Research Article

An Efficient Communication Intrusion Detection Scheme in AMI Combining Feature Dimensionality Reduction and Improved LSTM

Guanyu Lu and Xiuxia Tian

College of Computer Technology and Science, Shanghai University of Electric Power, Shanghai 200090, China

Correspondence should be addressed to Xiuxia Tian; xxtian@shiep.edu.cn

Received 17 December 2020; Revised 13 February 2021; Accepted 22 March 2021; Published 21 April 2021

Academic Editor: Jinguang Han

Copyright © 2021 Guanyu Lu and Xiuxia Tian. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Communication intrusion detection in Advanced Metering Infrastructure (AMI) is an eminent security technology to ensure the stable operation of the Smart Grid. However, methods based on traditional machine learning are not appropriate for learning high-dimensional features and dealing with the data imbalance of communication traffic in AMI. To solve the above problems, we propose an intrusion detection scheme by combining feature dimensionality reduction and improved Long Short-Term Memory (LSTM). The Stacked Autoencoder (SAE) has shown excellent performance in feature dimensionality reduction. We compress high-dimensional feature input into low-dimensional feature output through SAE, narrowing the complexity of the model. Methods based on LSTM have a superior ability to detect abnormal traffic but cannot extract bidirectional structural features. We designed a Bi-directional Long Short-Term Memory (BiLSTM) model that added an Attention Mechanism. It can determine the criticality of the dimensionality and improve the accuracy of the classification model. Finally, we conduct experiments on the UNSW-NB15 dataset and the NSL-KDD dataset. The proposed scheme has obvious advantages in performance metrics such as accuracy and False Alarm Rate (FAR). The experimental results demonstrate that it can effectively identify the intrusion attack of communication in AMI.

1. Introduction

In recent years, as the Internet of Things (IoT) technology is commonly used in the power industry, Smart Grid has become the development direction of future power grids. The core architecture of Smart Grid connecting with the computer network is AMI. AMI is a complicated system directly related to electricity consumption information, privacy information, and electricity transaction information. The possible threat of network intrusion has a huge impact on the reliable operation of the Smart Grid [1, 2]. As an influential research content of network communication security, intrusion detection has been widely discussed by experts and scholars. The application of intrusion detection algorithms represents one of the research hotspots in the field of communication in AMI in recent years. Radoglou-Grammatikis and Sarigiannidis [3] summarized the

contribution of intrusion detection and prevention system (IDPS) to the Smart Grid paradigm and provided an analysis of 37 cases. Intrusion detection can be viewed as a classification problem, using machine learning algorithms and data mining algorithms to classify network data into normal traffic and intrusion attack traffic [4]. When the intrusion detector finds misbehavior, it can take appropriate actions immediately so that any harm to the system will be minimized [5]. At present, related research can be divided into misuse-based detection [6] and anomaly-based detection [7] according to detection technology. The misuse-based intrusion detection scheme matches the extracted network traffic with the data traffic, which has the existing type tags. If the detected traffic and intrusion attack traffic have similar characteristics, the system will send out an alarm message. Such a method has good performance in identifying existing attacks by establishing a pattern library of intrusion attacks.

However, as an emerging model in the Smart Grid, there will be many new types of attacks appearing in AMI. The accuracy of the Misuse-based intrusion detection has decreased significantly, so it cannot meet the existing needs of the communication environment in AMI.

By judging the degree of deviation among the features of the collected traffic and the normal traffic, the anomaly-based intrusion detection scheme identifies intrusion attacks. It can be divided into intrusion detection based on statistical learning [8], traditional machine learning [9], and deep learning [10]. Because of the requirements of data distribution, intrusion detection methods based on statistical learning have been eliminated. With the development of Artificial Intelligence (AI) technology, the accuracy of methods based on machine learning has been significantly improved. However, communication traffic in AMI presents the characteristics of large data volume, high-dimensional data, and complex feature information. Methods based on traditional machine learning have the limitation of manually setting features in feature selection. Such methods can only be applied to simple and shallow learning. By constructing a deep hierarchical network structure, the methods based on deep learning can learn advanced features from data automatically. This method saves time for feature engineering [11]. The experimental result shows that Autoencoder (AE) has satisfactory performance in feature dimensionality reduction, and LSTM has an exceptional ability to solve classification problems. Currently, researchers have proposed a variety of intrusion detection schemes based on these two models. Dong et al. [12] proposed an intrusion detection model named AE-AlexNet based on deep learning. They use AE to realize dimensionality reduction of high-dimensional traffic. This method fails to achieve layer-by-layer training for high-dimensional traffic, and the robustness of the model is relatively poor. Due to the availability of LSTM on time series data, Althubiti et al. [13] proposed an intrusion detection scheme based on LSTM. In this scheme, only the unidirectional structural features are extracted, and it cannot determine important features. It has serious limitations in its application.

To deal with the above problems, we propose a communication intrusion detection scheme by combining feature dimensionality reduction and improved LSTM in AMI. The main contributions of this paper can be summarized as follows:

- (1) First, we propose a Stacked Autoencoder method to achieve feature dimensionality reduction for the high-dimensional features of data in AMI. By extracting the key point information of attributes, it can reduce the calculation time and improve the efficiency of communication intrusion detection in AMI. SAE can modularize and improve the robustness of the neural network.
- (2) Second, for the problem that LSTM cannot extract bidirectional structural features, we propose a Bidirectional Long Short-Term Memory model for the classification of traffic. It can reduce the high FAR due to data imbalance in AMI.
- (3) Third, to determine the criticality of the dimensionality and the feature, we improve the classification model by the Attention Mechanism. It sets weight coefficients to allocate more attention to key dimensions and important features, to realize accurate detection.
- (4) Fourth, the proposed method is compared with the methods based on traditional machine learning and the recent papers on intrusion detection. The experimental results indicate that our scheme is preferable to other competing schemes.

The rest of this paper is organized as follows: Section 2 summarizes the related research work. Section 3 presents the basic theory required in the scheme. Section 4 introduces our intrusion detection scheme in detail. Section 5 is the analysis and comparison of the experiment. Section 6 reports possible threats to the validity of the scheme. The paper is concluded in Section 7.

2. Related Work

This section discusses two types of related work: methods based on traditional machine learning and methods based on deep learning. According to the requirements of communication intrusion detection in AMI, both methods face certain challenges. The challenge of methods based on traditional machine learning is whether the selected features are appropriate for the classification model. Methods based on deep learning face the challenges of high-dimensional features of communication traffic and data imbalance in AMI.

2.1. Intrusion Detection Based on Traditional Machine Learning. Most of the previous research studies are based on traditional machine learning methods, such as Naive Bayes, Decision Tree, K-Nearest Neighbor (KNN), Support Vector Machine (SVM), and Hidden Markov Model (HMM). Farid et al. [14] proposed a learning algorithm of intrusion detection, which uses the methods of Naive Bayes and Decision Tree to reconstruct the data. The purpose was to reduce the noise and eliminate the redundant attributes in the training data. This scheme improved the accuracy of different types of network intrusion attacks. Radoglou-Grammatikis and Sarigiannidis [15] proposed an intrusion detection system in AMI based on the CART decision tree, and the system was tested on the intrusion detection dataset CICIDS2017. Accuracy can reach 99.66%, and True Positive Rate (TPR) can reach 99.30%. Senthilnayagi et al. [16] used the Rough Sets Attribute Reduction algorithm to select features and data, extracted the more important features and data, and realized the feature dimensionality reduction of the data attributes in the dataset. Finally, the improved KNN classifier was utilized to complete the classification of traffic, which effectively

reduces the FAR of detecting intrusion attacks. For the threats in AMI, Vijayanand et al. [17] constructed an intrusion detection system in Smart Grid through the SVM classifier method. The feature was selected by mutual information value. Finally, through simulation experiments, the detection accuracy of normal records obtained could reach 93.4%, and the detection accuracy of intrusion attack records could reach 89.2%. The advantage of the SVM model is its brilliant generalization ability. The FAR of the final results is low, but the SVM model is only suitable for solving the binary classification problem. For detecting multiclass intrusion attacks in AMI, the performance of the SVM model is not good. Hurley et al. [18] used HMM algorithm to develop an adaptive network intrusion detection system, which had a superior performance in detecting intrusion attacks in Software Defined Network (SDN). High-quality training datasets can be constructed through the clustering algorithm in traditional machine learning. This kind of method can make the type of the dataset from complex to simple. It can reduce the spatial dimension of the data and the computational overhead. In previous studies, clustering algorithms used in intrusion detection include k -means algorithm [19], hierarchical clustering [20], and Principal Component Analysis (PCA) [21]. Unsupervised machine learning technology is an imperative method for data processing and feature engineering in the context of massive data in AMI. However, such algorithms are sensitive to the outliers and noise of data.

