Article

An Efficient Intrusion Detection Method Based on LightGBM and Autoencoder

Chaofei Tang ¹, Nurbol Luktarhan ^{2,*} and Yuxin Zhao ¹

¹ College of Software, Xinjiang University, Urumqi 830046, China; fei1318281@stu.xju.edu.cn (C.T.); zhaoyuxin@stu.xju.edu.cn (Y.Z.)

² College of Information Science and Engineering, Xinjiang University, Urumqi 830046, China

* Correspondence: nurbol@xju.edu.cn

Received: 19 August 2020; Accepted: 2 September 2020; Published: 4 September 2020

Abstract: Due to the insidious characteristics of network intrusion behaviors, developing an efficient intrusion detection system is still a big challenge, especially in the era of big data where the number of traffic and the dimension of each traffic feature are high. Because of the shortcomings of traditional common machine learning algorithms in network intrusion detection, such as insufficient accuracy, a network intrusion detection system based on LightGBM and autoencoder (AE) is proposed. The LightGBM-AE model proposed in this paper includes three steps: data preprocessing, feature selection, and classification. The LightGBM-AE model adopts the LightGBM algorithm for feature selection, and then uses an autoencoder for training and detection. When a set of data containing network intrusion behaviors are inputted into an autoencoder, there is a large reconstruction error between the original input data and the reconstructed data obtained by the autoencoder, which provides a basis for intrusion detection. According to the reconstruction error, an appropriate threshold is set to distinguish symmetrically between normal behavior and attack behavior. The experiment is carried out on the NSL-KDD dataset and implemented using Pytorch. In addition to autoencoder, variational autoencoder (VAE) and denoising autoencoder (DAE) are also used for intrusion detection and are compared with existing machine learning algorithms such as Decision Tree, Random Forest, KNN, GBDT, and XGBoost. The evaluation is carried out through classification evaluation indexes such as accuracy, precision, recall, F1-score. The experimental results show that the method can efficiently separate the attack behavior from normal behavior according to the reconstruction error. Compared with other methods, the effectiveness and superiority of this method are verified.

Keywords: intrusion detection; LightGBM; feature selection; autoencoder; classification; deep learning

1. Introduction

In recent years, computer networks have developed rapidly, gradually playing the role of central information systems in modern life. The increase in the size, application, and infrastructure of computer networks has exposed them to various serious threats such as malicious activities, network intruders, and network criminals. Dealing with these harmful network activities is one of the priorities and important research fields in the world today. Network intrusion detection is an important data analysis task that helps identify network intrusions and protect network security [1]. The detection methods of intrusion detection systems are divided into two categories depending on the modeling methods used [2]: one is based on misuse detection methods, and the other is based on abnormal detection methods. The misuse-based detection method uses signatures that compare known attacks to detect. This method is effective for known attacks, but it is not effective in detecting unknown attacks. In contrast, anomaly-based intrusion detection methods can identify unknown or zero-day attacks.

Because the normal behavior of network traffic has changed greatly, it is difficult to get a precise definition of normal behavior. Therefore, false alarms of the abnormal-based detection method are higher than other methods.

However, the high dimensionality and complex data types of network traffic data make network intrusion detection a challenging task. Many machine learning algorithms, such as SVM [3], logistic regression [4], and XGBoost [5], have been used to develop intrusion detection models and have achieved good results. In recent years, deep learning has received extensive attention due to its ability to mine high-dimensional and large-scale data. It has successfully solved many problems faced by research fields such as text classification [6], target recognition [7], and image classification [8], and have also been gradually applied in network intrusion detection systems. Deep learning algorithms such as RNN [9] and DNN [10] have also achieved good results in intrusion detection.

In this paper, an intrusion detection system based on LightGBM and AE (Autoencoder) is proposed to detect normal and attack behavior. The published NSL-KDD dataset (an improved version of the original KDD Cup 1999 data (KDD99) [11]) is used as a benchmark to evaluate the proposed deep learning-based IDS (Intrusion Detection System). Specifically, the IDS introduced here includes three main steps: (1) data preprocessing, (2) feature selection, and (3) classification. The data preprocessing part normalizes the features by min-max, that is, scaling the features to the range [0,1] and converting the categorical features into numeric values with one-hot-encoding technology. Feature selection uses the LightGBM algorithm to select the most important feature according to the feature importance score. In the classification module step, when a set of data with attack behavior is input into the trained AE model, the model will reconstruct the input data, and the reconstruction of the attack behavior will produce a large error. Setting an appropriate threshold according to the reconstruction error can detect symmetrical attack behaviors in network traffic data. This article only carries out binary classification. Binary classification includes normal and attacks, where attacks include Probe, DoS(Denial of Service), R2L (Root to Local) and U2R (User to Root). In addition to the AE model, this paper also uses VAE and DAE for intrusion detection and compares it with commonly used machine learning algorithms. The performance of the AE model proposed in this paper is better than other methods, with an accuracy rate of 89.82.

The main contributions of this article are summarized as follows:

- The feature selection method based on the LightGBM algorithm is applied to intrusion detection;
- According to the error of reconstructed data obtained by the autoencoder, an appropriate threshold is set to identify normal and attack behaviors. Based on this, an innovative IDS is developed;
- Compared with the deep learning algorithms such as VAE, DAE, and other machine learning algorithms such as Decision Tree, Random Forest, etc., the performance of the proposed LightGBM-AE is verified, and the LightGBM-AE can effectively distinguish the normal behavior from the attack behavior in the NLS-KDD dataset.

