Journal of *Marine Science and Engineering*

MDPI

Article

A Marine Hydrographic Station Networks Intrusion Detection Method Based on LCVAE and CNN-BiLSTM

Tianhao Hou [1], Hongyan Xing [1,*], Xinyi Liang [1], Xin Su [2] and Zenghui Wang [3]

[1] Jiangsu Key Laboratory of Meteorological Observation and Information Processing, Nanjing University of Information Science & Technology, Nanjing 210044, China
[2] College of IoT Engineering, Hohai University, Changzhou 213022, China
[3] Department of Electrical and Mining Engineering, University of South Africa, P.O. Box 392, Florida 1709, South Africa
* Correspondence: xinghy@nuist.edu.cn

Abstract: Marine sensors are highly vulnerable to illegal access network attacks. Moreover, the nation's meteorological and hydrological information is at ever-increasing risk, which calls for a prompt and in depth analysis of the network behavior and traffic to detect network attacks. Network attacks are becoming more diverse, with a large number of rare and even unknown types of attacks appearing. This results in traditional-machine-learning (ML)-based network intrusion detection (NID) methods performing weakly due to the lack of training samples. This paper proposes an NID method combining the log-cosh conditional variational autoencoder (LCVAE) with convolutional the bi-directional long short-term memory neural network (LCVAE-CBiLSTM) based on deep learning (DL). It can generate virtual samples with specific labels and extract more significant attack features from the monitored traffic data. A reconstructed loss term based on the log-cosh model is introduced into the conditional autoencoder. From it, the virtual samples are able to inherit the discrete attack data and enhance the potential features of the imbalance attack type. Then, a hybrid feature extraction model is proposed by combining the CNN and BiLSTM to tackle the attack's spatial and temporal features. The following experiments evaluated the proposed method's performance on the NSL-KDD dataset. The results demonstrated that the LCVAE-CBiLSTM obtained better results than state-of-the-art works, where the accuracy, F1-score, recall, and FAR were 87.30%, 87.89%, 80.89%, and 4.36%. The LCVAE-CBiLSTM effectively improves the detection rate of a few classes of samples and enhances the NID performance.

Keywords: network intrusion detection; marine information security; deep learning

Citation: Hou, T.; Xing, H.; Liang, X.; Su, X.; Wang, Z. A Marine Hydrographic Station Networks Intrusion Detection Method Based on LCVAE and CNN-BiLSTM. *J. Mar. Sci. Eng.* **2023**, *11*, 221. https://doi.org/10.3390/jmse11010221

Academic Editor: Christos Tsabaros

Received: 12 December 2022
Revised: 7 January 2023
Accepted: 11 January 2023
Published: 14 January 2023

1. Introduction

Regional marine hydrographic information (RMHI) includes sea temperature, ocean current conditions, and tidal cycles and their intensity, which are intimately related to the navigation conditions for vessels [1]. Therefore, RMHI is regarded as vital strategic information by various countries worldwide. In general, RMHI is usually monitored for normalization by permanent or semi-permanent automatic hydrographic monitoring stations (AHMSs) [2]. Only based on publicly available data, the U.S. National Oceanic and Atmospheric Administration (NOAA) has deployed more than 20,000 AHMSs in the global ocean as of 2022. Meanwhile, other countries are also accelerating the deployment of AHMSs with different specifications [3].

With the rapid development of 5G technology and IoT technology in recent years, automation and networking have become the most-common working mode of AHMSs [4,5]. Ordinarily, in a sea area, several adjacent AHMSs can be formed into a monitoring network [6]. In this network, the stations communicate with each other through wireless protocols, complete the initial data analysis, and package the data to upload to the cloud.

This operation mode helps improve the robustness of RMHI, reduce the load on the network, and control the cost of data storage and downloading [3].

It is difficult to maintain hydrological monitoring networks (HMNs) deployed in remote oceans. The cloud computing strategy plays an important role in HMNs [7]. According to cloud computing, remote servers are able to store, read, and analyze the data. Because of this, highly automated HMNs are vulnerable to various forms of nefarious attacks (including man-in-the-middle attacks, network sniffing, denial-of-service attacks, port scanning, and other forms of attack) [8]. Once the HMN is under these attacks, confidential RMHI may be leaked, threatening national strategic security. In this context, an intrusion detection system (IDS), which enables detecting and alerting about network intrusions by identifying network traffic data (NTD), is essential in the HMN.

In terms of detection devices, host-based IDSs (HIDSs) and network-based IDSs (NIDSs) are the most-popular types of network intrusion detection (NID) systems [9]. The former are mainly deployed in local devices (e.g., PCs, database devices, etc.) and detect network intrusions by analyzing the operation behavior of the files [10]. NIDSs are generally used to prevent network attacks against the server layer. There are two typical operation strategies for NIDSs, signature and anomaly [11]. The former defines a set of rules to determine whether a network behavior is an intrusion based on the description from known attack behavior features. This approach performs well for known attacks, but weakly detects unprecedented attacks (0-day attacks). Anomaly NIDSs analyze a large amount of normal NTDSs to obtain the best features of the normal network behavior. They are not sensitive to the attack occurrence frequency, obtain better generalization ability, and have gained wide attention in recent years.

NIDs can be regarded as a classification problem with respect to network behavior. In the past two decades, much research has been carried out in this area to propose a more advanced intrusion detection method [12,13]. Some Euclidean-distance-based ML methods were first applied to this field, such as k-nearest neighbors (KNNs), support vector machine (SVM), and logistic regression (LR) [8,14,15]. These methods may have satisfactory performance in a simple network environment, but because they cannot effectively trace the deeper features of the intrusion behavior, they are not suited to handling today's diverse attacks [16].

Since the 2010s, DL technology has developed significantly, and the GAN, SAE, DNN, and other related algorithms are widely used in NIDs [17,18]. DL-based detection methods are adept at extracting abstract features from massive amounts of NTDs. They overcome the limitations of shallow ML algorithms and provide a new way of more accurate intrusion detection. However, for AHMSs, these methods generally suffer from the following two drawbacks:

(1) DL cannot address unbalanced intrusion samples. The actual NTD database often contains a large number of normal behaviors and a small number of attacks. At the same time, even with this minority of attacks, it may be filled with a number of 0-day attacks. This is fatal for DL methods that heavily rely on large amounts of data for training. They may not be able to obtain enough effective features in the training phase, especially for those 0-day attacks [19]. AHMSs typically work in a unattended and open environment. They are exposed to more diverse security threats, and AHMSs are vulnerable to new types of cyber attacks at all times. Therefore, the effectiveness of traditional DL-based detection methods is a concern for AHMSs.

(2) DL fails to consider the time cost of intrusion detection [20]. Since the actual NTD generally has high-dimensional characteristics (above 40 dimensions), complex detection neural networks incur a huge computational load, which takes much time and even affects the real-time performance of detection. On the other hand, due to cost considerations, AHMSs generally do not have high-performance master control units. Therefore, they are extremely inefficient in handling complex intrusion detection algorithms. We are very clear that too delayed detection results are meaningless, and the damage caused by the attack may have been irreversible.

In order to solve the dilemma faced by AHMSs and secure RMHI transmission and storage, this paper proposes a BiLSTM combining the CNN NID method based on the LCVAE. As show in Figure 1 , the overall structure of the LCVAE-CBiLSTM, which consists of two main parts, the encoder and the classification. The former can be regarded as an extended model of the VAE and CVAE. A basic VAE consists of an encoder and a decoder to model the distribution of observed data by latent variables in an unsupervised manner [21]. On this basis, the CVAE can refer to the sample feature vector by itself to guide the encoder generation process and obtain more efficient features in the potential space [22]. More importantly, it can generate new samples given conditional information (labels). In the classification part, the CNN is used to extract the data features in space. Considering the NTD's potential sequential features, BiLSTM can extract its temporal features further. Moreover, this model will reduce the data dimension and control the detection time [23,24].

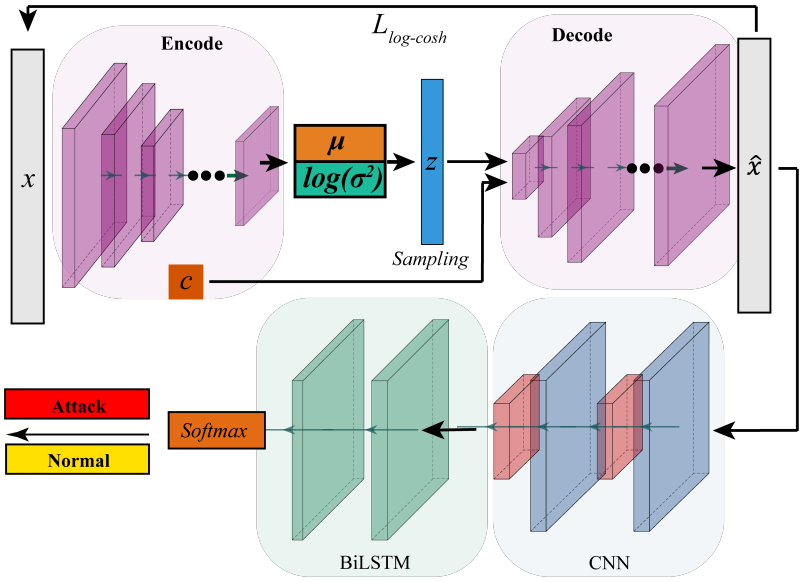


Figure 1. Overall framework of the LCVAE-CBiLSTM.