The arrival of the big data era indicates that intrusion detection has entered a stage of large data volume, high data dimension, high network bandwidth, and complex feature information. Methods based on traditional machine learning need to manually set features, which are relatively shallow learning methods. It is therefore difficult to achieve the purpose of prediction and analysis.

2.2. Intrusion Detection Based on Deep Learning. The deep learning method constructs a network structure constituted by multiple hidden layers to adapt to the higher-dimensional learning process by learning the internal laws and representation levels of sample data. At the stage of feature engineering, the convergence time of the model is saved. The deep learning algorithms commonly used in the field of communication intrusion detection in AMI include Autoencoder, Recurrent Neural Network (RNN) and its excellent variants, and Convolutional Neural Network (CNN). To obtain hidden information, Sun et al. [22] adopted the idea of the Variational Autoencoder (VAE) to achieve feature dimensionality reduction. They can extract more advanced features than manually set features. The proposed scheme has shown good performance on the KDD-CUP dataset, Mnist dataset, and UCSD pedestrian's dataset. Distributed Denial of Services (DDoS) is one of the most notorious attacks in AMI. Learning features through a multilayer AutoEncoder, Ali and Li [23] proposed an efficient DDoS attack detection technique. Bhardwaj et al. [24] combined stacked sparse AutoEncoder and Deep Neural Network (DNN) to detect DDoS attacks in cloud computing.

However, there are various types of intrusion attacks in AMI, which cannot guarantee the robustness of the scheme. Gao et al. [25] proposed a new intrusion detection method based on the LSTM model, which can be used in Supervisory Control And Data Acquisition (SCADA) systems. Agarap [26] introduced a linear SVM to replace the *softmax* function in the final output layer of the Gated Recurrent Unit (GRU) and built a model named GRU-SVM for intrusion detection. Finally, through simulation experiments, they showed the superiority of their scheme in training and testing time. Roy and Cheung [27] proposed to detect attacks based on the BiLSTM model in IoT. They used the UNSW-NB15 dataset for testing and achieved an accuracy of over 95% in attack detection. Khan et al. [28] built the intrusion detection system consisting of two stages: The first stage is the anomaly detection module based on Spark-ML, and the second stage is the misuse detection module based on Convolutional-LSTM (Conv-LSTM). In the cross-validation, the accuracy rate can reach 97.29%. Riyaz and Ganapathy [29] used CNN to select the most contributory feature and classify the traffic when identifying and detecting intrusion attacks on wireless networks. The proposed intrusion detection system achieved an overall accuracy of 98.88%. Lin et al. [30] proposed a framework named IDSGAN by Generative Adversarial Networks (GAN) to generate adversarial attacks to deceive and evade intrusion detection systems. Based on the NSL-KDD dataset, experiments had proved the feasibility of this model to attack systems that can detect multiple different attacks, and had achieved excellent results.

Although intrusion detection based on deep learning has many advantages in feature learning, it also has some shortcomings. During the communication of AMI, the training dataset contains a large number of normal data samples, and the proportion of intrusion attack traffic is very small. The records of data show the phenomenon of unbalance. Meanwhile, the structural data are high-dimensional, and the features need to be selected and extracted. In response to the need for communication intrusion detection in AMI, this paper proposes a corresponding scheme to solve the above problems.

3. Preliminary: Basic Theory

This section introduces the basic theory of the model used in the next section.

3.1. Autoencoder. Autoencoder is an unsupervised neural network algorithm, which can reconstruct the vector which is input into the model [31]. It shows powerful nonlinear generalization capabilities and has been applied in many fields, such as image denoising [32] and anomaly detection [33]. Figure 1 is the hierarchical structure of the Autoencoder. The Autoencoder consists of two parts: encoder (visible layer to hidden layer) and decoder (hidden layer to output layer). The encoder is represented by the function $h = f_{\theta}(x)$, and it maps the input data to the feature space. The decoder is represented by the function $x' = g_{\theta'}(h)$, and it maps the encoded data back to the sample space for the generation and reconstruction of the input vector.

FIGURE 1: The hierarchical structure of the Autoencoder. The structure consists of a visible layer, a hidden layer, and an output layer. The visible layer and the hidden layer constitute the encoder part. The hidden layer and the output layer constitute the decoder part.

The learning target of an Autoencoder is to make the input equal to the output. The objective function of the network is $g_{\theta'}(f_{\theta}(x)) \approx x$, which means learning an identity. We suppose that the set of data samples input to the Autoencoder model is $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$. The training dataset formed after removing the category labels is $\{x_1, x_2, \dots, x_n\}$.

The encoder can be expressed by the following equation:

$$h_k = f_{\theta}(x_k) = R\left(\sum_{i=1}^{n_1} W_{ij} \cdot x_{ki} + b_i\right). \quad (1)$$

The decoder can be expressed by the following equation:

$$x'_k = g_{\theta'}(h_k) = R\left(\sum_{i=1}^{n_2} W_{ij} \cdot h_{ki} + b_i\right). \quad (2)$$

The loss function can be expressed by the following equation:

$$L_{W,b}(x_k, x'_k) = \frac{1}{2} \sum_{i=1}^{n_1} (x_k - x'_k)^2. \quad (3)$$

In equations (1)–(3), x_k is the k th training sample. h_k is the value of each neuron in the hidden layer of the k th training sample. x'_k is the value of each neuron in the output layer of the k th training sample. n_1 is the number of neurons in the visible layer, and n_2 is the number of neurons in the hidden layer. W_{ij} is the connection weight of the i th neuron in the previous layer and the j th neuron in the next layer. b_i is the bias term of the i th neuron on the corresponding layer. R indicates that the activation function used in the Autoencoder is the *Relu* function.

For the task of feature dimensionality reduction, the final output x'_k and the original data x_k have the same feature dimensions, which is meaningless. The result of h_k in the hidden layer is the dimensionality reduction expression of x_k . It is obtained without losing the original data information as much

as possible. We can accomplish the target of transforming high-dimensional features into low-dimensional features.

3.2. Long Short-Term Memory. Long Short-Term Memory neural network was proposed as an improved variant of Recurrent Neural Network, which mainly solves the problems of gradient disappearance or explosion that may occur during RNN training. It is more suitable for use in sequence data processing with long-term correlation [34]. Figure 2 displays the LSTM network structure.

Figure 3 is the internal structure of the memory storage unit of LSTM, which is mainly composed of the forget gate, the input gate, and the output gate [35]. The forget gate is responsible for processing the output of the previous layer, selecting useful information, and filtering useless information. The input gate is responsible for judging the importance of information and updating the status of the unit through critical information. The output gate is responsible for determining which unit status can be input to the unit of the next layer.

The forget gate can be expressed by the following equation:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f). \quad (4)$$

In the equation, the value of h_{t-1} and x_t is 0 or 1. After the forget gate, if the output is 0, the current useful information will be stored. And if the output is 1, the current useless information will be deleted.

The calculation process of the input gate consists of two parts: one is to determine the important information that needs to be added to the unit status through the *sigmoid* activation function, and the other is to use the *tanh* activation function to form a new vector to update the unit status. The equations of the two parts are shown in the following equations:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i), \quad (5)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C). \quad (6)$$

At this time, the original unit status C_{t-1} is updated to C_t , and the equation is as follows:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t. \quad (7)$$

The output gate determines the output through the *sigmoid* function, which can be expressed by the following equations:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \quad (8)$$

$$h_t = o_t * \tanh(C_t). \quad (9)$$

In equations (4)–(9): σ indicates that the activation function used is *sigmoid*. h_{t-1} is the hidden layer status of the previous layer unit. W_f is the weight of the forget gate, and b_f is the bias term of the forget gate. W_i is the weight of the input gate, and b_i is the bias term of the input gate. W_C is the weight of the unit status, and b_C is the bias term of the unit

FIGURE 2: The framework of the LSTM model. *Sigmoid* and *tanh* activation functions are used inside the unit.

FIGURE 3: The internal structure of the LSTM unit. (a) Forget gate. (b) Input gate. (c) Output gate.

status. W_o is the weight of the output gate, and b_o is the bias term of the output gate.