2. Related Work

This section discusses the literature work of network intrusion detection systems. Intrusion detection has become an important part of the infrastructure of the information security defense network system [12]. In intrusion detection systems, various machine learning algorithms and deep learning algorithms are used to detect and distinguish normal and attack behavior in network traffic.

Due to the complex network intrusion behavior, single classifier IDS cannot achieve high accuracy. Ansam Khraisat et al. [13] combined the C5 decision tree classifier with a One-Class SVM and evaluated the proposed IDS using the benchmark NSL-KDD dataset. The accuracy rate reached 83.24%, and the performance of IDS was improved. T. Ait Tchakoucht and M. Ezziyyani [14] modeled intrusion detection based on the multilayered echo-state machine (ML-ESM). In the public datasets KDD99, NSL-KDD, and UNSW-NB15, the performance of its binary classification and multi-classification was evaluated. The first method proposed by Samrat Kumar Dey et al. [15] is based on a machine-learning algorithm, uses the gain ratio for feature selection, and then uses a random forest classifier for

detection. The NSL-KDD dataset can guarantee 82% accuracy. The second method uses a deep learning algorithm, combined with a gated periodic unit long-short-term memory (GRU-LSTM) network intrusion detection system, the accuracy rate is close to 88%. More and more deep learning methods are applied to intrusion detection and show excellent performance. Kaichen Yang et al. [16] studied how adversarial examples affect the performance of deep neural networks (DNN) that are trained to detect network intrusion in black-box models. It proves that even when the internal information of the target model is separated from the adversary, the adversary can produce an effective adversarial example against the trained DNN classifier. They trained a DNN model for a network intrusion detection system using the NSL-KDD dataset, and achieved 89% accuracy. In addition to DNN, other neural networks can also effectively perform intrusion detection, such as RNN. Chuanlong Yin et al. [17] use deep learning methods-recurrent neural networks (RNN) for intrusion detection. The performance of the model in binary classification was studied, and experiments were performed on the NSL-KDD dataset. The accuracy of RNN is 83.28%, which is higher than that of machine learning methods such as J48, artificial neural networks, random forests, and support vector machines. Experimental results prove that the RNN algorithm is very suitable for high-precision modeling of classification models. Autoencoder is an unsupervised deep learning framework designed to reconstruct the input in the output while minimizing the reconstruction error [18]. S. Zavrak et al. [19] adopted the method of an autoencoder and variational autoencoder, and compared it with the OCSVM algorithm. This involved an experiment on the CICDS2017 dataset, extraction of the stream-based features from it, a calculation of the ROC curve and AUC value, and an analysis of the performance of the method under different thresholds. Experimental results show that the AUC value obtained by the variational autoencoder is 0.7596, which is better than that of the autoencoder and single-class support vector machine, but it is not easy to determine an appropriate threshold that provides high detection accuracy or low false alarm rate. Cosimo Ieracitano et al. [20] have developed an intelligent intrusion detection system based on statistical analysis and autoencoder. Combine data analysis and statistical techniques for feature extraction and then an autoencoder is used to reduce the 102-dimensional feature vector z to a 50-dimensional latent feature vector e , and then reconstruct the original input with 50 compressed features. The feature vector e is used as the input of the final softmax layer for binary classification. The effectiveness of the proposed IDS was tested using the benchmark NSL-KDD dataset, and an accuracy of 84.21% was achieved, which is superior to algorithms such as LSTM and MLP.

3. Dataset and Methodology

In this section, we first introduce the NSL-KDD dataset used in this article. On this basis, the method is introduced, including data preprocessing, feature selection and classification.

3.1. NSL-KDD Dataset

The NSL-KDD dataset is an improved version of the original KDD99 dataset and is widely used as a benchmark in many intrusion detection systems. The NSL-KDD dataset solves some of the inherent problems of the previous KDD99, such as the existence of a large number of redundant and repeated records in the training and test set, these records bias the classifier to more frequent samples. It has training and test dataset, denoted here as KDDTrain+ and KDDTest+, containing 125,973 and 22,544 instances, respectively. In addition, the NSL-KDD dataset contains four different attack classes: Probe, DoS, R2L, and U2R. The distribution information of KDDTrain+ and KDDTest+ in normal and four attack types are shown in Table 1. Also, the attack types of the NSL-KDD dataset are grouped into four different attack categories:

1. Probe: Probe includes attacks that collect information about the network to effectively avoid the security control systems.
2. DoS: DoS includes attacks that cause the machine to slow down or shut down by sending traffic information that exceeds the system's processing capacity to the server. Legitimate network traffic or access to services is affected by DoS attacks.
3. R2L: R2L includes attacks that illegally access computers by sending remote spoofing packets to the system.
4. U2R: U2R include attacks that provide root access. In this case, the hacker finds out the system vulnerability and starts using the system as a normal user.

As shown in Table 2, the NSL-KDD dataset contains a total of 39 attacks, each of which is divided into one of the following four categories (Probe, DoS, R2L, and U2R). Furthermore, only a new set of attacks are introduced in the test set, and these new attacks are shown in bold.

Table 1. Detailed quantity distribution information of the NSL-KDD dataset.

NSL-KDD	Normal	Probe	DoS	R2L	U2R
Train(125,973)	67,343 53.46%	11,656 9.25%	45,927 36.46%	995 0.79%	52 0.04%
Test(22,544)	9711 43.07%	2421 10.74%	7458 33.08%	2754 12.22%	200 0.89%

Table 2. List of attacks presented in the NSL-KDD dataset.