In summary, the main contributions of this paper are as follows:

(1) We introduced the CVAE to the NID for reconstructing attack samples in the NTD. On the basis of the CVAE, we constructed the loss function using the log-cosh function. This enables the novel LCVAE to better extract the discrete features of the NTD. It also allows the new samples generated by the LCVAE to complement the minority types and provide a database for more accurate intrusion detection.

(2) We constructed a classification model combining the CNN and BiLSTM. Among them, the CNN is suitable for extracting local spatial features and BiLSTM takes into account bi-directional time series information. The model can analyze the NTD in two dimensions, time and space, to extract more significant network behavior features, which provides a guarantee for a more accurate NID. The multiple-feature extraction strategy also effectively controls the data's dimensionality to ensure the intrusion detection's efficiency.

(3) We evaluated the accuracy of the LCVAE-CBiLSTM compared to several other state-of-the-art works in a simulation environment. A miniature hydrological monitor was developed to evaluate the efficiency of different methods on a low-power platform. The experimental results show that our model is superior to other methods in detection performance. This demonstrates that the LCVAE-CBiLSTM provides an effective method for intrusion detection.

The remaining parts of this article are organized as follows. Section 2 introduces the related works. Section 3 describes the proposed algorithm's theory in detail. Section 4 shows the experimental setup, experimental results, and discussion. Section 5 provides the conclusion of the work, as well as future work.

2. Related Works

Machine learning strategies have been long applied to improve IDS through hybrid models. Zhao proposed a method based on the hybrid kernel function least-squares support vector machine (LSSVM) [25]. He introduced a particle swarm optimization (PSO) algorithm to find the optimal parameters in the LSSVM, which effectively improved the accuracy of intrusion detection. Tao used a genetic algorithm (GA) to select features, weights, and parameters in SVM [26]. Compared to the traditional SVM approach, his strategy had better performance in terms of the accuracy rate and the false alarm rate. In order to improve the detection efficiency, Peng filtered the more significant traffic features with the network behavior by principal component analysis (PCA) first and, then, identified the attack type by K-means [27].

Traditional ML methods all belong to shallow learning. Along with the development of the Internet, the large number of more complex and hidden nonlinear network features brings new challenges to NID. DL algorithms are able to extract high-level abstract features from the original samples through multiple iterations. Therefore, they will effectively overcome the limitations of ML methods and improve detection accuracy. The use of DL methods for NID has become a trend in recent years [28].

Some neural-network-based classification models have been proposed. Ingre deployed artificial neural networks (ANNs) for IDSs, but the experimental results on a publicly available network traffic dataset, NSL-KDD, did not show model superiority [29]. He also tried to propose several hybrid detection methods combining different algorithms, but the results still needed improvement. Yan proposed a greedy multilayer deep belief network model (GDBN) [30]. The model focuses on using the restricted Boltzmann machine (RBM) to separate noisy and anomalous data in the samples [31]. A back-propagation strategy was used to control the training parameters in the DBN, which in turn implemented the NID. Tang selected six basic features from the NSL-KDD dataset using the software-defined networking (SDN) strategy [32]. These features support effective network behavior classification by a simple deep neural network (DNN). The above DL-based NID approaches ignore the imbalance of samples in NID, which has resulted in no significant improvement in the detection performance for IDS for a long time.

At present, there are two main types of solutions for the imbalanced characteristics of the samples: algorithmic compensation and data compensation. The former considers the model's sensitivity to different intrusion types and improves the detection accuracy by designing the attack detection weights [33]. However, since the weights of 0-day attacks are unknown, the algorithmic compensation method is constantly weak when dealing with them [34]. Data compensation reduces the degree of type imbalance in modifying the original sample distribution. Simply put, it adds minority-type samples or removes a part of the normal samples. Common methods are the random oversampler (ROS) [35], synthetic minority oversampling technique (SMOTE) [36], variational autoencoder (VAE) [37], etc. DL can extract additional significant intrusion features from minority types that have already been increased. Numerous experiments have shown that combining data compensation and DL is a unique NID method. This means that this hybrid technique has much promise for study in intrusion detection.

The VAE is a feature extraction algorithm derived from the autoencoder (AE) [37,38]. The AE uses a neural network to fit a mapping relationship from sample x to a potentially encoded sample z. For a high-dimensional x, the structure of z may be straightforward. Since the mapping relationship between x and z is unique, the AE does not work for data generation. In NID, the AE is more often used for feature dimensionality reduction. The VAE assumes $x \rightarrow z$ tallies with a normal distribution. After describing the encoding process of z by means and variances, the new data can be obtained by sampling from them. However, the VAE uses an unsupervised learning strategy, and we cannot control the generated data by the distribution of the samples. There is not much one VAE can do to solve the problem of attack imbalance. To address the problem, Kingma proposed the conditional variational autoencoder (CVAE) [22]. It establishes a distribution relationship

between sample labels c and z based on the VAE. In other words, the class of z can be determined by c. This feature makes the decoder of the CVAE generate the specified data type. It is of great significance for NID because it enables expanding minority attacks in the original sample and helps the intrusion detection algorithm dig out deeper features.

The CVAE-based NID model designed by Martin is poor in intrusion detection, but he validated that the data generation capability of the CVAE can be used in IDS [39]. Hannan was not interested in data generation. He focused on a semi-supervised CVAE-based feature extraction method by marking attack data in the training set [40]. The experimental results showed that the CVAE was more satisfactory for the detection on the CIC-IDS2017 dataset than other traditional methods, but less effective for NSL-KDD. Liu compared the detection effectiveness of the VAE and CVAE for NTDs in the setting of the CNN and RNN as the classifiers [41]. In the experiments for the CSE-CIC-IDS2018 dataset, he found that the CVAE can effectively improve the detection of NIDs from multiple scales such as the precision, recall, and F1-score. However, some scholars have pointed out some drawbacks of the CVAE deployed on NID. (1) The CVAE suffers from KL vanishing in training, which leads to the model's failure to converge [42]. (2) The L2 loss function in the traditional CVAE tends to result in local optima and is sensitive to noise [43]. In this context, many CVAE-based improvement models have been proposed. For example, the supervised adversarial variational autoencoder with regularization (SAVAER), the classwise focal loss variational autoencoder (CFLVAE), the conditional denoising adversarial autoencoder (CDAAE), etc., but these methods failed to bring better performance to NID [43–45].

Recently, related scholars have attempted to combine multiple DL methods into a hybrid model to improve intrusion detection performance. Hawawreh proposed an NID model combining a deep autoencoder with a deep feed-forward neural network (DAE-DFFNN), which validated the possibility of hybrid models [46]. Zhang proposed the CWGAN-CSSAE, which combines the improved conditional Wasserstein generative adversarial network and cost-sensitive stacked autoencoders. The experimental results on the KDDTest-21 showed that the detection accuracy was improved by 2–5% compared to other simple DL models [47].

Combining the advantages of the above methods, this paper first proposes an LCVAE model, which avoids the problem of KL divergence for insufficient control of the model by introducing a log-cosh function to reconstruct the loss term of the CVAE. Then, we expand the original dataset with the few classes of attack samples generated by the LCVAE to provide sufficient training conditions for the classifier. Finally, a hybrid detection model, the CNN-BiLSTM, is constructed to ensure intrusion detection accuracy.

3. Methods

The flowchart of the proposed LCVAE-CBiLSTM method is shown in Figure 2 and consists of 4 steps:

Step 1: numerical processing. To make the intrusion detection model more accurate in identifying the data features, at the beginning of all processes, performing the necessary pre-processing of the original data is necessary. It first digitizes the character-based features and, then, normalizes each feature. Then, it separates the training set and testing set from the dataset.

Step 2: LCVAE-based sample generator for minority class attacks. DL-based detection algorithms are usually insensitive to a few classes of samples. We propose an improved CVAE algorithm for generating minority class attack samples to enforce their missing features. The training set data are fed into the model to obtain the newly generated enhanced data.

Step 3: Hybrid feature extraction strategy. In order to comprehensively extract features from traffic data that are more closely associated with attack behavior, we designed a hybrid feature extraction strategy. In this strategy, the CNN is used to extract the spatial features of the samples, and the BiLSTM is used to extract the temporal features. The training set

and the enhanced data are fed into the proposed algorithm after the iterations of training to obtain a high-performance feature extraction model.

Step 4: Obtain test results. We input the testing set into the model obtained by Step 3 to extract the deep features of the samples. Finally, a Softmax algorithm completes the intrusion decision based on these features.