4. Proposed Method

The intrusion detection scheme we proposed is mainly for the communication scenario in AMI, and Figure 4 provides a detailed description of this scenario. AMI is generally composed of smart meters, concentrators, grid servers of measurement management, and its communication network. The bottom component of AMI is the smart meter. It is responsible for collecting and analyzing user information on Home Area Network (HAN), Business Area Network (BAN), and Industry Area Network (IAN), while monitoring and recording electricity consumption data and other statistical data. The intermediate component of the system is the data collector deployed in the Neighbor Area Network (NAN), responsible for summarizing the data information received from the smart meter. The top component of the system is the device of AMI headend, which is deployed in the Wide Area Network (WAN) and is responsible for collecting data from multiple data collectors. There are multiple feasible communication methods and protocols at each level. For example, the ZigBee protocol stack and Bluetooth communication are used in the HAN. In the NAN, the communication standard of WiFi is used. There are numerous communication methods in the WAN, such as Digital Subscriber Line (DSL), optical fiber communications,

and GPRS communication. Figure 4 also captures the collection environment of communication data in AMI. The Programmable Logic Controller (PLC) and the data acquisition unit are connected to the communication server on the data bus through the switch. On the data bus, potential intrusion attack threat terminals send abnormal traffic during the communication, carry out various attacks, and affect the communication between normal devices. The database server is responsible for storing communication log files between devices. The data acquisition unit is in charge of collecting normal traffic and intrusion attack traffic.

In this section, we first introduce the data preprocessing approach. Then, we describe the proposed communication intrusion detection model in AMI and explain how to classify the records of normal traffic and intrusion attack traffic.

4.1. Data Preprocessing. The communication traffic in the AMI system contains many nondigital features. Such features cannot be directly used as input to the model, so it is necessary to convert nondigital features into digital features. We apply the idea of one-hot encoding to process data and use n -bit status registers to encode n states of nondigital features. Assume that nondigital features have n states such as $\{\text{Status}_1, \text{Status}_2, \dots, \text{Status}_n\}$, and Table 1 is the final result of the encoding.

To eliminate the difference of the feature quantification results and prevent the features with a large value range from affecting the model results, we use the procedure of normalization for all features to process the obtained data. Normalization of data can improve the accuracy of the model and speed up the solution of the model. In our proposed scheme, the Min-Max normalization method is used [36]. Assuming that the dataset of a group of features is $\{X_1, X_2, \dots, X_n\}$, the equation for normalizing a certain data X_i in the set is shown in the following equation:

$$X'_i = \frac{X_i - X_{\min}}{X_{\max} - X_{\min}}. \quad (10)$$

Here, X'_i is the normalized data and $X_{\min}$ and $X_{\max}$ are the minimum and maximum values in the dataset, respectively.

The pseudocode description of the algorithm at this stage is shown in Algorithm 1.

FIGURE 4: Communication scenario in AMI. (i) Communication network hierarchy structure in AMI. (ii) The collection environment of communication data in AMI. (iii) Protocols used in each layer of AMI.

TABLE 1: Encoding results of n states of nondigital features.

Description of the status	One-hot encoding result (n bits)
Status_1	(0, 0, ..., 0, 1)
Status_2	(0, 0, ..., 1, 0)
...	...
Status_ $n-1$	(0, 1, ..., 0, 0)
Status_ n	(1, 0, ..., 0, 0)

4.2. Intrusion Detection Model. The preprocessed data can be used for classification detection. Figure 5 is the established communication intrusion detection model in AMI, which is divided into two parts: the first part is to perform dimensionality reduction operations on the features of the data through the Stacked Autoencoder, which is marked by (i) in Figure 5. The second part is to classify the traffic through the improved LSTM for the data after feature reduction, to achieve the purpose of identifying intrusion attacks. This part is indicated by (ii) in Figure 5.

4.2.1. SAE for Feature Dimensionality Reduction. The Stacked Autoencoder is a neural network made up of multiple layers of sparse Autoencoders, which can effectively extract features and reduce feature dimensions [37]. Figure 6 is the structure of the Stacked Autoencoder. This SAE is composed of n Autoencoders stacked. Hidden layers (1~ $n-1$) are the output of the previous Autoencoder and the input of the next Autoencoder. In this case, one Autoencoder is nested inside another, and learning takes place in a layer-by-layer greedy learning manner [38]. After every Autoencoder is trained, the decoder part is removed, and the final target output is attached to the innermost encoder layer.

To use the AMI dataset for communication traffic, after completing the data preprocessing, the vector with a dimension of 196 is obtained. This vector is used as the input of SAE. Figure 7 is the design of SAE to achieve feature selection and dimensionality reduction. In the proposed scheme, the number of hidden layers of the SAE network

FIGURE 5: Communication intrusion detection model in AMI. The overall model has two stages: (i) SAE for feature dimensionality reduction. (ii) Improved LSTM for classification.

structure is 4 layers. The number of neurons in the hidden layer is {128, 64, 32, 32}, and finally, a 32-dimensional vector is selected as the output.

The processes of SAE for feature dimensionality reduction generally include two stages: pretraining and fine-

FIGURE 6: The structure of SAE. The output of the Autoencoder of the previous layer is used as the input of the Autoencoder of the next layer.

FIGURE 7: The process of feature selection and dimensionality reduction implemented by SAE in the proposed scheme. The structure of the hidden layer in SAE is {128,64,32,32}.

tuning. Pretraining is an unsupervised training process that uses a large amount of unlabeled traffic data to perform layer-by-layer greedy learning and training in SAE. The steps of the pretraining stage are as follows:

Step 1: input the preprocessed AMI communication data into the visible layer of the SAE, and initialize the connection weight W and bias term b randomly.

Step 2: train the network parameters of the Hidden layer₁ and calculate the output of the Hidden layer₁ through the trained parameters.

Step 3: use the unsupervised learning method to train the Autoencoder and calculate the *loss* function value $L_{W,b}$. Keep updating the weight W and the bias term b until the *loss* function value finally reaches the set threshold and no longer changes.

Step 4: use the output of the previous network layer as the input of the next network layer and apply the same

method to train the parameters of this hidden network layer. Repeat Step 3 until all layers of the SAE have been trained.

The above pretraining process cannot obtain a mapping from the input communication traffic in AMI to the output label, so one or more connection layers need to be added to the last layer of the SAE network, and the backpropagation method is used for training. This stage is called the fine-tuning process. The steps of the fine-tuning stage are as follows:

Step 1: construct the entire SAE by connecting the hidden layers trained by each AE, and set the connection weight W and the bias term b to the values obtained in the pretraining stage.

Step 2: cascade a *softmax* classifier after the last layer, and train the network parameters of the *softmax* classifier in combination with the labeled original data.

Step 3: use the network parameters of the pretraining stage and the fine-tuning stage as the initialization parameters of the entire deep network. Find the parameter values around the minimum value of the cost function as the optimal parameter.

Step 4: use the backpropagation algorithm to fine-tune the optimal parameters obtained in the SAE model.

The pseudocode of the algorithm using the SAE to complete feature dimensionality reduction is shown in Algorithm 2.

The scheme has been put forward since the process of completing feature dimensionality reduction. As the network deepens, the training process becomes more difficult and the convergence speed becomes slower. We adopted the idea of Batch Normalization (BN) [39] to solve this problem. In Step 4 of the fine-tuning stage, there is the phenomenon that gradient disappears in the low-layer neural network during backpropagation. BN means that the input value distribution of any neuron in each layer of the neural network is forced back to the standard normal distribution with the mean of 0 and the variance of 1 through the normalization method. In this way, the input value of the nonlinear transformation function falls into an area that is sensitive to the input, to avoid the problem of vanishing gradient. Figure 8 displays the process of improving the hidden layer network

FIGURE 8: Using the BN method to improve the hidden layer network structure of the Autoencoder. (a) The original network structure between two layers. (b) The improved network structure between two layers.

structure of the Autoencoder. We use the idea of BN for processing behind each hidden layer.

In the BN operating experience, the equation for transforming the activation value of each neuron in the hidden layer is shown in the following equation:

$$\hat{x}^{(k)} = \frac{x^{(k)} - E[x^{(k)}]}{\sqrt{\text{Var}[x^{(k)}]}}. \quad (11)$$

In the equation, $x^{(k)}$ is the linear activation value of the corresponding neuron in this layer. $E[x^{(k)}]$ is the average value of linear activation values obtained by all training instances in this training process. $\text{Var}[x^{(k)}]$ is the variance of the linear activation value. Assuming that there are n instances in the training process, the calculation equations of $E[x^{(k)}]$ and $\text{Var}[x^{(k)}]$ are

$$E[x^{(k)}] = \frac{1}{n} \sum_{i=1}^n x_i^{(k)}, \quad (12)$$

$$\text{Var}[x^{(k)}] = \frac{1}{n} \sum_{i=1}^n (x_i^{(k)} - E[x^{(k)}])^2. \quad (13)$$

To prevent the expression ability of the SAE network from decreasing after changing the distribution, the adjustment parameters *scale* and *shift* are added at each neuron to activate the inverse transformation operation. The equation for the inverse operation is shown in the following equation:

$$y^{(k)} = \gamma^{(k)} \hat{x}^{(k)} + \beta(k). \quad (14)$$

4.2.2. Improved LSTM for Classification. After completing data dimensionality reduction, it is necessary to classify the normal traffic and intrusion attack traffic of communication data in AMI. The scheme in this paper applies the Bi-directional Long Short-Term Memory [40] method. Figure 9 is the structure of the BiLSTM model.

