Efficient BiSRU Combined With Feature Dimensionality Reduction for Abnormal Traffic Detection

PENGPENG DING, JINGUO LI, (Member, IEEE), MI WEN, (Member, IEEE), LIANGLIANG WANG, (Member, IEEE), AND HONGJIAO LI

College of Computer Technology and Science, Shanghai University of Electric Power, Shanghai 200090, China
Corresponding author: Jinguo Li (lijg@shiep.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61702321, Grant 61872230, Grant 61802249, Grant 61802248, and Grant U1936213; and in part by the Shanghai University Young Teacher Training Support Program under Grant ZZsdl18006.

ABSTRACT Abnormal traffic detection is an important network security technology to protect computer systems from malicious attacks. Existing detection methods are usually based on traditional machine learning, such as Support Vector Machine (SVM), Naive Bayes, etc. They rely heavily on manual design of traffic features and usually shallow feature learning, which get a low accuracy for high-dimensional traffic. Although the method based on Long Short-Term Memory (LSTM) has an excellent ability to detect abnormal traffic. The sequence-dependent structure of LSTM cannot realize parallel computation, which leads to slow model training and limits its applicability. To address the above problem, we propose an efficient Bidirectional Simple Recurrent Unit (BiSRU) combined with feature dimensionality reduction for abnormal traffic detection. Specifically, in order to perform feature dimensionality reduction on the original high-dimensional network traffic, we design a stack Sparse Autoencoder (sSAE) to extract the compressed high-level features. For the purpose of realizing efficient parallel computation and accurate feature extraction, a BiSRU is utilized to extract the bidirectional structural features of the traffic. Finally, the experimental results show that our proposed method significantly outperforms existing methods in terms of accuracy and training time. The method we propose can timely and accurately detect various abnormal traffic and achieve effective network security protection.

INDEX TERMS Abnormal traffic detection, dimensionality reduction, sSAE, BiSRU.

I. INTRODUCTION

With the rapid development of information technology, the structure of the internet has become more and more complex [1]. It plays an essential role in economic and social development [2]. However, more and more network security issues are exposed, especially abnormal network traffic [3]. Abnormal network traffic has the characteristics of sudden onset and unknown precursor. It can bring great harm to computer equipment, seriously affecting the security and performance of the network. Therefore, timely detection of abnormal network traffic and accurate judgment of the cause of network abnormality is of great significance for ensuring the reliability of the network [4]. Abnormal network traffic detection has also attracted extensive attention in the industry [5], [6].

The Intrusion Detection System (IDS) provides an effective way for abnormal traffic detection. As shown in Fig. 1, it is usually placed behind a gateway or firewall to capture and to monitor all incoming and outgoing network packets [7]. The IDS passively monitor the original traffic transmitted on the network and extract feature information from it, and then identify abnormal traffic by comparing it with known normal features and attack features. Thus, the IDS is a reasonable supplement to the firewall, helping the system to deal with network attacks, expanding the system security management capabilities (including security auditing, monitoring, attack identification and response), and improving the integrity of the information security infrastructure [8].

Abnormal flow detection presents two major challenges. Firstly, it is tough to construct traffic features because not
FIGURE 1. Abnormal traffic detection runs in a network intrusion detection system. It is divided into three parts: information collection, information analysis and result processing. The information collection part includes the collection of network data and user status and behavior. The information analysis section generates an alert and sends it to the console when it detects some misuse pattern. The result processing section takes the appropriate action according to the predefined response generated by the alarm.

Each feature information is directly related to anomalies. Secondly, in terms of feature dimensions, one-dimensional features may be difficult to characterize fully, but multi-dimensional features have a massive impact on computation.

There have been many studies on abnormal traffic detection, mainly include traditional machine learning methods and deep learning methods [9]. The traditional machine learning methods are just shallow feature learning. They have certain limitations when processing complex data. The feature processing that traditional machine learning must do is time-consuming and requires specialized knowledge. The performance of most machine learning algorithms depends on the accuracy of the extracted features. Deep learning methods reduce the manual design effort of feature extractors for each problem by automatically retrieving advanced features directly from original data. Deep learning methods provide excellent feature representation learning capabilities [10]. It has the characteristics of self-learning and meets the needs of accurate real-time detection. Besides, a large amount of network data makes deep learning methods more effective than traditional machine learning methods.

In order to meet the above problems and challenges, we design to apply the excellent feature learning capabilities of deep learning to achieve highly accurate network abnormal traffic detection. The main contributions of this article can be summarized as follows:

- **First**, to extract the advanced features of compression from the raw high-dimensional data, we propose a stack Sparse Autoencoder (sSAE) to perform feature dimensionality reduction on the input traffic data. Which maps the data in the original high-dimensional space to the low-dimensional space and extract the comprehensive features. The sSAE can reduce the calculation amount of the model and running time.
- **Second**, to achieve parallel computing and accurate feature learning, we propose a BiSRU based on Simple Recurrent Unit (SRU) and apply it into abnormal traffic detection. It can extract the bidirectional structural features of the traffic and realize accurate detection.
- **Third**, we compare the proposed method with other traditional machine learning methods and deep learning methods. Experimental results show that our proposed method is better than state-of-the-art methods in terms of accuracy and training time.

The rest of this article is organized as follows: We discussed related work in Section II and introduced the basic theory in Section III. In Section IV, we introduce our propose detection model, and then in Section V, we introduce experimental comparative analysis. Finally, we conclude Section VI.

II. RELATED WORK
This section provides an account of previous studies on abnormal traffic detection using traditional machine learning and deep learning.