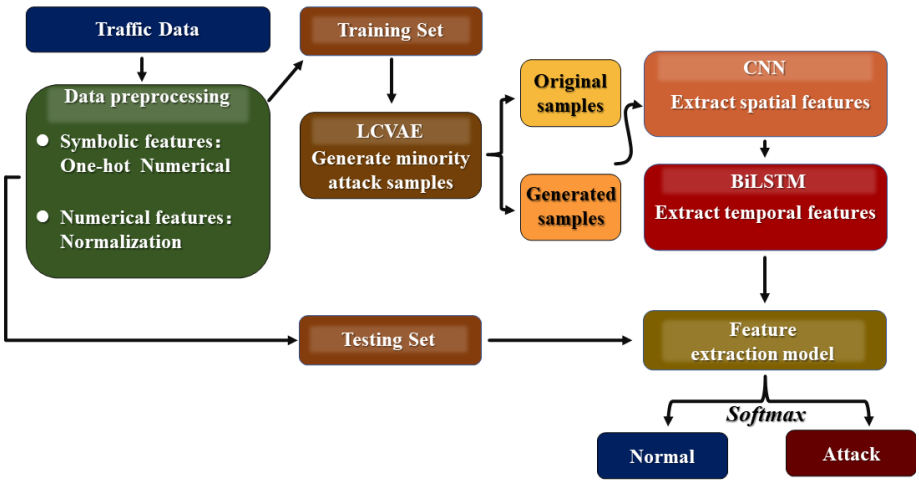


Figure 2. Flow chart of the LCVAE-CBiLSTM method.

3.1. Data Preprocessing

Data preprocessing includes two steps as follows:

Numerical processing.

The three symbolic features (protocol_type, service protocol, and flag protocol) in NSL-KDD need to be converted into digital features to facilitate model training. We used one-hot encoding to accomplish this task. The above three features are converted into 3, 70, and 11 numeric features. Together with the original 38 numeric features, the original 41-dimensional features become 122-dimensional after numerical processing. It is worth noting that the feature *num_outbound_cmds* is all 0, which can be considered a redundant feature [29]. Therefore, we removed it and finally obtained 121-dimensional features.

Normalization processing.

In NTDs, the difference in the range of values of various features may be too huge. For example, in NSL-KDD, the *num_root* feature takes a range of [0,7468], while the *num_shells* feature only takes a range of [0, 5]. In DL, the exaggerated range of feature values would lead to the model not being able to converge. Therefore, we completed the normalization process for all the features by mapping each feature within the [0, 1] interval uniformly and linearly. The normalization process for variable X is shown in Equation (1).

$$X_{\text{nom}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}} \tag{1}$$

In Equation (1), $X_{\max}$ is the maximum value of the sample data, $X_{\min}$ is the minimum value of the sample data, and X_{nom} is the data after normalization.

3.2. Log-Cosh Conditional Variational Autoencoder

The VAE and CVAE are the prerequisites for understanding the mechanism of the LCVAE. Assume that the original sample is x and the generated sample by the model is $\hat{x}$. A typical CVAE model is shown in Figure 3, which adds a conditional factor c (which can be interpreted as the sample's label) to the VAE. The target of the encoder Q_ϕ in the VAE is to establish a mapping relationship from the original high-dimensional sample x to the low-dimensional feature vector z through a neural network. In Figure 3, the CVAE introduces c as a selection condition for x, at which point, the mapping relationship from x to z is

$z \sim Q_\phi(z \mid x, c)$. In contrast, the decoder P_θ constructs a neural network $\hat{x} \sim P_\theta(x \mid z, c)$ that maps z and c back to $\hat{x}$. The objective loss function of the VAE is shown in Equation (2).

$$\begin{aligned} L_{VAE} =& \mathbb{E}_{Q(z|x)}[\log p(x)] \\ =& \underbrace{\mathbb{E}_{Q(z|x)}[\log P(x \mid z)]}_{\text{Reconstruct term } L_{\text{Rec}}} - \underbrace{D_{KL}(Q(z \mid x) \| P(z))}_{KL \text{ term } L_{KI}} \end{aligned} \tag{2}$$

Then, the CVAE loss function model can be introduced as:

$$\begin{aligned} L_{CVAE} =& \mathbb{E}_{Q(z\|x)}[\log P(x \mid c)] \\ =& \underbrace{\mathbb{E}_{Q(z\|x,c)}[\log P(x \mid z, c)]}_{\text{Reconstruct term } L_{\text{Rec}}} - \underbrace{D_{KL}(Q(z \mid x, c) \| P(z \mid x, c))}_{KL \text{ term } L_{KI}} \end{aligned} \tag{3}$$

where the loss function consists of two components. (1) The first term, $\mathbb{E}_{Q(z|x,c)}[\log P(x \mid z, c)]$, is a logarithmic reconstruction likelihood function that describes the correct rate of compressing high-dimensional features into a low-dimensional space and, then, restoring them correctly. (2) The second term, $D_{KL}(Q(z \mid x, c) \| P(z, c))$, evaluates the performance of the encoder and decoder by constructing the KL scatter of the prior and posterior distributions of z. Obviously, the training process of the CVAE is to ensure the maximization of the reconstruction term and the minimization of the KL scatter term in Equation (3). Due to the presence of the reconstruction term $P(x \mid z, c)$, the type of new samples generated can be selected by controlling z and c.

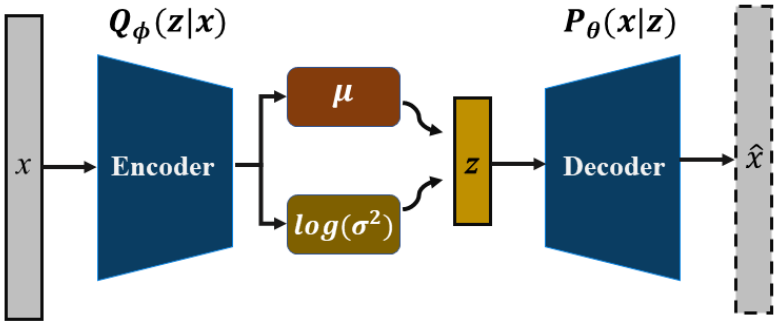


Figure 3. A typical VAE model.

From Equation (3), the reconstruction term is constructed based on the L_2 function, as shown in Equation (4).

$$L_{\text{Rec}}(x, \hat{x}) = \|x - \hat{x}\|_2^2 = \sum_i |x_i - \hat{x}|^2 \tag{4}$$

In Equation (4), x_i denotes the ith feature in x. Two shortcomings of L_2 limit the performance of the reconstructed model. L_2 will generate a large loss value when x is significantly higher than $\hat{x}$. It may cause a gradient explosion, and the reconstructed model may not converge effectively. Furthermore, the gradient explosion will seriously increase the time complexity and lead to a huge computational overhead of the model. Because of the presence of the squared term in L_2, when the loss value is small (<1), L_2 may ignore part of this error. It would cause the reconstruction model of the CVAE to be completely controlled by KL and even fall into the local optimum. Chen argued that this reconstruction model significantly weakens the accuracy of sample generation when dealing with high-dimensional features [48].

To solve the problem that the L_2 limits the performance of the CVAE, we propose the log-cosh function as the loss term of the reconstruction, as Equation (5):

$$L_{\log-\cosh}(x, \hat{x}) = \frac{1}{a} \sum_i \log(\cosh(a(x_i - \hat{x}_i))) \tag{5}$$

Equation (5) in $a \in R$ is a parameter. The images of log-cosh and L_2 are shown in Figure 4. It can be seen that, when the difference between x and $\hat{x}$ is large, the loss value of the log-cosh is much smaller than L_2, which can effectively avoid the gradient explosion. On the contrary, when the loss is close to 0, the log-cosh gradient decreases, ensuring the model's accuracy.

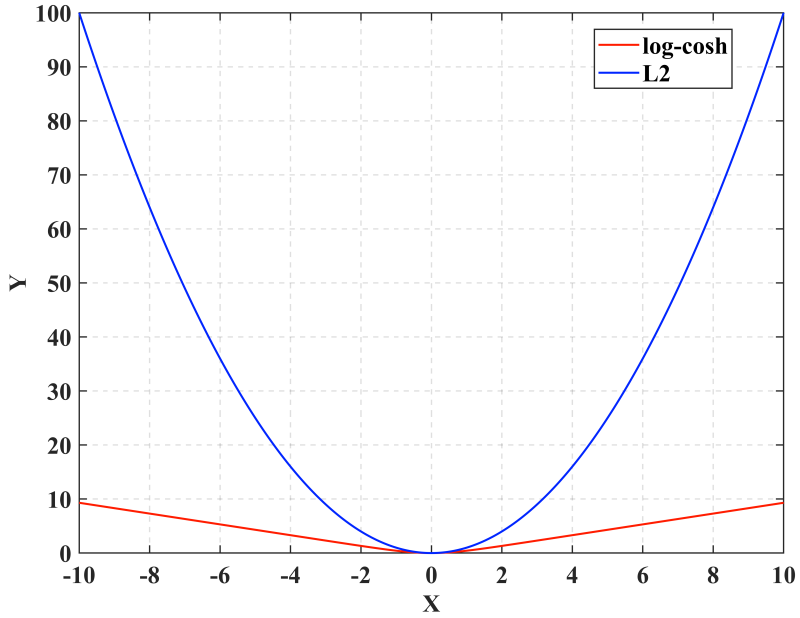


Figure 4. Images of log−cosh and L_2 functions.