In the proposed scheme, the input layer is responsible for sequence encoding of the data after feature

FIGURE 9: The structure of the BiLSTM network. It consists of a forward working LSTM and a backward working LSTM.

dimensionality reduction. The forward working LSTM unit is responsible for extracting the forward features of the data sequence in the input layer, and the backward working LSTM unit is responsible for extracting the backward features of the data sequence in the input layer. The output layer integrates the data output by the forward and backward transmission layers.

The calculation equations inside the LSTM unit of forward transmission are shown in the following equations:

$$\vec{f}_t = \sigma(\vec{W}_f \cdot [\vec{h}_{t-1}, \vec{x}_t] + \vec{b}_f), \quad (15)$$

$$\vec{i}_t = \sigma(\vec{W}_i \cdot [\vec{h}_{t-1}, \vec{x}_t] + \vec{b}_i), \quad (16)$$

$$\vec{C}_t = \vec{f}_t * \vec{C}_{t-1} + \vec{i}_t * \tanh(\vec{W}_C \cdot [\vec{h}_{t-1}, \vec{x}_t] + \vec{b}_C), \quad (17)$$

$$\vec{o}_t = \sigma(\vec{W}_o \cdot [\vec{h}_{t-1}, \vec{x}_t] + \vec{b}_o), \quad (18)$$

$$\vec{h}_t = \vec{o}_t * \tanh(\vec{C}_t). \quad (19)$$

The calculation equations inside the LSTM unit of backward

transmission are shown in the following equations:

$$\bar{f}_t = \sigma\left(\bar{W}_f \cdot \left[\bar{h}_{t-1}, \bar{x}_t\right] + \bar{b}_f\right), \quad (20)$$

$$\bar{i}_t = \sigma\left(\bar{W}_i \cdot \left[\bar{h}_{t-1}, \bar{x}_t\right] + \bar{b}_i\right), \quad (21)$$

$$\bar{C}_t = \bar{f}_t * \bar{C}_{t-1} + \bar{i}_t * \tanh\left(\bar{W}_C \cdot \left[\bar{h}_{t-1}, \bar{x}_t\right] + \bar{b}_C\right), \quad (22)$$

$$\bar{o}_t = \sigma\left(\bar{W}_o \cdot \left[\bar{h}_{t-1}, \bar{x}_t\right] + \bar{b}_o\right), \quad (23)$$

$$\bar{h}_t = \bar{o}_t * \tanh\left(\bar{C}_t\right). \quad (24)$$

The output vector h_t of the output layer can be calculated from the output vectors $\bar{h}_t$ and $\bar{h}_t$ of the forward and backward hidden layers, respectively. The calculation equation is shown in the following equation:

$$h_t = \left[\bar{h}_t \oplus \bar{h}_t\right]. \quad (25)$$

In equation (25), $\oplus$ is the combination method of forward and backward output vectors.

In the communication intrusion detection scheme in AMI, to pay corresponding attention to the different features of the intrusion attack traffic data, the Attention Mechanism is introduced. This will improve the accuracy of intrusion detection. Attention Mechanism is widely used in image processing [41], natural language processing [42], target detection [43], and other fields. The core idea of this method is to imitate the way the human body observes objects and select more critical parts from a large amount of information to achieve the purpose of feature extraction. The Attention Mechanism is used in two measures in the improved LSTM to classify traffic data. One is to use the Attention Mechanism to determine which dimensions play a critical role in classification. The other is that the data sequence results obtained by the BiLSTM output layer are added to the Attention Mechanism layer to obtain a more accurate classification. The calculation equations are shown in the following equations:

$$u_t = \tanh(W_w h_t + b_w), \quad (26)$$

$$a_t = \text{soft max}(u_t^T, u_w), \quad (27)$$

$$v = \sum a_t h_t. \quad (28)$$

In equations (26)–(28), u_t is the attribute representation of the output vector h_t of the BiLSTM output layer, a_t is the weight of importance, v is the result of the importance weighting operation on the output vector h_t , and u_w is a randomly generated context vector during training.

The final result is input to the fully connected neural network layer for classification, and the prediction result is obtained. The pseudocode of the algorithm for

implementing traffic data classification using improved LSTM is shown in Algorithm 3.

5. Experimental Results and Analysis

This section first introduces the experimental environment and the dataset used. Then, we compare with other methods and debug the internal structure and parameters of the model to illustrate the superiority of the proposed scheme of communication intrusion detection scenarios in AMI.

5.1. Experimental Settings and Dataset Description. The experiment was run on a machine with Windows 10 operating system, Intel Core i9-9900K CPU, NVIDIA RTX2080 Ti GPU, and 32 GB RAM. To compare the running time of the proposed model on CPU and GPU, we conducted the comparative experiment on a machine with Windows 10 operating system with Intel Core i7-5500U CPU and 8 GB RAM. The methods mentioned in the scheme are all programmed with Python 3.7, and the compiler used is Pycharm2020. A large number of programming libraries in python are employed in programming, such as Numpy, Pandas, Keras, and Sklearn. Numpy provides the basic packages for data analysis and high-performance scientific computing, which can calculate the matrix precisely and work with vectors. Pandas is a tool based on Numpy, including different libraries and various standard data types. It is used to accurately process large-scale datasets. Keras is the most important library in the process of programming, and it can run on TensorFlow or Theano. Our scheme is running on TensorFlow. Because some deep learning models (AE, LSTM) are used in the designed scheme, Keras provides a flexible deep learning framework for it. We can easily and quickly implement the scheme programming through Keras. Sklearn encapsulates a large number of machine learning algorithms, such as classification, regression, and clustering.

To evaluate our proposed communication intrusion detection scheme in AMI, the public intrusion detection standard dataset UNSW-NB15 is employed for verification. This dataset was created by the cyber security research group of the Australian Centre for Cyber Security (ACCS) [44], which addresses the issue of data redundancy in other datasets. The traffic obtained by the AMI system has the characteristic of more normal data and less intrusion attack data. An unbalanced dataset is needed to verify the proposed scheme. Table 2 contains the distribution of the data. The UNSW-NB15 dataset has 175341 records in the training dataset and 82332 records in the testing dataset. In addition to normal data records, the dataset has 9 types of intrusion attacks: Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms. There are 93000 normal records in the dataset, accounting for 36.09%. However, Shellcode and Worms have 1511 records and 174 records, respectively, accounting for 0.59% and 0.07%. Therefore, the UNSW-NB15 dataset meets the verification characteristics of the unbalanced dataset.

```

Input: original training dataset Original_train, testing dataset Original_test
Output: preprocessed training dataset Preprocessed_train, testing dataset Preprocessed_test
train = pd.read_csv (Original_train)
test = pd.read_csv (Original_train)
/* concat() complete data splicing */
Spliced_data = pd.concat([train, test])
/* get_dummies() complete one-hot encoding */
Encoded_data = get_dummies(Spl_data, ["Feature_1", "Feature_2", . . . , "Feature_n"])
Encoded_data.drop(["label", "attack_cat"])
/* MinMaxScaler() normalizes the data to [0, 1] */
Preprocessed_train = MinMaxScaler (Encoded_data, train, feature_range = (0, 1))
Preprocessed_test = MinMaxScaler (Encoded_data, test, feature_range = (0, 1))
End

```

ALGORITHM 1: Algorithm description of data preprocessing stage.

```

Input: Preprocessed_train, Preprocessed_test, Train_label, Test_label
Output: Encoded_train, Encoded_test, Train_label, Test_label
Load processed data Preprocessed_train, Preprocessed_test
While not reach terminating condition: n-layer autoencoder training ( $n = 1, 2, 3, 4$ )
  for Epoch in range (1, 100):
    /* complete filepath stitching */
    os.path.join()
    /* Save the model results after each epoch to filepath */
    AE_n_point = ModelCheckpoint (filepath, monitor = "val_loss", verbose = 1, save_best_only = True, mode = "min")
    /* Save the best model to prevent overfitting */
    AE_n_stops = EarlyStopping (monitor = "val_loss", patience = 10, mode = "min")
    break
  AutoEncoder_n.load_weights()
  Output the prediction result of this layer: layer_n_output, test_n_out
End While
Encoded_train = SAE_encoder.predict (train)
Encoded_test = SAE_encoder.predict (test)
/* save SAE final result */
np.save (Encoded_train, Encoded_test, Train_label, Test_label)
End

```

ALGORITHM 2: Using the SAE to implement feature dimensionality reduction algorithm.