Attack Category	Attack Name
Probe	Satan, Saint , Ipsweep, Portsweep, Nmap, Mscan
DoS	Apache2 , Smurf, Neptune, Back, Teardrop, Pod, Land, Mailbomb , Processtable , UDPstorm
R2L	WareZClient, Guess_Password, WareZMaster, Imap, Ftp_Write, Named , MultiHop, Phf, Spy, Sendmail , SnmpGetAttack , Worm , Xsnoop , Xlock , SnmpGuess
U2R	Buffer_Overflow, Http tuneel , Rootkit, Perl, Ps, Xterm , SQL attack , LoadModule

3.2. Methodology

Figure 1 shows a flow chart of the proposed method. Firstly, the NSL-KDD dataset is preprocessed, and the min-max normalization technique is applied to scale the data to the interval [0,1]. Then, the symbolic features are converted into numerical values by using one-hot-encoding technology. Afterward, the LightGBM algorithm is used for feature selection, and the optimal features are selected from the 41 features to form the optimal feature subset. Finally, we developed AE technology to evaluate the detection performance of IDS in binary classification scenarios.

Figure 1. Scheme of the proposed framework. It consists of a data preprocessing, a feature selection, and a classification module.

3.3. Data Preprocessing

Data preprocessing is a necessary step before training the model. It includes two parts: data normalization and one-hot-encoding.

3.3.1. Data Normalization

The min-max normalization method is adopted to scale the value x_{ij} into the numeric range $[0,1]$, according to:

$$\tilde{x}_{fj} = \frac{x_{fj} - \min(x_f)}{\max(x_f) - \min(x_f)}. \quad (1)$$

Among them, $\max(x_f)$ and $\min(x_f)$ represent the maximum and minimum value of the f_{th} (numerical) feature x_f ; whereas $\tilde{x}_{fj}$ is the normalized feature value ranged between $[0,1]$.

3.3.2. One-Hot-Encoding

One-hot-encoding technology is used to convert three categorical features *protocol_type*, *service*, and *flag* (x_2 , x_3 , x_4 , respectively) into numeric values. In particular, each categorical attribute is represented by binary values. For instance, the x_2 feature (*protocol_type*) has three attributes: *tcp*, *udp*, and *icmp*. One-hot-encoding technology converts them into binary vectors: $[1,0,0]$, $[0,1,0]$, $[0,0,1]$, respectively. In the same way, x_3 and x_4 features (*service* and *flag*) are also converted into one-hot-encoding vectors. In general, 41-dimensional features are mapped to 122-dimensional features (38 continuous, and 84 binary values related to features x_2 , x_3 , x_4).

3.4. Feature Selection

Next, feature selection is applied, which is essential in the classification task. Feature selection has the advantages of reducing computational complexity, improving the performance of learning algorithms, eliminating redundant information in the dataset, and improving the generalization of data [21]. LightGBM is a new boosting framework proposed by Microsoft in 2017, which is more powerful, faster, and greatly improved in performance than Xgboost, as described in [22]. The performance of the LightGBM model has been widely recognized in several data mining and machine learning challenges. Therefore, we use LightGBM technology and feature importance scores are applied for feature selection.

The LightGBM model is a collection of decision trees. Different from other GBDT models, LightGBM's method of calculating the gain of variation occurs under weak and strong learners (small and big gradients, g_i). The training instances are arranged in descending order according to the absolute value of their gradients, and the first $a\%$ of the instances with larger gradients are retained to form the instance subset A . For the residual set A^c formed by the $(1 - a)\%$ of instances with smaller gradients, a subset B of size $b * |A^c|$ is randomly formed. Finally, the instance is split according to the estimated variance gain $V_j^*(d)$ on the subset $A \cup B$.

$$V_j^*(d) = \frac{1}{n} \left(\frac{(\sum_{x_i \in A_l} g_i + \frac{1-a}{b} \sum_{x_i \in B_l} g_i)^2}{n_l^j(d)} + \frac{(\sum_{x_i \in A_r} g_i + \frac{1-a}{b} \sum_{x_i \in B_r} g_i)^2}{n_r^j(d)} \right), \quad (2)$$

where $A_l = \{x_i \in A : x_{ij} \leq d\}$, $A_r = \{x_i \in A : x_{ij} > d\}$, $B_l = \{x_i \in B : x_{ij} \leq d\}$, $B_r = \{x_i \in B : x_{ij} > d\}$, d is the point in the data where the split is calculated to find the best gain invariance, and the coefficient $\frac{1-a}{b}$ is used to normalize the gradient sum over B back to the size of A^c .

The trees in the LightGBM model are constructed based on the above steps. Let the feature set, x_i be $x_1, x_2, \dots, x_m$ where $i = 1, 2, \dots, m$. Then, according to the number of times each feature is used to split the training data across all trees, the feature importance score FIS_i is calculated. Therefore, the feature importance score set is represented as:

$$FIS_i = \{s | s = w_i x_i\}, \quad (3)$$

where w_i represents the weight of each feature, and x_i represents the feature set. Figure 2 shows the best feature importance score of the NSL-KDD feature used by the LightGBM algorithm.

Figure 2. LightGBM algorithm is applied to score the feature importance of the NSL-KDD dataset.