When the log-cosh replaces the reconstructed loss term of the CVAE, the proposed LCVAE model is shown by Equation (6).

$$L_{\text{LCVAE}} = \frac{1}{a}\sum_{i}\log(\cosh(a(x_i - \hat{x}_i))) - D_{KI}(Q(z \mid x, c) \| P(z \mid x, c)) \tag{6}$$

The percentage of certain attack types might be quite low because there is an imbalance in the types of network traffic. For example, samples belonging to the remote-to-local and user-to-root account for less than 1% of the KSL-KDD dataset. Most intrusion detection algorithms are powerless to deal with similar minority classes. Even though they are non-converging when training the detection model, the meager ratio cannot affect the model's overall training results. This will cause the detection models to ignore the detection sensitivity of these types.

Similar to the traditional CVAE, in the LCVAE, Equation (6) is used as the loss function to train the generator. We first chose c as belonging to the minority types in the dataset generation phase. The appropriate z is determined by the normal distribution of c and fed back to the decoder network to generate new intrusion samples. By adding the generated $\hat{x}$ to the original dataset, the minority types sample in the whole dataset may be increased to alleviate the data category imbalance.

3.3. Temporal Features Extraction by BiLSTM

Considering the potential temporal correlation of NTD, a neural network model combining the BiLSTM [19], as shown in Figure 5, was established to extract the temporal features.

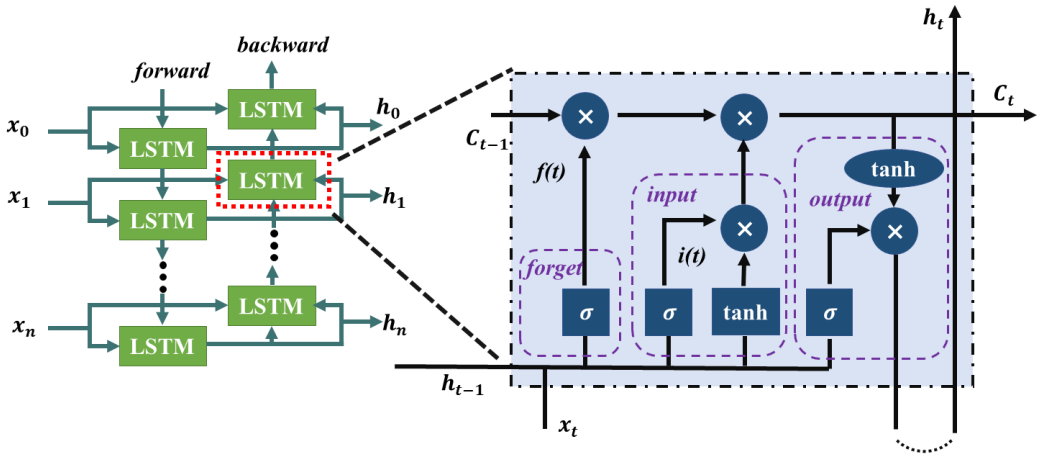


Figure 5. BiLSTM network structure model.

As can be seen from Figure 5, the BiLSTM is formed by two inverse LSTM networks. The single LSTM network consists of multiple neuron nodes. Each neuron node contains three decision-makers: the input gate, the output gate, and the forget gate. The computational flow of each neuron node in the LSTM is as follows.

The state h_{t-1} of the hidden layer at the moment $t-1$ and the input x_t at moment t are fed into the forget gate of a new neuron node. Equation (7) determines whether the information of the current input x_t will be updated in the neuron, where W is the weight matrix and b is the decision preference.

$$i_t = \sigma(W_t \cdot [h_{t-1}, x_t] + b_t) \tag{7}$$

At this point, the input gate also generates $\tilde{C}_t$ to describe the temporary state of the neuron node, as shown in Equation (8).

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{8}$$

Then, the input gate will determine the information that the neuron node at moment t will retain from h_{t-1} based on x_t, as shown in Equation (9).

$$f_t = \sigma\Big(W_f \cdot [h_{t-1}, x_t] + b_f\Big) \tag{9}$$

The forget gate selects the information that needs to be updated from the node's state at $t-1$. The input gate will control the temporary state that needs to be retained. The current node state C will be obtained by associating the node states of these two temporal states:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{10}$$

Finally, the output gate will update the state of the hidden layer, as shown in Equation (11).

$$h_t = \sigma(W_o[h_{t-1}, x_t] + b_o) * \tanh(C_t) \tag{11}$$

In Figure 5, two LSTM networks are responsible for the forward and backward feature extraction, respectively. Using the BiLSTM model provides a better consideration of the effect of each attribute before and after in the sequence data and obtains more comprehensive feature information. The BiLSTM states at time t include the forward and backward outputs, as shown in Equations (12)–(14).

$$h_t^{\text{forward}} = LSTM^{\text{forward}}(h_{t-1}, x_t, C_{t-1}) \tag{12}$$

$$h_t^{\text{backward}} = LSTM^{\text{backward}}(h_{t-1}, x_t, C_{t-1}) \tag{13}$$

$$H_t = \left[h_t^{\text{forward}}, h_t^{\text{backward}} \right] \tag{14}$$

3.4. Convolutional Neural Network

Besides temporal features, the huge amount of NTD data also has significant spatial features. Therefore, we designed a CNN model before the BiLSTM to extract the spatial distribution features. Furthermore, the multi-layer convolution and pooling process enables the downsizing of the features and improves the computational efficiency of the BiLSTM network.

In the CNN, the NTDs are firstly spread into a matrix of size n*m after pre-processing. This matrix is input into the convolution layer, and the convolution process is shown in Equation (15).

$$Z = f(M \otimes W + b) \tag{15}$$

where M is the feature map. b is the bias. $\otimes$ is the convolution function, and $f(x)$ is the activation function. $f(x)$ helps to reduce the linearity of the model and improve the performance of model feature extraction.

It is important to create a nonlinear mapping for each convolutional kernel's convolution result in order to facilitate the computation. Here, we used the RULE activation function, as show in Equation (16).

$$f(x) = \text{RULE} = \max(0, x) \tag{16}$$

Then, the activated feature vectors are integrated by a pooling layer. The maximum value is kept in a window of a certain size to reduce the size of the output features and avoid transition fitting.

After several convolutional and pooling processes, the output of the CNN is reshaped into a one-dimensional vector by one fully connected layer. This one-dimensional vector is the one that contains the spatial features of the NTD.

3.5. CNN-BiLSTM Feature Extraction Model

After introducing the CNN, the CNN-BiLSTM network model as shown in Figure 6 is established to extract the NTD's features.

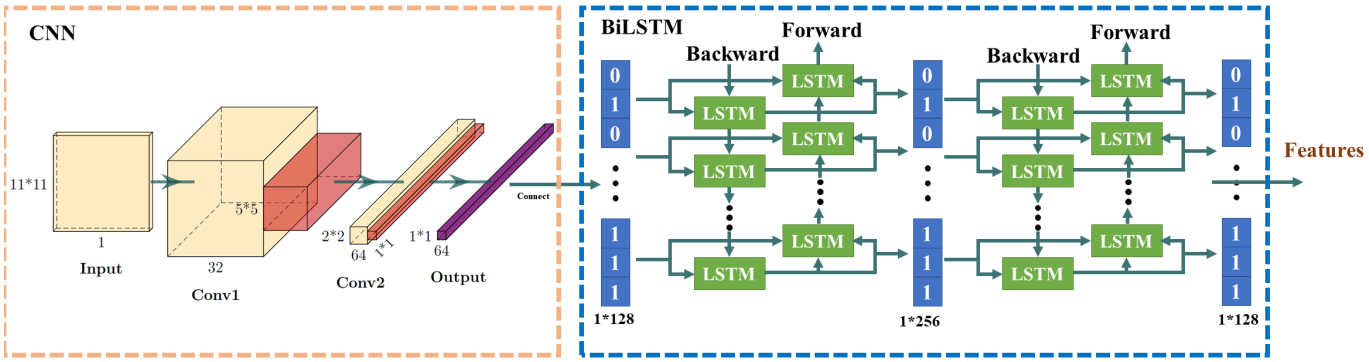


Figure 6. The CNN-BiLSTM feature extraction model.

By combining the CNN and BiLSTM, the layered network model is established to extract the spatiotemporal features of the NTD simultaneously, in which the CNN uses two sets of convolutional–pooling layers to complete the extraction of the spatial features. The strides of Conv1 and Conv2 are 1 and 3, respectively. In these two convolutions, the convolutional kernel size is 3*3, padding is 1, the activation function is tanh, the pooling kernel size is 2*2, and the pooling step is 2. A two-layer BiLSTM model was used for temporal feature extraction. The first BiLSTM contains 128 neurons, and the second contains 256. Each recursive operation of the BiLSTM results in the fusion of all previous

and current features. A fully connected layer is linked to all output layers of the BiLSTM to combine the previously extracted features. Then, the fully connected layer's output value is transmitted to a Softmax classifier to complete the network behavior detection. Finally, according to the extracted features, the network behavior is decided by a Softmax.