Network communication technology is applied to AMI, and massive data present a multidimensional characteristic structure. The UNSW-NB15 dataset comprises 44 features [45], which like the characteristic of communication traffic with many features in AMI. Table 3 displays all the features and types of the dataset. These features are divided into 6 categories:

Flow features: it includes the data flow attributes that the communication terminal interacts with. In this dataset, only proto belongs to Flow Features, which is used to mark the transaction protocol used in the communication.

Base features: it shows the basic attributes of the traffic in the records, such as the features of the protocol connection.

Content features: it is related to the attributes of the TCP and the HTTP.

Time features: it includes all the time attributes of the data in the record, such as the arrival time of the data packet, the return confirmation time, and the survival time.

Additional generated features: it can be divided into two parts: one is general feature attributes, and the other is connection feature attributes. In general feature attributes, each feature gets its use from the defense point of view. Connection feature attributes only provide defenses in connection attempts.

Labeled features: it indicates whether the record is normal data or generated from an intrusion attack. Both the normal and attack records are marked with a Boolean type.

5.2. Performance Evaluation Metrics. We are required to set model evaluation standards to test the effectiveness of the designed communication intrusion detection scheme in

```

Input: Encoded_train, Encoded_test, Train_label, Test_label
Output: Classification result
Load feature reduced data Encoded_train, Encoded_test, Train_label, Test_label
Define parameters time_steps, batch_size
Train_label_ = np. insert (Train_label)
Test_label_ = np.insert (Test_label)
train_generator = TimeseriesGenerator (Encoded_train, Train_label_)
test_generator = TimeseriesGenerator (Encoded_test, Test_label_) /* define Attention Mechanism function */
def attention_3d_block (inputs)/* define Attention Mechanism function */
lstm1 = Bidirectional (LSTM (units = 24)) (input_traffic) /* define the first layer lstm */
Call the attention_3d_block () function to judge the criticality of the dimension
lstm2 = Bidirectional (LSTM (units = 12)) (attention_output) /* define the second layer lstm */
Input BiLSTM output layer results into Attention Mechanism layer
mlp = Dense (units = 6, activation = "relu") (attention_output2)
mlp2 = Dense (units = 1, activation = "sigmoid") (mlp) /* The fully connected neural network layer outputs the classification results */
for Epoch in range(1, 250):
    history = classifier.fit_generator (train_generator, steps_per_epoch, callbacks = [], validation_data = test_generator, validation_steps)
np.save (Classification result) /* save classification final result */
End

```

ALGORITHM 3: Use improved LSTM for data classification.

AMI. Intrusion attack traffic detection is a classification problem, and its performance metrics depend on the confusion matrix [46]. Table 4 enumerates the definition and explanation of each item in the confusion matrix.

The confusion matrix can be used to calculate the following performance metrics to evaluate our proposed scheme:

Accuracy can be used to represent the proportion of all traffic data (normal and intrusion attack) being classified correctly. The calculation equation is shown in the following equation:

$$\text{Accuracy} = \frac{TP + TN}{TP + FN + FP + TN}. \quad (29)$$

Precision can be used to express the probability that the data detected as a positive sample is truly a positive sample. The calculation equation is shown in the following equation:

$$\text{Precision} = \frac{TP}{TP + FP}. \quad (30)$$

Recall can express the ratio of intrusion attack traffic detected as positive samples by our proposed scheme to the overall intrusion attack traffic. *Recall* can be used as an important performance metric in the detection of datasets with an unbalanced category. The calculation equation is shown in the following equation:

$$\text{Recall} = \frac{TP}{TP + FN}. \quad (31)$$

F1_score is the harmonic value of *Precision* and *Recall*, and the calculation equation is shown in the following equation:

$$F1_score = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \quad (32)$$

FAR reports the ratio of the normal traffic detected as a positive sample to the overall normal traffic. The calculation equation is shown in the following equation:

$$\text{FAR} = \frac{FP}{FP + TN}. \quad (33)$$

5.3. Experimental Results. According to our proposed communication intrusion detection scheme in AMI, the UNSW-NB15 dataset is used for verification. First, we preprocess the dataset and use one-hot encoding to convert the nondigital features in the dataset into digital features. Then, we input the preprocessed data into the Stacked Autoencoder for feature dimensionality reduction. The encoded data are input into the improved LSTM model for classification. Finally, the result of classification is output through the fully connected layer. Table 5 is the confusion matrix obtained from the detection results of the proposed scheme on the UNSW-NB15 dataset. Table 6 comprises the results of performance metrics calculated by the values of items in the confusion matrix.

5.4. Comparison with Other Methods

5.4.1. Comparison with Traditional Machine Learning Methods. To prove the advantages of our proposed AMI communication intrusion detection scheme, traditional machine learning methods are utilized to classify and detect the UNSW-NB15 dataset. Table 7 compares their final results with the results of our proposed scheme. Traditional

TABLE 2: The distribution of records in the UNSW-NB15 dataset.

Number	Record class	Size	Distribution (%)
1	Normal	93000	36.09
2	Fuzzers	24246	9.41
3	Analysis	2677	1.04
4	Backdoors	2329	0.90
5	DoS	16353	6.35
6	Exploits	44525	17.28
7	Generic	58871	22.85
8	Reconnaissance	13987	5.43
9	Shellcode	1511	0.59
10	Worms	174	0.07
	Totals	257673	100

machine learning methods are not particularly effective for intrusion detection on the dataset with unbalanced categories. The best methods about the results are Random Forest and Decision Tree. The performance metric of Accuracy can reach 0.8583 and 0.8531, respectively. However, the FAR of these two traditional machine learning methods is very high. Support Vector Machines show good performance on the Binary Classification problems. After being used for the detection of intrusion attack traffic, the FAR value of the SVM model can reach 0.0079, but the classification Accuracy is only 0.6486. In summary, compared with traditional machine learning schemes, our proposed communication intrusion detection scheme in AMI greatly improves the Accuracy of intrusion detection, while ensuring low FAR. Especially, for datasets with unbalanced categories, it has better performance of classification.

5.4.2. Comparison with Recent Intrusion Detection Scheme.

Table 8 compares our proposed scheme with some recent intrusion detection schemes. DO_IDS [47] proposed an intrusion detection algorithm that relies on mixed data optimization. The Time-related NIDS [48] scheme uses a time-related deep learning method to detect intrusion attacks in the network. The SDAE+SVM [49] scheme also uses Denoising Autoencoder (DAE) to reduce the feature dimension. But, different from the scheme we proposed, this scheme finally uses the idea of SVM for classification.

Table 8 suggests that our proposed communication intrusion detection scheme in AMI can better detect intrusion attack traffic compared to the recently proposed papers of intrusion detection. Reference [48] used the time-series model for the scheme. Reference [49] used the feature dimensionality reduction of the Denoising Autoencoder. The scheme we propose is to use an improved LSTM to classify time series data based on the feature dimensionality reduction of SAE. So, we compare the accuracy and loss of each Epoch on the training dataset and the testing dataset of the proposed model and the two deep learning methods. Figure 10 is the curve drawn founded on the results obtained. The three models were trained with 180 Epochs, and the values of accuracy and loss after each Epoch were recorded. The proposed scheme converges significantly faster during training and testing, and the final classification accuracy of our model is considerably higher.

TABLE 3: Features in the UNSW-NB15 dataset.