The accuracy of the LightGBM algorithm has been verified in a large number of experiments, and multiple thresholds are set based on feature importance scores to select features. Using all 41 original features as the beginning of the experiment, and finally ended with a subset containing the selected features. It can be seen from Table 3 that the accuracy of the model changes when different numbers of features are selected, and the highest accuracy is 99.20% when 21 features are selected. Then, the three categorical features (*protocol_type*, *service* and *flag*) are converted into 84 dimensions by one-hot-encoding technology, plus the other 18 features as the optimal feature subset, so the dimension of input data is 102 dimensions. The optimal feature subset is taken as the input of intrusion detection, and the autoencoder is used for network intrusion detection. The selected 21 features are shown in Table 4.

Table 3. Using the LightGBM feature importance score with multiple different thresholds for feature selection and their respective accuracy.

Threshold	Number of Features	Accuracy (%)
0	41	99.12
0.001	33	99.10
0.002	31	99.10
0.003	30	99.07
0.004	29	99.07
0.005	28	99.10
0.006	25	99.13
0.007	23	99.18
0.009	21	99.20
0.013	20	99.07
0.019	17	99.06
0.021	16	99.03
0.024	15	98.80
0.029	14	98.69
0.031	13	98.72

Table 3. Cont.

Threshold	Number of Features	Accuracy (%)
0.036	11	98.67
0.037	10	98.55
0.044	9	98.37
0.047	7	98.34
0.048	6	98.40
0.049	4	95.05
0.061	3	94.17
0.087	2	92.88
0.154	1	88.55

Table 4. Twenty-one features selected based on feature importance.

ID	Data Type	Feature
F1	Continuous	duration
F2	Symbolic	protocol_type
F3	Symbolic	service
F4	Symbolic	flag
F5	Continuous	src_bytes
F6	Continuous	dst_bytes
F10	Continuous	hot
F12	Binary	logged_in
F23	Continuous	count
F24	Continuous	srv_count
F30	Continuous	diff_srv_rate
F32	Continuous	dst_host_count
F33	Continuous	dst_host_srv_count
F34	Continuous	dst_host_same_srv_rate
F35	Continuous	dst_host_diff_srv_rate
F36	Continuous	dst_host_same_src_port_rate
F37	Continuous	dst_host_srv_diff_host_rate
F38	Continuous	dst_host_serror_rate
F39	Continuous	dst_host_srv_serror_rate
F40	Continuous	dst_host_rerror_rate
F41	Continuous	dst_host_srv_rerror_rate

3.5. Autoencoders

To detect normal and attack behaviors (Probe, DoS, R2L, U2R) of the NSL-KDD dataset, deep learning models based on AE, VAE, and DAE are developed respectively. AE, VAE, and DAE are all deep learning models containing symmetry. See the following subsection for details.

3.5.1. Autoencoder

The autoencoder consists of encoder and decoder operations: First, the encoder converts the input data vector to a typical lower-dimensional representation; then, the decoder attempts to reconstruct the original input from the compressed vector. AE is trained in an unsupervised manner and can learn salient features from unlabeled data [23]. Figure 3 shows the AE structure used in this article. The input data vector x is encoded as a lower dimension representation e :

$$e = \zeta(xW + b). \quad (4)$$

In the formula, W denotes the weight matrix, b represents the bias vector, and ζ is the activation function of the encoder. Then, the decoding operation reconstructs the input vector x from the encoded representation e into:

$$\tilde{x} = \xi(eW^T + b), \quad (5)$$

where ξ represents the activation function of the decoder and $\tilde{x}$ is a vector reconstructed from e . The encoding of x to e is trained first, and then the decoding of e to $\tilde{x}$ is trained to minimize the error between the reconstructed $\tilde{x}$ and the input x .

Figure 3. The autoencoder structure adopted in this paper. The autoencoder (AE) consists of two parts: encoding and decoding. The encoding operation converts the input vector x into a compressed representation e ; while the decoding operation attempts to reconstruct the input variable from e , so that $\tilde{x} \approx x$.

3.5.2. Variational Autoencoder

Like the standard autoencoder, the variational autoencoder (VAE) is a deep generator model with latent variables and an architecture composed of encoder and decoder. The purpose of training is to minimize the reconstruction error between the encoded data and the input data. Using Bayesian inference and probabilistic graphical model methods, the input data is encoded into a low-dimensional latent coding space and then decoded back.

The posterior probability function $q_{\phi}(z|x)$ is used as a probability encoder to approximate the intractable posterior $p_{\theta}(z|x)$.

We assume that the prior distribution $p_{\theta}(z)$ is a multivariate Gaussian distribution with a diagonal covariance matrix. We randomly sample the point z from a prior distribution $p_{\theta}(z)$. To make it trainable, the reparameterization technique was introduced [24]. The decoder $p_{\theta}(x|z)$ converts this point in the latent space into the original input samples. The loss function of VAE is defined as:

$$\mathcal{L}_{VAE}(\theta, \phi) = -E_{q_{\phi}(z|x)}[\log p_{\theta}(x|z)] + D_{KL}[q_{\phi}(z|x)||p_{\theta}(z)], \quad (6)$$

where D_{KL} is the Kullback–Leibler divergence, which intuitively measures the degree of similarity between the prior distribution $p_{\theta}(z)$ and the posterior distribution $q_{\phi}(z|x)$.