4. Experimental Validation and Analyses

This research conducted experiments on a desktop platform to evaluate the effectiveness of the proposed method. The detailed configuration of the platform is shown in Table 1. The experimental target dataset was NSL-KDD.

Table 1. Experimental platform.

Project	Configuration
Operation system	Windows 11
CPU	AMD-5700X 4.7–5.0 Ghz
GPU	NVIDIA 3060TI 8 G
Memory	32 G
Frame	Pytorch

4.1. The Benchmark Datasets

The NSL-KDD dataset is an improved version of the KDDcup99 dataset [49]. It eliminates a large amount of duplicated and redundant data in KDDcup99, improves the sample quality, and has become a widely used intrusion detection dataset in recent years. NSL-KDD records 148,517 NTDs as samples, each sample consisting of 41 features and containing two subsets: (1) KDDTrain+ as a training set containing 125,973 samples, recording 22 types of attacks, and (2) KDDTest+ as a testing set containing 22,544 samples and 17 more types of attacks than KDDTrain+ to serve as 0-day attacks. Table 2 shows the distribution of the NTD samples in these two subsets.

Table 2. Attack type distributions of NSL-KDD.

Category		Description	Data	
			Train	Test
Attack	DoS	Malicious occupation of network resources.	45,927	7458
	Probe	Illegal obtaining users' privacy.	11,656	24,218
	R2L	Hijack from a remote computer.	99	2754
	U2R	Illegal obtaining of users' advanced authorization.	52	200
Normal		Normal connection.	67,343	9711
Total			125,973	22,544

4.2. Evaluation Criteria

We used five metrics: accuracy, precision, recall, F1-score, and false alarm rate (FAR), to evaluate the detection performance of the method. These metrics were calculated from the confusion matrix of the results, as shown in Table 3. For the NID, TP is the number of attack samples that are correctly identified as attacks. FP is the number of normal samples that are incorrectly identified as attacks. TN indicates the number of samples correctly labeled as normal. FN is the number of attack samples that are incorrectly identified as normal. The total number of samples tested is S, $S = TP + FN + FP + TN$.

Table 3. Definition of the confusion matrix.

	Predicted Attack	Predicted Normal
Actual Attack	True Positive (TP)	False Negative (FN)
Actual Normal	False Negative (FN)	True Negative (TN)

The accuracy characterizes the correctly classified samples as a percentage of the total, defined as follows. Accuracy evaluates the detection performance of the model at a macro level. However, it cannot show the model's ability to cope with normal or attack samples separately.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{17}$$

Precision refers to the percentage of correctly classified attack samples among all predicted attack samples, as Equation (18). This measure emphasizes the number of correct samples in the attack outcomes.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{18}$$

Recall is the percentage of correctly predicted attack samples out of all attack samples as defined in Equation (19). It shows the degree of completeness of all attack samples detected.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{19}$$

The F1-score is defined as the average of precision and recall, as shown in Equation (20), and is more suitable for evaluating the overall intrusion detection performance of the model.

$$\text{F1} = \frac{2TP}{2TP + FP + FN} \tag{20}$$

The FAR is the percentage of samples that are incorrectly predicted as attacks to all normal samples. This metric shows the models discrimination ability between different samples and measures its generalization.

$$\text{FAR} = \frac{FP}{FP + TN} \tag{21}$$

Among the above metrics, except for the FAR, which should be as low as possible, larger values of the other metrics indicate better model intrusion detection performance.

4.3. LCVAE Performance

4.3.1. Model Parameters

The network depth and the number of neurons per hidden layer are important parameters for the LCVAE. They directly affect the model's performance: if the model is too simple, it may not be able to encode the samples. By contrast, models with more complex structures imply the presence of better capabilities. The data it generates may be closer to the real sample. Therefore, a suitable LCVAE structure needs to be set up to assist the CNN-BiLSTM in detecting intrusions according to the needs of network intrusion detection.

From KDDTrain, we chose 60% of the samples as the training set and the rest as the validation set. We verified the effect of the encoding results on the CNN-BiLSTM by adjusting the number of hidden layers and the neurons of the LCVAE. Figure 7 shows the average detection error for ten experiments.

As shown in Figure 7, positive correlations were between the complexity of the model and the detection performance in general. However, the model performance did not increase linearly with the number of hidden layers. For example, the LCVAE model improved the accuracy by 2.45% when it was boosted from 1 to 3 hidden layers. When the model hidden layer number was raised from 4 to 5, the accuracy improved by only 0.03%.

From the perspective of the time complexity, the number of hidden layers and the time taken to complete the verification are exponentially related. When there is only one hidden layer, the verification process took about 5.5 s. As the number of hidden layers increased, the verification time grew to 12.0 s, 23.1 s, and 64.3 s.

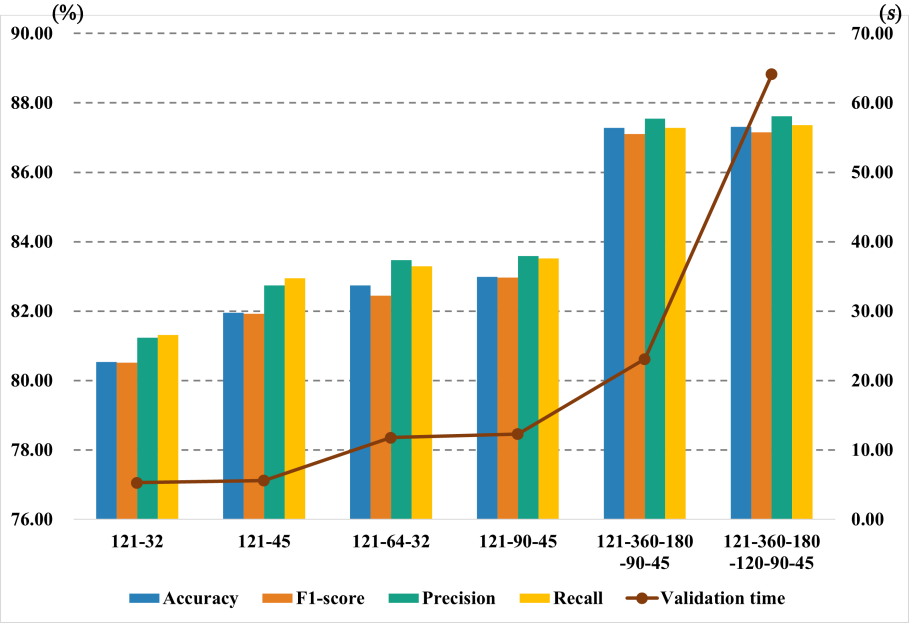


Figure 7. Effect of different LCVAE structures on detection

It is worth noting that, although the most complex one with 5 layers had the best results, the improvement was insignificant (less than 0.03%) compared to the 4-layer one. Moreover, the former took almost three times as long as the latter. Considering the detection results and computational cost, it is reasonable to set the hidden layer's neurons as 121-360-180-90-45, respectively.

4.3.2. Data Generation

As can be seen from Table 2, the number of attack types R2L and U2R was significantly less than other types in NSL-KDD. To enable the CNN-BiLSTM to extract their features effectively, we generated a total of 78,280 minority class samples (among them: 39,800 R2L and 38,480 U2R) based on the model in Section 4.3.1. The sample type percentages are shown in Figure 8.

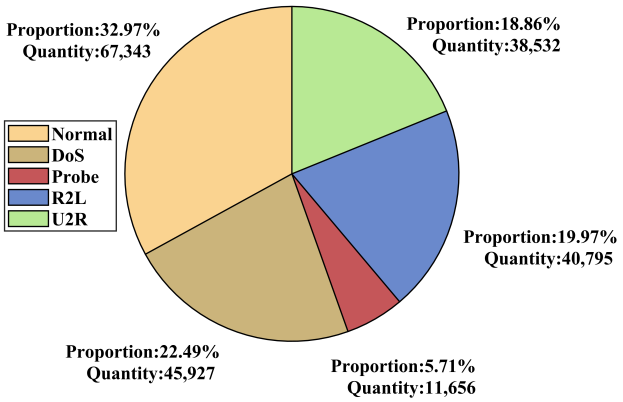
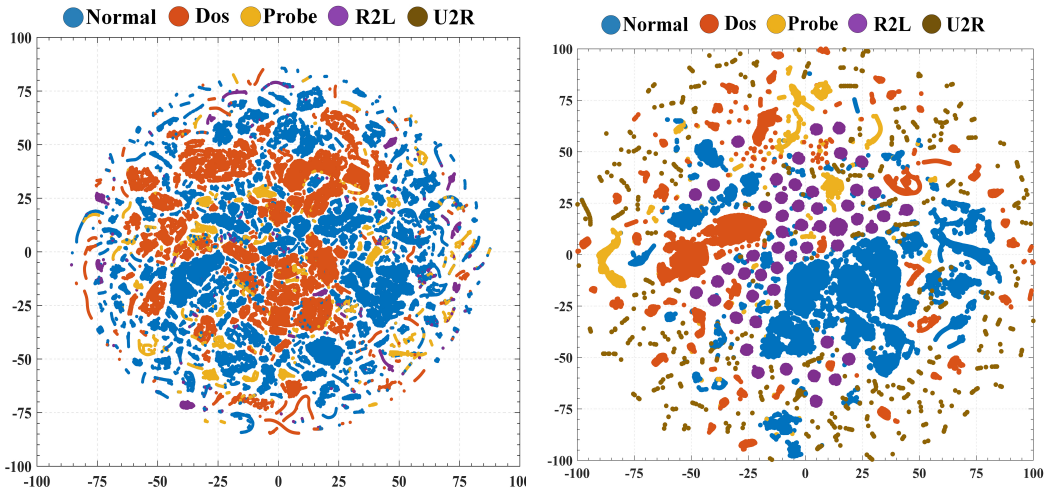


Figure 8. The percentage of different types of samples in the training set.