Number	Feature	Feature type	Data type
1	proto	Flow features	Nominal
2	state	Base features	Nominal
3	dur	Base features	Float
4	sbytes	Base features	Integer
5	dbytes	Base features	Integer
6	sttl	Base features	Integer
7	dttl	Base features	Integer
8	sloss	Base features	Integer
9	dloss	Base features	Integer
10	service	Base features	Nominal
11	sload	Base features	Float
12	dload	Base features	Float
13	spkts	Base features	Integer
14	dpkts	Base features	Integer
15	swin	Content features	Integer
16	dwin	Content features	Integer
17	stcpb	Content features	Integer
18	dtcpb	Content features	Integer
19	smeansz	Content features	Integer
20	dmeansz	Content features	Integer
21	trans_depth	Content features	Integer
22	res_bdy_len	Content features	Integer
23	sjit	Time features	Float
24	djit	Time features	Float
25	rate	Time features	Float
26	sintpkt	Time features	Float
27	dintpkt	Time features	Float
28	tcprrt	Time features	Float
29	synack	Time features	Float
30	ackdat	Time features	Float
31	is_sm_ips_ports	Additional generated features	Binary
32	ct_state_ttl	Additional generated features	Integer
33	ct_flw_http_mthd	Additional generated features	Integer
34	is_ftp_login	Additional generated features	Binary
35	ct_ftp_cmd	Additional generated features	Integer
36	ct_srv_src	Additional generated features	Integer
37	ct_srv_dst	Additional generated features	Integer
38	ct_dst_ltm	Additional generated features	Integer
39	ct_src_ltm	Additional generated features	Integer
40	ct_src_dport_ltm	Additional generated features	Integer
41	ct_dst_sport_ltm	Additional generated features	Integer
42	ct_dst_src_ltm	Additional generated features	Integer
43	attack_cat	Labeled features	Nominal
44	Label	Labeled features	Binary

To compare the computational cost of the proposed scheme and other schemes, Figure 11 reports the running time for each parameter in the training dataset and the testing dataset to reach the set threshold. We use GPU to

TABLE 4: Explanation of the meaning of each item in the confusion matrix.

Items in confusion matrix	Explanation
TP	The number of intrusion attack traffic detected as positive samples
TN	The number of normal traffic detected as negative samples
FP	The number of normal traffic detected as positive samples
FN	The number of intrusion attack traffic detected as negative samples

TABLE 5: Confusion matrix over the UNSW-NB15 dataset using SAE + Attention – BiLSTM.

		Predicted	
		Anomalous	Normal
Actual	Anomalous	36275	313
	Normal	172	45160

accelerate the training speed of all models. Although the proposed scheme spends more time on training for each Epoch, less Epoch is used to reach the threshold. Considering the overall running time, we can complete intrusion detection faster in AMI. The time for the power system to build up a defense mechanism has been extended.

5.5. Comparison with Different Structures of SAE and LSTM.

To explore the influence of different structures of SAE and LSTM on our proposed scheme, experiments were carried out with different SAE structures and LSTM structures. The SAE network uses three different structures: {128, 64, 32, 32}, {128, 64, 32}, and {128, 32, 32}. LSTM uses two separate structures: the original model and the improved model. Table 9 highlights the different experimental performance results. When the Stacked Autoencoder structure adopts {128, 64, 32, 32} four hidden layer structures and the improved LSTM network in our scheme is employed to classification, the model has the best effect. But, the structure of a Stacked Autoencoder with four hidden layers will increase the convergence time of the algorithm. Improving LSTM to BiLSTM and adding the Attention Mechanism will also increase the amount of calculation in the model. Therefore, in the actual scenes in Smart Grid, it is necessary to comprehensively consider the detection accuracy, the required calculation configuration of the model, and the implementation time of the scheme. We should select the optimal structure for communication intrusion detection in AMI.

5.6. Comparison with Different Timesteps. Because our scheme uses the method of generating time series data in the final classification model, the selection of different timestep values will also affect the performance metrics of the model. In the LSTM classification model, timestep refers to how much data the current input data of the model is related to before. In the final classification process, through code debugging, the timestep value is selected in the set {4, 8, 12, 16, 20}, and we record the results of our model. Figure 12 shows the comparison of four performance metrics after different timesteps. With the increase of timesteps, the performance of various metrics gradually becomes better and eventually tends to a stable value. *Accuracy*, *Precision*,

and *FAR* have the best performance when the timestep value is equal to 20. *Recall* has the best performance when the timestep value is equal to 16, but the *Recall* value is only 0.0005 lower when the timestep value is equal to 20. However, while improving the performance of the model, the calculation time of the model should be taken into account. Figure 13 explains the calculation time of each Epoch for different timesteps. It indicates that the cost of increasing timesteps to obtain good performance is that the calculation time becomes longer.

We set different timestep values in the classification model. Figure 14 records the accuracy and loss values of the training dataset and the testing dataset after each Epoch. As the timestep value increases, the final performance that the model can eventually achieve becomes better and better, but this does not mean that the time to reach the best performance is getting faster. Figure 14 suggests that the overall convergence trend is almost the same in the process of the model reaching the best performance. When the timestep value increases evenly, the performance improvement of the model is less and less obvious.

5.7. Experiments on the NSL-KDD Dataset. The proposed scheme shows excellent performance on the UNSW-NB15 dataset. To ensure the robustness of the proposed scheme, we use the NSL-KDD dataset for experiments. The NSL-KDD dataset is improved based on the KDD Cup99 dataset [50]. Compared to KDD Cup99, NSL-KDD does not include redundant records in the training dataset and duplicate records in the testing dataset. This dataset also conforms to the essential characteristics of traffic in the AMI communication environment: data imbalance and multidimensional feature structure. In the NSL-KDD dataset, there are 125973 records in the training dataset and 22544 records in the testing dataset. Table 10 shows the distribution of various types of intrusion attacks. In addition to the data marked as Normal, there are samples of four types of intrusion attacks: DoS, U2R, R2L, and Probe. The dataset has 42 features (1 category feature, 7 discrete features, and 34 continuous features).

Under the same environmental setting as 5.1, we conducted experiments on the NSL-KDD dataset according to the proposed scheme. Table 11 is the confusion matrix obtained through the testing dataset. Through the confusion matrix, we calculate the performance metrics, as shown in Table 12. The proposed scheme is also applicable to the NSL-KDD dataset, showing excellent performance in intrusion attack detection. Unlike the UNSW-NB15 dataset, the testing dataset in NSL-KDD includes many new attack variants. Therefore, the scheme could detect new attack variants.

FIGURE 10: Changes in accuracy and loss of the training dataset and the testing dataset with the increase of Epoch. (a) Accuracy changes in the training dataset. (b) Loss changes in the training dataset. (c) Accuracy changes in the testing dataset. (d) Loss changes in the testing dataset.

FIGURE 11: The running time of each program when the set threshold is reached. (i) The accuracy of the training dataset reaches 0.95. (ii) The accuracy of the testing dataset reaches 0.93. (iii) The loss of the training dataset reaches 0.12. (iv) The loss of the testing dataset reaches 0.15.

TABLE 6: Results of each performance evaluation metric on the UNSW-NB15 dataset.

Evaluation index	Value
Accuracy	0.9941
Precision	0.9914
Recall	0.9952
F1_score	0.9933
FAR	0.0069

FIGURE 12: Continued.

FIGURE 12: The results of performance metrics with different timesteps. (a) Accuracy. (b) Precision. (c) Recall. (d) FAR.

FIGURE 13: The calculation time of each Epoch under different timestep values. (i) The machine configuration used is CPU. (ii) The machine configuration used is GPU 2080Ti.

6. Threats to Validity

In this section, we report possible threats to the validity of the proposed scheme.

6.1. Threats to Internal Validity. In the process of intrusion detection, the dependent variable is the performance evaluation metric, which is obtained finally. It is calculated based

on the confusion matrix of the classification results. Time performance is also used as one of the evaluation metrics in comparison with related work. However, we should ensure the consistency of the machines used when comparing the time performance of schemes. GPU will significantly accelerate the training speed of the model.

The independent variables that affect the dependent variable can be split into the structural setting of the model and the internal parameters of the model. In the structure of

FIGURE 14: Changes in performance metrics with increasing Epoch when setting different timesteps. (a) Accuracy changes in the training dataset. (b) Loss changes in the training dataset. (c) Accuracy changes in the testing dataset. (d) Loss changes in the testing dataset.

TABLE 7: The comparison of results between the proposed method and traditional machine learning methods.

Method	Accuracy	Precision	Recall	FAR
K-nearest neighbor	0.7544	0.7942	0.6977	0.1870
Naive Bayesian	0.8358	0.7399	0.8731	0.1869
Decision tree	0.8531	0.7811	0.8765	0.1624
Random forest	0.8583	0.8434	0.8400	0.1268
Logistic regression	0.8471	0.7212	0.9190	0.1917
Support vector machines	0.6486	0.9964	0.5599	0.0079
Multilayer perceptron	0.8095	0.7147	0.8350	0.2063
SAE + Attention-BiLSTM	0.9941	0.9914	0.9952	0.0069

TABLE 8: The comparison of results among the proposed scheme and recent intrusion detection papers.

Method	Accuracy	Precision	Recall	FAR
DO_IDS [47]	0.9282	0.9670	0.8966	0.0330
Time-related NIDS [48]	0.9793	0.9685	0.9848	0.0251
SDAE + SVM [49]	0.9311	0.9679	0.8879	0.0280
SAE + Attention-BiLSTM	0.9941	0.9914	0.9952	0.0069

TABLE 9: Evaluation results of SAE and LSTM with different structures.