3.5.3. Denoising Autoencoder

Denoising autoencoder (DAE) is a variant of the autoencoder [25]. DAE adds a noise process $P(\tilde{x}|x)$ in the process of reconstructing input x to $\tilde{x}$. Then, uses x to construct the encoder as $f_{\theta}(\tilde{x}) = s(\omega\tilde{x} + b)$, and reconstructs the decoder as $g_{\theta'}(y) = s(\omega'y + b')$. To calculate the reconstruction error, DAE uses the same method as the autoencoder, except that $\tilde{x}$ is reconstructed by $P(\tilde{x}|x)$, as shown below:

$$\arg \min_{\theta, \theta'} \frac{1}{n} \sum_{x=1}^n \mathcal{L}(x^{(2)}, g_{\theta'}^{(2)}(f_{\theta}^{(2)}(\tilde{x}^{(2)}))). \quad (7)$$

3.6. Classification

The AE model can extract the characteristics of the input data by adjusting the model parameters, and completely retain the key information of the input data to maintain the optimal reconstruction error. When building an autoencoder model for intrusion detection, the key parameters that need to be set include the number of network layers, the number of neurons in the hidden layer, epochs, the learning rate, and the batch size. However, there is currently no good way to find their optimal parameters.

The number of network layers is related to the dimension of the input data. When the input data dimension is large, it generally sets a larger number of network layers. At the same time, the three-layer model can often achieve better detection results for most data [26]. By reducing the number of neurons layer by layer, you can compress the data layer by layer and obtain important information. At the same time, the compression of the number of neurons in each layer cannot be too large, and excessive compression will result in the loss of important information.

At this time, a set of test data is input into the trained intrusion detection model, and a large reconstruction error will occur between the input data and the reconstructed data obtained by the autoencoder. In this paper, the mean square error is used to estimate the error, and the reconstruction error is defined as:

$$MSE = \frac{1}{n} \sum_{i=1}^n (\tilde{x}_i^2 - x_i^2), \quad (8)$$

where x is the input data, $\tilde{x}$ is the corresponding reconstruction vector, and x and $\tilde{x}$ have the same dimension.

The specific detection process is shown in Algorithm 1.

Algorithm 1 The Detection Algorithm With Trained AE

Input: the test dataset $X = \{x\}$;

Output: accuracy, precision, recall, and F1-score;

```

1: Step1: Encoder processes
2:    $e_1 = f(W_1x + b_1)$ 
3:   for  $i = 2$  to  $L$  do
4:      $e_i = f(W_ix_{i-1} + b_i)$ 
5:   end for
6: EndStep
7: Step2: Decoder process
8:    $\tilde{x}_L = f(W'_L e_L + b'_L)$ 
9:   for  $i = L - 1$  to  $1$  do
10:     $x_i = f(W'_i e_{i+1} + b'_i)$ 
11:   end for
12: EndStep
13: Step3: Detection
14:   Calculate the score  $= \frac{1}{n} \sum_{i=1}^n (\hat{x}_i^2 - x_i^2)$ 
15:   Calculate TPR, FPR = roc_curve(x_label, scores)
16:   Set Threshold = max { (TPR-FPR) }
17:   if score > Threshold then
18:     the data point is normal
19:   else
20:     the data point is the attack
21:   end if
22: EndStep

```

The size of the reconstruction error is the basic criterion for evaluating data points to judge normality or attack. To perform network intrusion detection more accurately, the reconstruction error will be further analyzed here. First, determine the threshold T of the reconstruction error. If the reconstruction error is greater than the threshold T , the data point is classified as an attack behavior. Data points in which the reconstruction error is less than or equal to the threshold are classified as normal behavior. Using the label of the test and error set as input, calculate TPR and FPR. For intrusion detection, the higher the TPR and the lower the FPR, the better the detection result. Therefore, find $\max(\text{TPR}-\text{FPR})$, and use this as the threshold. A set of test data is input into the trained model, and the detection process is carried out according to Algorithm 1.

4. Experimental Results

4.1. Experimental Conditions

This experiment is based on Python version 3.6 and PyTorch version 1.3, the experimental environment uses the Ubuntu18.04 64-bit operating system, the GPU is RTX-2080Ti, and there is 64 GB of memory.

4.2. Performance Evaluation

To measure the effectiveness of the AE algorithm in intrusion detection, the detection results are divided into four types: true positive (TP), false positive (FP), true negative (TN) and false negative (FN), which can be expressed in the form of the confusion matrix [27], as shown in Table 5.

Table 5. Confusing matrix.

Actual \ Predict	Predict	
	Normal	Attack
normal	TP	FN
attack	FP	TN

The performance of the proposed IDS is measured using traditional metrics: accuracy, precision, recall, and F1-score (or F-measure):

$$accuracy = \frac{TP + TN}{TP + TN + FN + FP}. \quad (9)$$

$$precision = \frac{TP}{TP + FP}. \quad (10)$$

$$recall = \frac{TP}{TP + FN}. \quad (11)$$

$$\text{F1-score} = 2 \times \frac{precision \times recall}{precision + recall}. \quad (12)$$

4.3. Parameter Settings and Training Details

The performance of AE, VAE, and DAE models in intrusion detection is experimentally studied. Figure 3 shows the structure ([102:48:32:16:32:48:102]) adopted by our proposed autoencoder, and VAE and DAE also adopt the same structure. The activation function of each layer is relu, and the optimizer is Adam. Also, two important parameters need to be set: learning rate and epoch.

The learning rate is an important parameter in the AE model, with values ranging from 0 to 1. Excessive learning rate will lead to loss explosion, while too small a learning rate will lead to slow convergence or over-fitting of the model. For the setting of the learning rate, we try 0.1, 0.01, 0.001, 0.0001, 0.00001, and so on as test values in turn, and then find the optimal value in a certain cell.