4.3.3. Data Encoding

Figure 9 shows the distribution of the original dataset and the dataset enhanced by the LCVAE on the T-distributed stochastic neighbor embedding (T-SNE) space. The T-SNE is an unsupervised learning algorithm that maps high-dimensional data to low-dimensional

data. Its clustering results provide an intuitive assessment of the degree of association between the comprehensive features of the samples and the network behavior.



(**a**) Original sample (**b**) Sample after LCVAE

Figure 9. T−SNE visualization of before and after LCVAE encoding results in KDDTrain+.

Figure 9a shows that the original samples are almost randomly distributed in the T-SNE plane. There is no clear boundary between the different sample types of clusters. There was almost no significant trend of aggregation for the same type of samples. A large number of other samples were mixed in the limited type of clustering. For example, many probes are mixed in the DoS. Meanwhile, it can be easily found that the number of R2L and U2R samples was too small to be detected in the T-SNE. These factors are detrimental to the subsequent intrusion detection process.

In Figure 9b, the distribution of the normal, DoS, probe, and R2L sample points show a clear aggregation phenomenon, with clear boundaries separating the clusters. It is not easy to find these four types of sample points that are outside the clusters. There are two interesting phenomena here. (i) Each cluster of R2L has a distinct center. The proves the data generation process, where the center is the original data, and the other points represent the new samples generated by the LCVAE. (ii) The points of U2R are almost randomly distributed over the whole T-SNE space. This phenomenon may be related to the number of U2R in KDDTrain+ (only 52). The tiny training samples caused the LCVAE to not be able to be an encoding strategy for U2R, which caused the generated data to show large deviations from the true values. This can lead to a high error frequency of the classifier in response to such samples.

Overall, the LCVAE-enhanced encoded samples possessed more distinctive category characteristics, which is undoubtedly beneficial for NID.

4.4. Detect Results

Table 4 shows the detection results of the LCVAE-CBiLSTM on KDDTest+. The overall accuracy was 87.30%, the normal accuracy 95.64%, and the attack 80.99%. In the case of the DoS and probe attacks, the LCVAE-CBiLSTM exhibited excellent results, 96.55% and 92.14%. The R2L and U2R showed much poorer detection accuracy for the LCVAE-CBiLSTM than the others, with the former being 32.39% and the latter being 50.75%. The R2L and U2R are regarded as minority samples that are particularly challenging to identify. As a result, the outcome is still acceptable. This suggests that, albeit to a relatively small extent, the LCVAE-CBiLSTM optimizes the detection ability for a minority type.

Table 4. The NID results of the proposed method on KDDTest+

Class		Type (Misclassification/Total)						Accuracy	
Normal				**443/9711**				**95.43%**	
Attack	**DoS**	back	0/359	neptune	2/4656	smurf	0/665	96.55%	80.99%
		teardrop	0/12	land	1/7	pod	0/41		
		apache2	68/737	**mailbomb**	137/293	**processtable**	49/685		
	Probe	satan	1/735	portsweep	0/153	ipsweep	3/141	92.14%	
		nmap	0/73	**mscan**	184/996	**saint**	2/319		
	R2L	warezmaster	365/944	ftpwrite	1/3	multihop	2/18	32.39%	
		guesspassword	1011/1231	imap	0/1	phf	1/2		
		sendmail	4/14	**worm**	2/2	**snmpguess**	320/331		
		named	1/17	**snmpgetattack**	178/178	**xsnoop**	3/4		
		xlock	6/9	**httptunnel**	41/133				
	U2R	rootkit	8/13	bufferoverflow	7/20	loadmodule	1/2	50.75%	
		perl	1/2	**ps**	10/15	**sqlattack**	0/2		
		xterm	6/13						
Total								87.30%	

For the 0-day attacks marked in bold font in Table 4, the LCVAE-CBiLSTM showed a satisfactory detection performance with an overall accuracy of 73.02%. This proves that the LCVAE-CBiLSTM is able to deal with threats from unknown attacks effectively. The detection rates of samples belonging to the 0-day attack in DoS, probe, R2L, and U2R were 85.19%, 85.86%, 19.33%, and 46.88%, respectively. It can be clearly seen that the detection rate of a 0-day attack was lower for samples belonging to a few categories.

Although the detection results of LCVAE-CBiLSTM are satisfactory from a macroscopic point of view, the experiments exhibited a large difference in sensitivity between different attack types. For attack types such as back, neptune, teardrop, and portsweep, their detection accuracy was nearly 100% regardless of the total number of samples (from a few to several thousand). However, in the face of the attack types guesspassword and warezmaster samples, its accuracy was only 61.33% and 17.87%. Noting that they all belong to a minority class of samples, we speculated the causes of this phenomenon: (1) The samples generated by the LCVAE are weak in describing the minority type. (2) The CNN-BiLSTM network cannot extract this type's features effectively. These will be dealt with in subsequent research.

4.5. Discussion and Additional Comparison

To verify the performance of the LCVAE-CBiLSTM. In this section, we compare the detection results on KDDTest+ with several state-of-the-art NID methods, as shown in Table 5.

Table 5. Comparison results (%) of different detection models on the KDDTest+ dataset.

Methods	Accuracy	Precision	F1-Score	Recall	FAR
SVM [50]	72.28	91.26	69.97	56.73	7.17
RF [51]	76.44	81.55	72.17	76.65	23.37
DNN [52]	80.22	95.85	79.70	68.21	3.90
CNN-BiLSTM [53]	83.58	85.82	85.14−	84.49+	17.55
SMOTE-NDD [36]	81.16	96.42−	80.76	67.41	2.90−
LCVAE-CNN [54]	85.51−	97.61+	80.78	68.90	1.34+
ROS-NDD [35]	78.16	92.32	77.93	67.41	7.39
LCVAE-CBiLSTM	**87.30+**	**96.08**	**87.89+**	**80.89−**	**4.36**

There were several NID methods selected for comparison, including machine learning methods (RF), single-structured neural networks (DNN), hybrid structured neural networks (CNN-BiLSTM), and multiple neural networks combined with data reinforcement (SMOTE-NDD, LCVAE-CNN, and SMOTE-NDD). As shown in Table 4, the LCVAE-CBiLSTM achieved the best results in the accuracy and F1-score as 87.30% and 87.89%, respectively. The precision value was 96.08%, and the recall value was 80.89%, which were slightly lower than the best ones, the LCVAE-CNN and CNN-BiLSTM, respectively. While achieving these excellent metrics, the LCVAE-CBiLSTM also maintained a high FAR of 4.36%, ranking third. Based on the values of the precision and FAR and combined with Table 5, we can conclude that LCVAE-CBiLSTM showed outstanding sensitivity to most network intrusions without overfitting. At the same time, it can be seen from the low recall rate that the method has a detection blind spot and cannot obtain accurate detection results for particular attacks.

The same drawbacks also appeared in other results. SMOTE-NDD, LCVAE-CNN, and ROS-NDD all exhibited a high accuracy (>90.00%), low recall (<70.00%), and low FAR (<5.00%). This proves that these methods' generalization is significantly flawed. The minority samples caused this phenomenon, where the classifier cannot obtain sufficient features to perform effective intrusion detection. In comparison, the recall of the proposed method was satisfactory, and it improved by more than 12%. This indicates that it has a greater advantage in dealing with a small number of classes of samples.

We can also find that both the LCVAE-CNN and CNN-BiLSTM are mediocre methods. However, the results obtained by the LCVAE-CBiLSTM after combining their points appeared to be greatly improved. The data generation model enabled the classifier to obtain better detection results. The classifier can also mine the deep features of the samples from both temporal and spatial perspectives. It optimizes the detection performance of the model and is more sensitive in dealing with minority types.

Overall, the LCVAE-CBiLSTM network intrusion detection model had a better classification performance on KDDTest+.

4.6. Calculation Time Comparison

Real-time performance is an important metric for IDS deployed in AHMSs. It not only directly affects the response time to an intrusion, but also indicates the NID algorithm's computing overhead. This is important for AHMSs with limited computing power.