Structures of SAE and LSTM		Accuracy	Precision	Recall	FAR
{128, 64, 32, 32}	Attention-BiLSTM	0.9941	0.9914	0.9952	0.0069
	LSTM	0.9859	0.9778	0.9905	0.0178
{128, 64, 32}	Attention-BiLSTM	0.9893	0.9822	0.9938	0.0142
	LSTM	0.9809	0.9682	0.9888	0.0252
{128, 32, 32}	Attention-BiLSTM	0.9861	0.9763	0.9924	0.0189
	LSTM	0.9767	0.9595	0.9880	0.0320

TABLE 10: The distribution of various types of intrusion attacks in the NSL-KDD dataset.

Dataset class	Attack class	Size	Distribution (%)
Training dataset	Normal	67343	53.46
	DoS	45927	36.46
	Probe	11656	9.25
	R2L	995	0.79
	U2R	52	0.04
Testing dataset	Normal	9711	43.07
	DoS	7458	33.08
	Probe	2421	10.74
	R2L	2754	12.22
	U2R	200	0.89

TABLE 11: Confusion matrix over the NSL-KDD dataset using SAE + Attention – BiLSTM.

		Predicted	
		Anomalous	Normal
Actual	Anomalous	11769	95
	Normal	46	12826

TABLE 12: Results of each performance evaluation metric on the NSL-KDD dataset.

Evaluation index	Value
Accuracy	0.9943
Precision	0.9961
Recall	0.9920
F1_score	0.9940
FAR	0.0036

the model, the number of layers in the SAE and the number of neurons in each layer will have a significant impact on the internal validity. When using LSTM for classification, choosing a bidirectional structure to extract features and adding an Attention Mechanism will improve accuracy. However, the complexity of the model structure will lead to an increase in the computational cost. Timesteps is one of the most important internal parameters, which affects the time performance of the model. In the programming models for deep learning, we also need to consider the threat of internal parameters (such as learning rate) to internal validity. Besides, some advanced technologies are used in the algorithm design of our scheme. For example, we use the *dropout* function to prevent the model from overfitting. Different probabilities of deactivation may cause the model to produce different classification results.

TABLE 13: Attack categories and subtypes of attacks in the NSL-KDD dataset.

Categories of attack	Subtype
DoS	apache2, back, land, mailbomb, neptune, pod, processtable, smurf, teardrop, udpstorm, worm
U2R	buffer_overflow, loadmodule, perl, ps, sqlattack, rootkit, xterm
R2L	ftp_write, guess_passwd, httptunnel, phf, imap, multihop, named, sendmail, snmpgetattack, snmpguess, spy, warezclient, warezmaster, xlock, xsnoop
Probe	ipsweep, portsweep, mscan, nmap, saint, satan

6.2. Threats to External Validity. Aiming at the characteristics of intrusion attack traffic in AMI, we propose a detection scheme. In the experiment, the UNSW-NB15 dataset was used to verify the effectiveness of the scheme. To ensure the robustness of the scheme, we use the NSL-KDD dataset to test. These two general datasets are collected by network communication, so the proposed scheme can adapt to other environments. In other industrial control traffic anomaly detection, there are also problems with high-dimensional features and data imbalance. We implement feature dimensionality reduction through the SAE part of the designed IDS and then use BiLSTM with Attention Mechanism to extract the bidirectional feature structure. Finally, the purpose of efficient intrusion detection can be achieved.

But in deep research, we found that the proposed scheme needs the ability to detect subtypes of attacks. For example, Table 13 shows the attack categories and subtypes of attacks in the NSL-KDD dataset. Based on the results of subtype intrusion attack detection, researchers can implement defense measures precisely.

7. Conclusion

In this paper, considering high-dimensional features of massive data and data imbalance in AMI, we propose the corresponding intrusion detection scheme. The scheme consists of two parts: feature dimensionality reduction and classification. In feature dimensionality reduction, we use the Stacked Autoencoder to convert the 196-dimensional original data features into 32-dimensional encoded data features. It could reduce the computational complexity of the model. In classification, the Attention Mechanism is applied to the BiLSTM model to determine the criticality of the dimensionality and select efficient features to improve the accuracy of classification. The proposed intrusion detection scheme is evaluated using the UNSW-NB15 dataset. Through experimental comparison: among all the performance indicators selected in this paper, the proposed communication intrusion detection scheme in AMI is much better than the methods based on traditional machine learning. In comparison with the new intrusion detection scheme, our scheme still shows good performance. By changing the structure of the SAE and LSTM models in the

scheme and debugging the timestep value, we explore the influence of the internal parameters of the model on the overall scheme. To ensure the robustness of the work, we test on the NSL-KDD dataset. It also shows superior performance.

In future work, although the accuracy of intrusion detection is improved and the FAR is reduced, the scheme we propose has a lot of time cost in terms of the SAE deep network structure and the calculation of the Attention Mechanism. Therefore, it is necessary to further explore how to reduce the computational cost of the model while ensuring high accuracy and low FAR. In the communication scenario of AMI, various new types of attacks emerge endlessly. In our future work, it is crucial to choose the structure and corresponding parameters to realize the optimized intrusion detection scheme and adapt to the changes of new attacks in AMI.

Data Availability

The UNSW-NB15 dataset and the NSL-KDD dataset used to support the findings of this study are included within the paper. Besides, other data are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No. 61772327), State Grid Gansu Electric Power Company Electric Power Research Institute (Grant No. H2019-275), and Shanghai Engineering Research Center on Big Data Management System (Grant No. H2020-216).

References

- [1] M. A. Faisal, Z. Aung, J. R. Williams, and A. Sanchez, "Data-stream-based intrusion detection system for advanced metering infrastructure in smart grid: a feasibility study," *IEEE Systems Journal*, vol. 9, no. 1, pp. 31–44, 2015.
- [2] F. M. Cleveland, "Cyber Security Issues for Advanced Metering Infrastructure (AMI)," in *Proceedings of the 2008 IEEE Power and Energy Society General Meeting - Conversion and Delivery of Electrical Energy In the 21st Century*, Pittsburgh, PA, USA, July 2008.
- [3] P. I. Radoglou-Grammatikis and P. G. Sarigiannidis, "Securing the smart grid: a comprehensive compilation of intrusion detection and prevention systems," *IEEE Access*, vol. 7, pp. 46595–46620, 2019.
- [4] A. Thakkar and R. Lohiya, "A review of the advancement in intrusion detection datasets," *Procedia Computer Science*, vol. 167, pp. 636–645, 2020.
- [5] S. Han, M. Xie, H. Chen, and Y. Ling, "Intrusion detection in cyber-physical systems: techniques and challenges," *IEEE Systems Journal*, vol. 8, no. 4, pp. 1052–1062, 2014.
- [6] V. Kumar, D. Sinha, A. K. Das, S. C. Pandey, and R. T. Goswami, "An integrated rule based intrusion detection