As can be seen from Figure 4, the learning rate gradually decreases at a rate of 10 times from 0.1, and the corresponding detection accuracy changes accordingly. When the learning rate is 0.1, the accuracy rate is 76.06%. When the learning rate gradually decreases, the accuracy rate increases. When the learning rate is 0.001, the accuracy rate reaches the highest at 86.84%. As the learning rate continues to decrease, the accuracy rate decreases. We can observe that in the interval [0.01, 0.001], the accuracy rate is the highest, so we continue to look for the optimal learning rate in this interval. As can be seen from Figure 5, when the learning rate is set within [0.001, 0.01], the overall accuracy rate is at a higher level. When the learning rate is 0.004, the highest accuracy rate is 89.82%, so we set the learning rate to 0.004.

Figure 4. The change of accuracy rate when the learning rate starts from 0.1 and decreases to 0.000001 at a rate of 10 times.

Figure 5. The change of accuracy rate when the learning rate starts from 0.001 and increases by 0.001 to 0.01 each time.

For the setting of the epoch, we can get it by observing the changes in training losses. Figure 6 shows the change of training loss after each round of epoch in the model when the epoch is set to 50. In the first few rounds of the epoch, the training loss decreased rapidly and then leveled off. After 20, the epoch does not change, so we set epoch to 20.

Figure 6. The relationship between train loss and epoch size.

4.4. Results and Analysis

This paper proposes an intrusion detection system based on feature selection and autoencoder and compares the performance of the three autoencoders, AE, VAE, and DAE. All experiments and analyses were performed on the benchmark NSL-KDD dataset. Standard evaluation indicators were used to evaluate the effectiveness of the proposed intrusion detection system, including accuracy, precision, recall, and F1-score. According to the feature importance score calculated by the LightGBM algorithm, the most important features were extracted and used as input for deep learning methods (AE, VAE, DAE) and ML methods, including DT, RF, KNN, GBDT, and XGBoost.

Table 6 gives the experimental results of AE, VAE, and DAE. The accuracy of AE is 89.82%, the precision is 91.81%, the recall is 90.16%, and the F1-score is 90.98%, and the accuracy of VAE is 84.28%, the precision is 84.92%, the recall is 88.01%, and the F1-score is 86.43%. Last, the accuracy of DAE is 84.30%, the precision is 85.22%, the recall is 87.63%, and the F1-score is 86.41%. The four evaluation indexes of AE are better than VAE and DAE, and the best test results are obtained.

Table 6. Performance comparison of AE with variational autoencoder (VAE) and denoising autoencoder (DAE).

Model	Accuracy/%	Precision/%	Recall/%	F1-Score/%
AE	89.82	91.81	90.16	90.98
VAE	84.28	84.92	88.01	86.43
DAE	84.30	85.22	87.63	86.41

Compare our three autoencoder models and measure their performance with feature selection or non-selection. The results are shown in Table 7. When all the features are used, the accuracy of the AE model is 87.48%. After feature selection, the accuracy rate is as high as 89.82%, an increase of 2.34%. Regardless of whether feature selection is performed, the accuracy of the AE model is higher than that of VAE and DAE. Some common machine learning algorithms have long been used in network

intrusion detection, such as DT, RF, KNN, GBDT, XGBoost. The results of these five machine learning algorithms are also shown in Table 7, from which it can be seen that DT achieved the highest accuracy. When DT uses all features for network intrusion detection, the accuracy rate is 78.72%, which is 8.76% lower than AE. After feature selection, the accuracy of DT is 80.09%, which is 9.73% lower than AE. Therefore, as can be observed, whether it is compared with the two deep learning models of VAE and DAE, or compared with the machine learning algorithm, the performance of AE is optimal.

Table 7. Comparison between the accuracy of our proposed model and common machine learning algorithms with or without feature selection.

Features	VAE	DAE	DT	RF	KNN	GBDT	XGBoost	Proposed Model (AE)
122	82.56	82.79	78.72	70.10	76.25	78.31	75.85	87.48
102	84.28	84.30	80.09	71.06	76.51	78.34	78.61	89.82

Setting different numbers of hidden layers (HL) and neurons, and evaluating their performance, we can find the optimal AE structure. Specifically, Table 8 reports the accuracy of different AE models with or without feature selection. As can be seen from the results in Table 8, the classification accuracy has a great relationship with the number of hidden layers of AE and the number of neurons. The number of hidden layers of AE is set to four types: 1, 2, 3, and 4, and the number of neurons is set to four types: 64, 48, 32, 16, and 8. The number of hidden layers here refers to the number of layers in the left half of the AE model, and the number of neurons is set to decrease in layers. When using all features for detection, AE_{32,16,8} had the lowest accuracy of 79.58%. After feature selection, AE_{64,8} had the lowest accuracy of 81.03%. The proposed AE_{48,32,16} have the highest accuracy of 87.48% and 89.82%, respectively, which are 7.9% and 8.79% higher than the lowest. When the number of neurons in the first hidden layer is set to 48, let us observe the performance of the four models AE₄₈, AE_{48,32}, AE_{48,32,16}, AE_{48,32,16,8} after feature selection. As the number of hidden layers increases, the accuracy increases from 82.45% of the AE₄₈ model to 89.82% of the AE_{48,32,16} models. As the number of hidden layers continues to increase, the accuracy of the AE_{48,32,16,8} model is 82.61%, which is 7.21% lower than the AE_{48,32,16} model. It can be seen that when using the AE model for intrusion detection, it is not that the more layers of the model, the better. When the hidden layer is three, the AE_{48,32,16} model performs best. For a fair comparison, both VAE and DAE adopt the same structure.