As an experimental platform for computational performance, we designed an STM32F104 microcontroller-based miniature hydrological monitor, shown in Figure 10. It was powered by a 3.7 V–2 A lithium battery, which enabled recording the water temperature and surface humidity every five seconds. The LCVAE-CBiLSTM model's weight values trained by PyTorch were compiled into this platform by C. We randomly selected 1000 samples from KDDTest+ for testing the computational efficiency of SVM, the DNN, and the LCVAE-CBiLSTM on the low-power platform. The results are shown in Table 6.

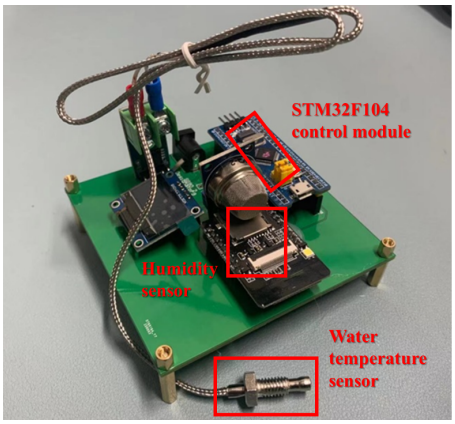


Figure 10. Hydrological sensor experiment platform.

Table 6. Detection time of different approaches.

Models	SVM	DNN	LCVAE-CBiLSTM
Detect time (s)	17.4	36.1	**26.8**

In Table 6, SVM and the DNN produced the shortest and longest detection times, respectively. Considering the intrusion detection accuracy in Table 5, they were unable to secure the AHMS. The time taken by the LCVAE-CBiLSTM to detect 1000 samples was only 9.4 s more than that of SVM, which is not worth mentioning compared to the improvement in accuracy. Therefore, it can be considered that the LCVAE-CBiLSTM has better computational performance and less computational effort.

5. Conclusions

This paper proposed a novel NID method, the LCVAE-CBiLSTM, for marine hydrographic monitoring networks, which has higher detection accuracy and less computational overhead. In the LCVAE-CBiLSTM, we first designed a CVAE based on the log-cosh loss function, which was used to reduce the dimensionality of the original NTD. More importantly, it generates new sample data with specific labels to supplement the number of minority samples and improve the detection capability of subsequent algorithms for minority class samples. Then, a hybrid feature extraction model that combines CNN and BiLSTM was proposed. This model combines the advantages of both neural networks and enables analyzing and extracting the deep features of the input data in space and time, providing for high-performance NID. Finally, high-precision attack detection was achieved using Softmax.

In order to illustrate LCVAE-CBiLSTM's performance, a series of experiments were conducted on the NSL-KDD dataset. We obtained satisfactory results regarding the detection accuracy, precision, F1-score, recall, and FAR of 87.30%, 96.08%, 87.89%, 80.89%, and 4.36%, respectively. At the same time, the LCVAE-CBiLSTM passed the computational efficiency test on a miniature hydrological monitor. Compared with the other methods, the LCVAE-CBiLSTM achieved more accurate detection results with less computational time.

The experimental results strongly demonstrated that the LCVAE-CBiLSTM has excellent NID performance. Predictably, if this method is widely deployed in AHMSs, it will certainly improve the security of data transmission and storage. However, the LCVAE-CBiLSTM also has the following drawbacks. The LCVAE-CBiLSTM shows a clear sensitivity bias for different attack types, which may mean that the method is not suitable for all network environments. On the other hand, the detection accuracy of the LCVAE-CBiLSTM is still poor for minority attack types, and it is powerless for 0-day attacks. Further research and analyses are needed concerning more significant feature relations based on the network protocol from the source to improve the detection efficiency. Our next research will focus on NTD prediction and early warning based on DL. Future research for this problem should return to the mechanism of intrusion generation. It is not enough to rely on deep learning alone to dig out the more useful features. Even if a data reinforcement algorithm is used, it is not effective at improving detection when the minority samples are few.

Author Contributions: T.H.: conceptualization, methodology, software, writing—original draft. H.X.: supervision. X.L.: data curation, software. X.S.: validation, formal analysis. Z.W.: writing—review and editing. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the National Key Research and Development Program of China (Grant Number 2021YFE0105500) and the National Natural Science Foundation of China (Grant Number 62171228).

Data Availability Statement: Publicly available datasets were analyzed in this study. This data can be found here: https://www.unb.ca/cic/datasets/nsl.html, accessed on 11 December 2022.

Acknowledgments: The authors would like to appreciate Nanjing University of Information Science and Technology, Hohai University, and the University of South Africa for supporting this research work.

Conflicts of Interest: The authors declare that they have no known competing financial interest or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Barale, V. A supporting marine information system for maritime spatial planning: the European Atlas of the Seas. *Ocean. Coast. Manag.* **2018**, *166*, 2–8. [CrossRef]
2. Heras, D.; Matovelle, C. Machine-learning methods for hydrological imputation data: analysis of the goodness of fit of the model in hydrographic systems of the Pacific-Ecuador. *Rev. Ambiente Água* **2021**, *16*, 3. [CrossRef]
3. Elsobeiey, M.E. Accuracy Assessment of Satellite-Based Correction Service and Virtual GNSS Reference Station for Hydrographic Surveying. *J. Mar. Sci. Eng.* **2020**, *8*, 542. [CrossRef]
4. Sławomir, G. Maritime Communications Network Development Using Virtualised Network Slicing of 5G Network. *NAŠE MORE Znanstveni Časopis Za More I Pomorstvo* **2020**, *67*, 78–86.
5. Sha, K.; Yang, T.A.; Wei, W.; Davari, S. A survey of edge computing based designs for IoT security. *Digit. Commun. Netw.* **2020**, *6*, 195–202. [CrossRef]
6. El-Diasty, M. Evaluation of KSACORS-based network GNSS-INS integrated system for Saudi coastal hydrographic surveys. *Geomat. Nat. Hazards Risk* **2020**, *11*, 1426–1446. [CrossRef]
7. Jouini, M.; Rabai, L.B.A. A security framework for secure cloud computing environments. In *Cloud Security: Concepts, Methodologies, Tools, and Applications*; IGI Global: Pennsylvania, PA, USA, 2019; pp. 249–263.
8. Fu, Y.; Lou, F.; Meng, F.; Tian, Z.; Zhang, H.; Jiang, F. An intelligent network attack detection method based on rnn. In Proceedings of the 2018 IEEE Third International Conference on Data Science in Cyberspace (DSC), Guangzhou, China, 18–21 June 2018; pp. 483–489.
9. Liu, H.; Lang, B. Machine learning and deep learning methods for intrusion detection systems: A survey. *Appl. Sci.* **2019**, *9*, 4396. [CrossRef]
10. Haider, W.; Hu, J.; Slay, J.; Turnbull, B.P.; Xie, Y. Generating realistic intrusion detection system dataset based on fuzzy qualitative modeling. *J. Netw. Comput. Appl.* **2017**, *87*, 185–192. [CrossRef]
11. Mahfouz, A.M.; Venugopal, D.; Shiva, S.G. Comparative analysis of ML classifiers for network intrusion detection. In *Proceedings of the Fourth International Congress on Information and Communication Technologies*; Springer: London, UK, 2020; pp. 193–207.
12. Abushark, Y.B.; Irshad Khan, A.; Alsolami, F.; Almalawi, A.; Mottahir Alam, M.; Agrawal, A.; Kumar, R.; Ahmad Khan, R. Cyber Security Analysis and Evaluation for Intrusion Detection Systems. *Comput. Mater. Contin* **2022**, *72*, 1765–1783. [CrossRef]
13. Hodo, E.; Bellekens, X.; Hamilton, A.; Dubouilh, P.L.; Iorkyase, E.; Tachtatzis, C.; Atkinson, R. Threat analysis of IoT networks using artificial neural network intrusion detection system. In Proceedings of the 2016 International Symposium on Networks, Computers and Communications (ISNCC), Yasmine Hammamet, Tunisia, 11–13 May 2016; pp. 1–6.
14. Chen, W.H.; Hsu, S.H.; Shen, H.P. Application of SVM and ANN for intrusion detection. *Comput. Oper. Res.* **2005**, *32*, 2617–2634. [CrossRef]
15. Wang, Y. A multinomial logistic regression modeling approach for anomaly intrusion detection. *Comput. Secur.* **2005**, *24*, 662–674. [CrossRef]
16. Agrawal, S.; Chowdhuri, A.; Sarkar, S.; Selvanambi, R.; Gadekallu, T.R. Temporal weighted averaging for asynchronous federated intrusion detection systems. *Comput. Intell. Neurosci.* **2021**, *2021*, 5844728. [CrossRef]
17. Ieracitano, C.; Adeel, A.; Gogate, M.; Dashtipour, K.; Morabito, F.C.; Larijani, H.; Raza, A.; Hussain, A. Statistical analysis driven optimized deep learning system for intrusion detection. In Proceedings of the International Conference on Brain Inspired Cognitive Systems, Xi'an, China, 7–8 July 2018; pp. 759–769.
18. Khraisat, A.; Gondal, I.; Vamplew, P.; Kamruzzaman, J. Survey of intrusion detection systems: Techniques, datasets and challenges. *Cybersecurity* **2019**, *2*, 1–22. [CrossRef]
19. Song, J.; Takakura, H.; Kwon, Y. A generalized feature extraction scheme to detect 0-day attacks via IDS alerts. In Proceedings of the 2008 International Symposium on Applications and the Internet, Turku, Finland, 28 July–1 August 2008; pp. 55–61.
20. Hou, T.; Xing, H.; Liang, X.; Su, X.; Wang, Z. Network intrusion detection based on DNA spatial information. *Comput. Netw.* **2022**, *217*, 109318. [CrossRef]
21. Kingma, D.P.; Welling, M. Auto-encoding variational bayes. *arXiv* **2013**, arXiv:1312.6114.
22. Kingma, D.P.; Mohamed, S.; Jimenez Rezende, D.; Welling, M. Semi-supervised learning with deep generative models. *Adv. Neural Inf. Process. Syst.* **2014**, *2*, 3581–3589.
23. Alavizadeh, H.; Alavizadeh, H.; Jang-Jaccard, J. Deep Q-Learning based Reinforcement Learning Approach for Network Intrusion Detection. *Computers* **2022**, *11*, 41. [CrossRef]
24. Alladi, T.; Kohli, V.; Chamola, V.; Yu, F.R. A deep learning based misbehavior classification scheme for intrusion detection in cooperative intelligent transportation systems. *Digit. Commun. Netw.* 2022, *in press*. [CrossRef]
25. Fuqun, Z. Detection method of LSSVM network intrusion based on hybrid kernel function. *Mod. Electron. Tech.* **2015**, *21*, 027.