- system: analysis on UNSW-NB15 data set and the real time online dataset,” *Cluster Computing*, vol. 23, no. 2, pp. 1397–1418, 2019.
- [7] Z. Wang, S. Xu, G. Xu et al., “Game theoretical method for anomaly-based intrusion detection,” *Security and Communication Networks*, vol. 2020, Article ID 8824163, 10 pages, 2020.
 - [8] C. Gu, S. Zhang, and H. Lu, “Online Internet intrusion detection based on flow statistical characteristics,” *Knowledge Science, Engineering and Management*, vol. 7091, pp. 160–170, 2011.
 - [9] T. Mehmood and H. B. M. Rais, “Machine learning algorithms in context of intrusion detection,” in *Proceedings of the 2016 3rd International Conference on Computer and Information Sciences (ICCOINS)*, Kuala Lumpur, Malaysia, August 2016.
 - [10] N. Altwaijry, A. ALQahtani, and I. AlTuraiki, “A deep learning approach for anomaly-based network intrusion detection,” *Big Data and Security*, vol. 1210, pp. 603–615, 2020.
 - [11] D. Kwon, H. Kim, J. Kim, S. C. Suh, I. Kim, and K. J. Kim, “A survey of deep learning-based network anomaly detection,” *Cluster Computing*, vol. 22, no. S1, pp. 949–961, 2017.
 - [12] Y. Dong, R. Wang, and J. He, “Real-time network intrusion detection system based on deep learning,” in *Proceedings of the 2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS)*, Beijing, China, October 2019.
 - [13] S. A. Althubiti, E. M. Jones, and K. Roy, “LSTM for anomaly-based network intrusion detection,” in *Proceedings of the 2018 28th International Telecommunication Networks And Applications Conference (ITNAC)*, Sydney, Australia, November 2018.
 - [14] D. M. Farid, H. Nouria, and M. Z. Rahman, “Combining naive bayes and decision tree for adaptive intrusion detection,” *International Journal of Network Security & Its Applications*, vol. 2, no. 2, pp. 12–25, 2010.
 - [15] P. I. Radoglou-Grammatikis and P. G. Sarigiannidis, “An anomaly-based intrusion detection system for the smart grid based on cart decision tree,” in *Proceedings of the 2018 Global Information Infrastructure and Networking Symposium (GIIS)*, Thessaloniki, Greece, October 2018.
 - [16] B. Senthilnayagi, K. Venkatalakshmi, and A. Kannan, “Intrusion detection system using fuzzy Rough set feature selection and modified KNN classifier,” *International Arab Journal of Information Technology*, vol. 16, no. 4, pp. 746–753, 2019.
 - [17] R. Vijayanand, D. Devaraj, and B. Kannapiran, “Support vector machine based intrusion detection system with reduced input features for advanced metering infrastructure of smart grid,” in *Proceedings of the 2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS)*, Coimbatore, India, January 2017.
 - [18] T. Hurley, J. E. Perdomo, and A. Perez-Pons, “HMM-based intrusion detection system for software defined networking,” in *Proceedings of the 2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, Anaheim, CA, USA, December 2016.
 - [19] W. L. Al-Yaseen, Z. A. Othman, and M. Z. A. Nazri, “Multi-level hybrid support vector machine and extreme learning machine based on modified K-means for intrusion detection system,” *Expert Systems with Applications*, vol. 67, pp. 296–303, 2017.
 - [20] B. Ismail, R. In-Ho, and S. Ravi, “An intrusion detection system based on multi-level clustering for hierarchical wireless sensor networks,” *Sensors*, vol. 15, no. 11, pp. 28960–28978, 2015.
 - [21] F. Salo, A. B. Nassif, and A. Essex, “Dimensionality reduction with IG-PCA and ensemble classifier for network intrusion detection,” *Computer Networks*, vol. 148, pp. 164–175, 2019.
 - [22] J. Sun, X. Wang, N. Xiong, and J. Shao, “Learning sparse representation with variational auto-encoder for anomaly detection,” *IEEE Access*, vol. 6, pp. 33353–33361, 2018.
 - [23] S. Ali and Y. Li, “Learning multilevel auto-encoders for DDoS attack detection in smart grid network,” *IEEE Access*, vol. 7, pp. 108647–108659, 2019.
 - [24] A. Bhardwaj, V. Mangat, and R. Vig, “Hyperband Tuned deep neural network with well posed stacked sparse AutoEncoder for detection of DDoS attacks in cloud,” *IEEE Access*, vol. 8, pp. 181916–181929, 2020.
 - [25] J. Gao, L. Gan, F. Buschendorf et al., “LSTM for SCADA intrusion detection,” in *Proceedings of the IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, Victoria, Canada, August 2019.
 - [26] A. F. M. Agarap, “A neural network architecture combining gated recurrent unit (GRU) and support vector machine (SVM) for intrusion detection in network traffic data,” in *Proceedings of the 2018 10th International Conference On Machine Learning And Computing (ICMLC 2018)*, Macau, China, February 2018.
 - [27] B. Roy and H. Cheung, “A deep learning approach for intrusion detection in internet of things using bi-directional long short-term memory recurrent neural network,” in *Proceedings of the 2018 28th International Telecommunication Networks and Applications Conference (ITNAC)*, Sydney, Australia, November 2017.
 - [28] M. Khan, M. Karim, and Y. Kim, “A scalable and hybrid intrusion detection system based on the convolutional-LSTM network,” *Symmetry*, vol. 11, no. 4, p. 583, 2019.
 - [29] B. Riyaz and S. Ganapathy, “A deep learning approach for effective intrusion detection in wireless networks using CNN,” *Soft Computing*, vol. 24, no. 22, pp. 17265–17278, 2020.
 - [30] Z. Lin, Y. Shi, and Z. Xue, “Generative adversarial networks for attack generation against intrusion detection,” *CoRR Abs*, vol. 2, no. 1, pp. 12–18, 2018.
 - [31] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, “Greedy layer-wise training of deep networks, in Advances in neural information processing systems 19,” in *Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems*, Vancouver, Canada, December 2006.
 - [32] H. Ma, S. Ma, Y. Xu et al., “Image denoising based on improved stacked sparse denoising autoencoder,” *Computer Engineering and Applications*, vol. 54, no. 04, pp. 199–204, 2018.
 - [33] B. Abolhasanzadeh, “Nonlinear dimensionality reduction for intrusion detection using auto-encoder bottleneck features,” in *Proceedings of the 2015 7th Conference On Information And Knowledge Technology (IKT)*, Urmia, Iran, May 2015.
 - [34] X. Wang and X. Lu, “A host-based anomaly detection framework using XGBoost and LSTM for IoT devices,” *Security and Communication Networks*, vol. 2020, Article ID 8838571, 13 pages, 2020.
 - [35] R. Hwang, M. Peng, V. Nguyen, and Y. Chang, “An LSTM-based deep learning approach for classifying malicious Traffic at the packet level,” *Applied Sciences*, vol. 9, no. 16, p. 3414, 2019.
 - [36] E. Bisong, *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners*, Apress, New York, NY, USA, 2019.

- [37] J. Zabalza, J. Ren, J. Zheng et al., "Novel segmented stacked autoencoder for effective dimensionality reduction and feature extraction in hyperspectral imaging," *Neurocomputing*, vol. 185, pp. 1–10, 2016.
- [38] V. Singhal, A. Gogna, and A. Majumdar, "Deep dictionary learning vs deep belief network vs stacked autoencoder: an empirical analysis," in *Proceedings of the International Conference on Neural Information Processing*, vol. 9950, Springer International Publishing, Bangkok, Thailand, November 2016.
- [39] S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," 2015, <https://arxiv.org/abs/1502.03167>.
- [40] W. Chen, S. Yang, X. A. Wang, W. Zhang, and J. Zhang, "Network malicious behavior detection using bidirectional LSTM," *Advances in Intelligent Systems and Computing*, vol. 772, pp. 627–635, 2018.
- [41] N. Xianyang, C. Yinbao, and W. Zhongyu, "Remote sensing semantic segmentation with convolution neural network using attention mechanism," in *Proceedings of the 2019 14th IEEE International Conference on Electronic Measurement & Instruments (ICEMI)*, Changsha, China, November 2019.
- [42] L. Shen, B. Zou, Y. Hong et al., "The emerging enernet: convergence of the smart grid with the Internet of Things," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Hong Kong, China, November 2017.
- [43] Y.-L. Li, S. Wang, and HAR-Net, "HAR-net: joint learning of hybrid attention for single-stage object detection," *IEEE Transactions on Image Processing*, vol. 29, pp. 3092–3103, 2020.
- [44] N. Moustafa and J. Slay, "The evaluation of Network Anomaly Detection Systems: statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set," *Information Security Journal: A Global Perspective*, vol. 25, no. 1–3, pp. 18–31, 2016.
- [45] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive dataset for network intrusion detection systems (UNSW-NB15 network dataset)," in *Proceedings of the 2015 Military Communications and Information Systems Conference (MilCIS)*, Canberra, Australia, November 2015.
- [46] A. D. Forbes, "Classification-algorithm evaluation: five performance measures based on confusion matrices," *Journal of Clinical Monitoring*, vol. 11, no. 3, pp. 189–206, 1995.
- [47] J. Ren, J. Guo, W. Qian et al., "Building an effective intrusion detection system by using hybrid data optimization based on machine learning algorithms," *Security and Communication Networks*, vol. 2019, Article ID 7130868, 11 pages, 2019.
- [48] Y. Lin, J. Wang, Y. Tu, L. Chen, and Z. Dou, "Time-related network intrusion detection model: a deep learning method," in *Proceedings of the 2019 IEEE Global Communications Conference (GLOBECOM)*, Waikoloa, USA, December 2019.
- [49] W. Wang, X. Du, D. Shan, and N. Wang, "A hybrid cloud intrusion detection method based on SDAE and SVM," in *Proceedings of the 12th International Conference on Intelligent Computation Technology and Automation (ICICTA)*, Xiangtan, China, October 2019.
- [50] M. Tavallaei, E. Bagheri, W. Lu, and A. A. Ghorbani, "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, Canada, July 2009.