Table 8. Evaluation performance of AE with different hidden layers and neurons.

Classifier	HL1	HL2	HL3	HL4	Accuracy All Features	Accuracy 21 Features
AE ₆₄	64	-	-	-	85.33	87.07
AE _{64,32}	64	32	-	-	82.37	83.89
AE _{64,32,16}	64	32	16	-	84.33	86.40
AE _{64,32,8}	64	32	8	-	83.98	84.07
AE _{64,32,16,8}	64	32	16	8	81.06	83.64
AE _{64,16}	64	16	-	-	84.75	85.31
AE _{64,8}	64	8	-	-	80.22	81.03
AE ₄₈	48	-	-	-	79.25	82.45
AE _{48,32}	48	32	-	-	81.32	83.21
AE _{48,32,16}	48	32	16	-	87.48	89.82
AE _{48,32,16,8}	48	32	16	8	80.75	82.61
AE ₃₂	32	-	-	-	82.65	83.49
AE _{32,16}	32	16	-	-	82.39	83.51
AE _{32,8}	32	8	-	-	81.57	82.69
AE _{32,16,8}	32	16	8	-	79.58	81.31

As shown in Table 9, the AE model proposed in this paper is compared with the latest methods proposed in the literature. They are also trained and tested on the NSL-KDD dataset.

Table 9. Comparison results of the autoencoder model and the latest method on the NSL-KDD dataset.

Work	Classifier	ACC (%)
Khraisat et al. [13]	C5.0 Decision tree+one-class SVM	83.24
Yang, Kaichen, et al. [16]	DNN	89.00
Yin, C., et al. [17]	RNN	83.28
Cosimo Ieracitano et al. [20]	AE+softmax	84.21
Proposed AE model	AE	89.82

5. Conclusions

In this paper, we discussed the shortcomings of existing intrusion detection systems and evaluated the performance of LightGBM-AE-based intrusion detection classification models. The LightGBM model is mainly used in classification and regression tasks. In our proposed model, we select features based on the feature importance score generated by the LightBGM model. This feature selection method is the first in intrusion detection and intrusion classification. After the input data is mapped by the encoder and the decoder, a reconstruction data is generated. According to the size of the reconstruction error, a reasonable threshold can be set to distinguish normal data from attack data. The model proposed in this paper is compared with two autoencoder models—VAE and DAE, and machine learning algorithms such as decision tree. From the comparison results, the classification accuracy of the model proposed in this paper is higher, reaching 89.82%, which is more advantageous than other models. The experimental results show that the method has a good detection effect on network intrusion detection.

In the future, we plan to develop a more accurate deep learning model to effectively carry out network intrusion detection. The proposed work is used for binary classification and can only distinguish normal from attack. Soon, this work can be extended to multiple categories and specific attack types can be identified. In order to perform intrusion detection faster, we will explore the application of the distributed learning method proposed by Langer, M. [28] to our research.

Author Contributions: Conceptualization, C.T. and N.L.; methodology, C.T.; software, Y.Z.; validation, C.T., Y.Z. and N.L.; formal analysis, C.T.; investigation, Y.Z.; resources, C.T.; data curation, Y.Z.; writing—original draft preparation, C.T.; writing—review and editing, C.T.; visualization, Y.Z.; supervision, N.L.; project administration, N.L.; funding acquisition, N.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 61433012, and in part by the Innovation Environment Construction Special Project of Xinjiang Uygur Autonomous Region under Grant PT1811.

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

Abbreviations

The following abbreviations are used in this manuscript:

IDS	Intrusion Detection System
AE	Autoencoder
VAE	Variational Autoencoder
DAE	Denosing Autoencoder
R2L	Remote-to-Local
U2R	User-to-Root
Dos	Denial-of-Service
TP	True Positives
TN	True Negatives
FP	False Positives
FN	False Negatives

