trans. Neural Netw.* **1996**, *7*, 1329–1338. [PubMed]
24. Hochreiter, S. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.* **1998**, *6*, 107–116. [CrossRef]
25. Engen, V.; Vincent, J.; Phalp, K. Exploring discrepancies in findings obtained with the KDD Cup 99 data set. *Intell. Data Anal.* **2011**, *15*, 251–276. [CrossRef]
26. Bisong, E. Introduction to scikit-learn. In *Building Machine Learning and Deep Learning Models on Google Cloud Platform*; Apress: Berkeley, CA, USA, 2019; pp. 215–229.
27. Chawla, N.V.; Bowyer, K.W.; Hall, L.O.; Kegelmeyer, W.P. SMOTE: Synthetic minority over-sampling technique. *J. Artif. Intell. Res.* **2002**, *16*, 321–357. [CrossRef]
28. Wisanwanichthan, T.; Thammawichai, M. A double-layered hybrid approach for network intrusion detection system using combined naive bayes and SVM. *IEEE Access* **2021**, *9*, 138432–138450. [CrossRef]
29. Ieracitano, C.; Adeel, A.; Morabito, F.C.; Hussain, A. A novel statistical analysis and autoencoder driven intelligent intrusion detection approach. *Neurocomputing* **2020**, *387*, 51–62. [CrossRef]
30. Ding, Y.; Zhai, Y. Intrusion detection system for NSL-KDD dataset using convolutional neural networks. In Proceedings of the 2018 2nd International Conference on Computer Science and Artificial Intelligence, Shenzhen, China, 8–10 December 2018; pp. 81–85.
31. Gao, X.; Shan, C.; Hu, C.; Niu, Z.; Liu, Z. An adaptive ensemble machine learning model for intrusion detection. *IEEE Access* **2019**, *7*, 82512–82521. [CrossRef]
32. Tama, B.A.; Comuzzi, M.; Rhee, K. TSE-IDS: A two-stage classifier ensemble for intelligent anomaly-based intrusion detection system. *IEEE Access* **2019**, *7*, 94497–94507. [CrossRef]
33. Kanakarajan, N.K.; Muniasamy, K. Improving the accuracy of intrusion detection using GAR-forest with feature selection. In Proceedings of the 4th International Conference on Frontiers in Intelligent Computing: Theory and Applications (FICTA) 2015, Durgapur, India, 16–18 November 2015; Springer: New Delhi, India, 2016; pp. 539–547.
34. Jiang, K.; Wang, W.; Wang, A.; Wu, H. Network intrusion detection combined hybrid sampling with deep hierarchical network. *IEEE Access* **2020**, *8*, 32464–32476. [CrossRef]
35. Pervez, M.S.; Farid, D.M. Feature selection and intrusion classifi-cation in NSL-KDD Cup 99 dataset employing SVMs. In Proceedings of the 8th International Conference on Software, Knowledge, Information Management and Applications (SKIMA 2014), Dhaka, Bangladesh, 18–20 December 2014; pp. 1–6.