26. Tao, P.; Sun, Z.; Sun, Z. An improved intrusion detection algorithm based on GA and SVM. *IEEE Access* **2018**, *6*, 13624–13631. [CrossRef]
27. Peng, K.; Leung, V.C.; Huang, Q. Clustering approach based on mini batch kmeans for intrusion detection system over big data. *IEEE Access* **2018**, *6*, 11897–11906. [CrossRef]
28. Dong, B.; Wang, X. Comparison deep learning method to traditional methods using for network intrusion detection. In Proceedings of the 2016 8th IEEE International Conference on Communication Software and Networks (ICCSN), Beijing, China, 4–6 June 2016; pp. 581–585.
29. Ingre, B.; Yadav, A. Performance analysis of NSL-KDD dataset using ANN. In Proceedings of the 2015 International Conference on Signal Processing and Communication Engineering Systems, Guntur, India, 2–3 January 2015; pp. 92–96.
30. Yan, B.; Han, G. Effective feature extraction via stacked sparse autoencoder to improve intrusion detection system. *IEEE Access* **2018**, *6*, 41238–41248. [CrossRef]
31. Mendonca, R.V.; Silva, J.C.; Rosa, R.L.; Saadi, M.; Rodriguez, D.Z.; Farouk, A. A lightweight intelligent intrusion detection system for industrial internet of things using deep learning algorithms. *Expert Syst.* **2022**, *39*, e12917. [CrossRef]
32. Tang, T.A.; Mhamdi, L.; McLernon, D.; Zaidi, S.A.R.; Ghogho, M. Deep learning approach for network intrusion detection in software defined networking. In Proceedings of the 2016 International Conference on Wireless Networks and Mobile Communications (WINCOM), Fez, Morocco, 26–29 October 2016; pp. 258–263.
33. Li, G.; Shen, Y.; Zhao, P.; Lu, X.; Liu, J.; Liu, Y.; Hoi, S.C. Detecting cyberattacks in industrial control systems using online learning algorithms. *Neurocomputing* **2019**, *364*, 338–348. [CrossRef]
34. Al-Sawwa, J.; Ludwig, S.A. Performance evaluation of a cost-sensitive differential evolution classifier using spark–Imbalanced binary classification. *J. Comput. Sci.* **2020**, *40*, 101065. [CrossRef]
35. Lemaître, G.; Nogueira, F.; Aridas, C.K. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *J. Mach. Learn. Res.* **2017**, *18*, 559–563.
36. Gonzalez-Cuautle, D.; Hernandez-Suarez, A.; Sanchez-Perez, G.; Toscano-Medina, L.K.; Portillo-Portillo, J.; Olivares-Mercado, J.; Perez-Meana, H.M.; Sandoval-Orozco, A.L. Synthetic minority oversampling technique for optimizing classification tasks in botnet and intrusion-detection-system datasets. *Appl. Sci.* **2020**, *10*, 794. [CrossRef]
37. Vu, L.; Nguyen, Q.U.; Nguyen, D.N.; Hoang, D.T.; Dutkiewicz, E. Learning latent distribution for distinguishing network traffic in intrusion detection system. In Proceedings of the ICC 2019—2019 IEEE International Conference on Communications (ICC), Shanghai, China, 20–24 May 2019; pp. 1–6.
38. Chen, J.; Wu, D.; Zhao, Y.; Sharma, N.; Yu, S. Fooling intrusion detection systems using adversarially autoencoder. *Digit. Commun. Netw.* **2020**, *7*, 453–460. [CrossRef]
39. Lopez-Martin, M.; Carro, B.; Sanchez-Esguevillas, A.; Lloret, J. Conditional variational autoencoder for prediction and feature recovery applied to intrusion detection in iot. *Sensors* **2017**, *17*, 1967. [CrossRef]
40. Hannan, A.; Gruhl, C.; Sick, B. Anomaly based Resilient Network Intrusion Detection using Inferential Autoencoders. In Proceedings of the 2021 IEEE International Conference on Cyber Security and Resilience (CSR), Rhodes, Greece, 26–28 July 2021; pp. 1–7.
41. Liu, C.; Antypenko, R.; Sushko, I.; Zakharchenko, O. Intrusion Detection System After Data Augmentation Schemes Based on the VAE and CVAE. *IEEE Trans. Reliab.* **2022**, *71*, 1000–1010. [CrossRef]
42. Zhang, Y.; Liu, Q. On IoT intrusion detection based on data augmentation for enhancing learning on unbalanced samples. *Future Gener. Comput. Syst.* **2022**, *133*, 213–227. [CrossRef]
43. Yang, Y.; Zheng, K.; Wu, C.; Yang, Y. Improving the classification effectiveness of intrusion detection by using improved conditional variational autoencoder and deep neural network. *Sensors* **2019**, *19*, 2528. [CrossRef]
44. Khanam, S.; Ahmedy, I.; Idris, M.Y.I.; Jaward, M.H. Towards an Effective Intrusion Detection Model Using Focal Loss Variational Autoencoder for Internet of Things (IoT). *Sensors* **2022**, *22*, 5822. [CrossRef]
45. Vu, L.; Nguyen, Q.U.; Nguyen, D.N.; Hoang, D.T.; Dutkiewicz, E. Deep Generative Learning Models for Cloud Intrusion Detection Systems. *IEEE Trans. Cybern.* **2022**, *53*, 565–577. [CrossRef]
46. Muna, A.H.; Moustafa, N.; Sitnikova, E. Identification of malicious activities in industrial internet of things based on deep learning models. *J. Inf. Secur. Appl.* **2018**, *41*, 1–11.
47. Zhang, G.; Wang, X.; Li, R.; Song, Y.; He, J.; Lai, J. Network intrusion detection based on conditional Wasserstein generative adversarial network and cost-sensitive stacked autoencoder. *IEEE Access* **2020**, *8*, 190431–190447. [CrossRef]
48. Chen, P.; Chen, G.; Zhang, S. Log hyperbolic Cosine Loss Improves Variational Auto-Encoder. 2018. Available online: https://openreview.net/forum?id=rkglvsC9Ym (accessed on 14 January 2023).
49. Alrawashdeh, K.; Purdy, C. Toward an online anomaly intrusion detection system based on deep learning. In Proceedings of the 2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA), Anaheim, CA, USA, 18–20 December 2016; pp. 195–200.
50. Tian, Y.; Mirzabagheri, M.; Bamakan, S.M.H.; Wang, H.; Qu, Q. Ramp loss one-class support vector machine; A robust and effective approach to anomaly detection problems. *Neurocomputing* **2018**, *310*, 223–235. [CrossRef]
51. 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; pp. 1322–1328.

52. Tang, C.; Luktarhan, N.; Zhao, Y. SAAE-DNN: Deep learning method on intrusion detection. *Symmetry* **2020**, *12*, 1695. [CrossRef]
53. Huda, S.; Miah, S.; Yearwood, J.; Alyahya, S.; Al-Dossari, H.; Doss, R. A malicious threat detection model for cloud assisted internet of things (CoT) based industrial control system (ICS) networks using deep belief network. *J. Parallel Distrib. Comput.* **2018**, *120*, 23–31. [CrossRef]
54. Xu, X.; Li, J.; Yang, Y.; Shen, F. Toward effective intrusion detection using log-cosh conditional variational autoencoder. *IEEE Internet Things J.* **2020**, *8*, 6187–6196. [CrossRef]