References

1. Ahmed, M.; Mahmood, A.N.; Hu, J. A survey of network anomaly detection techniques. *J. Netw. Comput. Appl.* **2016**, *60*, 19–31. [\[CrossRef\]](#)
2. Abuadlla, Y.; Kvascev, G.; Gajin, S.; Jovanovic, Z. Flow-based anomaly intrusion detection system using two neural network stages. *Comput. Sci. Inf. Syst.* **2014**, *11*, 601–622. [\[CrossRef\]](#)
3. Liu, W.; Ci, L.; Liu, L. A New Method of Fuzzy Support Vector Machine Algorithm for Intrusion Detection. *Appl. Sci.* **2020**, *10*, 1065. [\[CrossRef\]](#)
4. Maalouf, M.; Homouz, D.; Trafalis, T.B. Logistic regression in large rare events and imbalanced data: A performance comparison of prior correction and weighting methods. *Comput. Intell.* **2018**, *34*, 161–174. [\[CrossRef\]](#)
5. Bhattacharya, S.; Krishnan, S.S.R.; Maddikunta, P.K.R.; Kaluri, R.; Singh, S.; Gadekallu, T.R.; Alazab, M.; Tariq, U. A Novel PCA-Firefly Based XGBoost Classification Model for Intrusion Detection in Networks Using GPU. *Electronics* **2020**, *9*, 219. [\[CrossRef\]](#)
6. Li, Z.; Gurgel, H.; Dessay, N.; Hu, L.; Xu, L.; Gong, P. Semi-Supervised Text Classification Framework: An Overview of Dengue Landscape Factors and Satellite Earth Observation. *Int. J. Environ. Res. Public Health* **2020**, *17*, 4509. [\[CrossRef\]](#)
7. Malowany, D.; Guterman, H. Biologically Inspired Visual System Architecture for Object Recognition in Autonomous Systems. *Algorithms* **2020**, *13*, 167. [\[CrossRef\]](#)
8. Shankar, K.; Elhoseny, M.; Lakshmanaprabu, S.K.; Ilayaraja, M.; Vidhyavathi, R.M.; Elsoud, M.A.; Alkhambashi, M. Optimal feature level fusion based ANFIS classifier for brain MRI image classification. *Concur. Comput. Pract. Exp.* **2020**, *32*, e4887.
9. Almiani, M.; Abughazleh, A.; Alrahayfeh, A.; Atiewi, S.; Razaque, A. Deep recurrent neural network for IoT intrusion detection system. *Simul. Model. Pract. Theory* **2020**, *101*, 102031.
10. Congyuan, X.; Jizhong, S.; Xin, D. A Method of Few-Shot Network Intrusion Detection Based on Meta-Learning Framework. *IEEE Trans. Inf. Forensics Secur.* **2020**, *15*, 3540–3552.
11. 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; pp. 1–6.
12. Alqatf, M.; Lasheng, Y.; Alhabib, M.; Alsabahi, K. Deep Learning Approach Combining Sparse Autoencoder With SVM for Network Intrusion Detection. *IEEE Access* **2018**, *6*, 52843–52856. [\[CrossRef\]](#)
13. Khraisat, A.; Gondal, I.; Vamplew, P.; Kamruzzaman, J.; Alazab, A. Hybrid Intrusion Detection System Based on the Stacking Ensemble of C5 Decision Tree Classifier and One Class Support Vector Machine. *Electronics* **2020**, *9*, 173. [\[CrossRef\]](#)
14. Tchakoucht, T.A.; Ezziyyani, M. Multilayered Echo-State Machine: A Novel Architecture for Efficient Intrusion Detection. *IEEE Access* **2018**, *6*, 72458–72468. [\[CrossRef\]](#)
15. Dey, S.K.; Rahman, M.M. Effects of Machine Learning Approach in Flow-Based Anomaly Detection on Software-Defined Networking. *Symmetry* **2019**, *12*, 7. [\[CrossRef\]](#)
16. Yang, K.; Liu, J.; Zhang, C.; Fang, Y. Adversarial Examples Against the Deep Learning Based Network Intrusion Detection Systems. In Proceedings of the 2018 IEEE Military Communications Conference (MILCOM), Los Angeles, CA, USA, 29–31 October 2018; pp. 559–564.
17. Yin, C.; Zhu, Y.; Fei, J.; He, X. A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks. *IEEE Access* **2017**, *5*, 21954–21961. [\[CrossRef\]](#)
18. Lotfollahi, M.; Siavoshani, M.J.; Zade, R.S.; Saberian, M. Deep Packet: A Novel Approach For Encrypted Traffic Classification Using Deep Learning. *Soft Comput.* **2020**, *24*, 1999–2012. [\[CrossRef\]](#)
19. Zavrak, S.; iskefiyeli M. Anomaly-Based Intrusion Detection From Network Flow Features Using Variational Autoencoder. *IEEE Access* **2020**, *8*, 108346–108358. [\[CrossRef\]](#)
20. 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. [\[CrossRef\]](#)
21. Devan, P.; Khare, N. An efficient XGBoost–DNN-based classification model for network intrusion detection system. *Neural Comput. Appl.* **2020**, *32*, 12499–12514. [\[CrossRef\]](#)

22. Ke, G.; Meng, Q.; Finley, T.W.; Wang, T.; Chen, W.; Ma, W.; Qiwei, Y.; Liu, T. LightGBM: A highly efficient gradient boosting decision tree. In *Neural Information Processing Systems*; Neural Information Processing Systems Foundation: Long Beach, CA, USA, 2017.
23. Hinton, G.E.; Salakhutdinov, R. Reducing the dimensionality of data with neural networks. *Science* **2006**, *313*, 504–507. [[CrossRef](#)]
24. Doersch, C. Tutorial on Variational Autoencoders. *arXiv* **2016**, arXiv:1606.05908.
25. Lee, S.M.; Kim, H.J.; Kim, S.B. Dynamic dispatching system using a deep denoising autoencoder for semiconductor manufacturing. *Appl. Soft Comput.* **2020**, *86*, 105904. [[CrossRef](#)]
26. Wan, F.; Guo, G.; Zhang, C.; Guo, Q.; Liu, J. Outlier Detection for Monitoring Data Using Stacked Autoencoder. *IEEE Access* **2019**, *7*, 173827–173837. [[CrossRef](#)]
27. Zhou, Y.; Qin, R.; Xu, H.; Sadiq, S.; Yu, Y. A Data Quality Control Method for Seafloor Observatories: The Application of Observed Time Series Data in the East China Sea. *Sensors* **2018**, *18*, 2628. [[CrossRef](#)]
28. Langer, M.; Hall, A.; He, Z.; Rahayu, W. MPCA SGD—A Method for Distributed Training of Deep Learning Models on Spark. *IEEE Trans. Parallel Distrib. Syst.* **2018**, *29*, 2540–2556. [[CrossRef](#)]

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).