A. TRADITIONAL MACHINE LEARNING
Most of the previous studies are based on traditional machine learning methods, such as Support Vector Machine (SVM) [11], Naive Bayes [12] and Decision Tree [13]. Naive Bayes is an important machine learning algorithm, which is widely used in the field of classification. Panda and Patra [14] used the Naive Bayes algorithm for abnormal traffic detection. This algorithm has been tested on the KDD Cup99 dataset and has achieved good results in terms of false alarm rate and calculation time. Ashraf et al. [15] applied Naive Bayes to network intrusion detection. Their idea is to use a Bayesian algorithm to select the most likely category based on the premise of feature independence. Farid et al. [16] proposed a hybrid IDS of Naive Bayes and Decision Tree, and the detection rate on the KDD Cup99 dataset reached 99.63%. Nevertheless, the KDD Cup99 dataset is an outmoded dataset, which has much redundancy and cannot fully reflect the characteristics of abnormal traffic in modern networks. K. Rai et al. [17] apply the Decision Tree C4.5 to perform intrusion detection experiments on the NSL-KDD dataset. In this work, 16 attributes were selected as the detection features of the dataset. But the
B. DEEP LEARNING

Deep learning is a branch of machine learning. With the continuous development of big data and computing power, deep learning methods are rapidly emerging and have been widely used in various fields, such as image detection and speech recognition [20]. At the same time, many studies are applying deep learning to abnormal traffic detection. Recurrent Neural Network (RNN) is a deep learning model and widely used in time series data learning. In 2017, Yin et al. [21] proposed a RNN-IDS algorithm for intrusion detection system. This method is applied to the NSL_KDD dataset, and it is proved to be superior to traditional machine learning methods. LSTM is a variant of RNN, and it solves the shortcoming of gradient vanishing and gradient explosion of RNN. In [22], Roy and Cheung proposed a Bidirectional Long Short-Term Memory Recurrent Neural Network (BLSTM RNN) for intrusion detection in the Internet of Things (IoT). This model achieves a detection accuracy rate of approximately 97% on the UNSW_NB15 dataset. However, LSTM has a defect of not being able to parallelize calculations, when it computes the current state information, it must wait until the previous state to complete the computation. In 2018, Lei et al. proposed the Simple Recurrent Units (SRU) model based on LSTM [23]. This model uses a simpler structure to remove the previous sequence dependency and implements parallel computing. SRU achieves 5-8 times faster training speed than LSTM on classification domain and conversational systems domain.

Based on the above works, traditional machine learning methods that are typically used in abnormal flow detection often fail and cannot detect many known and new security threats, largely because those approaches provide less focus on accurate feature selection and classification. It is often inefficient for large scale network flow. For the current deep learning methods like LSTM, they often pay more attention to the improvement of the model and they are very time consuming. To address above problems, we propose an efficient BiSRU model combined with feature dimensionality reduction for abnormal traffic detection. Specifically, we propose an sSAE to perform feature dimensionality reduction on the original high-dimensional network traffic data. sSAE can not only help to represent the input data but also improve the calculation efficiency. BiSRU can learn the bidirectional structural features of network traffic through the input forward and backward, and achieve accurate detection with less model training time. Finally, we conducted adequate experiments on the UNSW_NB15 network intrusion detection benchmark datasets to verify our proposed model.

III. BASIC THEORY

A. SPARSE AUTOENCODER

The Sparse Autoencoder (SAE) is an unsupervised learning method. It provides powerful non-linear generalization and widely used in image denoising [24] and dimensionality reduction [25].

As shown in Fig. 2, the SAE model consists of three parts: input layer, hidden layer and output layer.

As a conventional autoencoder, the loss function $J(W, b)$ in the decoding phase is calculated as:

$$J(W, b) = \frac{1}{M} \sum_{d=1}^{M} \left( \frac{1}{2} \left\| x^d - \hat{x}^d \right\|^2 + \frac{\lambda}{2} \sum_{i=1}^{n_i-1} \sum_{j=1}^{S_i} \sum_{l=1}^{S_{i+1}} (W^{(l)}_{ji})^2 \right).$$

where the first term is the reconstruction Mean Squared Error (MSE) [26] and another term is the regularization for
FIGURE 3. The training structure of sSAE. It mainly includes two phases: pre-training and fine-tuning. Each SAE is trained with a large amount of unlabeled data. Optimize the weight parameters of the sSAE using a small amount of labeled data.

avoiding over-fitting. λ is the weight decay coefficient, \( n_l \) is the number of layers, \( s_l \) denotes the neuron number in layer \( l \), and \( W_{ij}^{(l)} \) denotes the connecting weight between neuron \( i \) in layer \( l+1 \) and neuron \( j \) in layer \( l \).

The SAE incorporates the Kullback-Leibler (KL) divergence to overcome feature redundancy, the selective activation of neurons helps to make the encoded data more characteristic and easier to recognize [27]. The KL divergence is defined as:

\[
KL(\rho \parallel \hat{\rho}_j) = \rho \log \frac{\rho}{\hat{\rho}_j} + (1 - \rho) \log \frac{1 - \rho}{1 - \hat{\rho}_j}. \tag{4}
\]

\[
\hat{\rho}_j = \frac{1}{M} \sum_{d=1}^{M} h_d^j. \tag{5}
\]

where \( \rho \) is the proportion of activated neurons, \( \hat{\rho}_j \) is the average activation degree of each hidden unit, \( M \) is the number of hidden neurons.

Therefore the loss function in the SAE is calculated as:

\[
J_{\text{sparse}}(W, b) = J(W, b) + \mu \sum_{j=1}^{M} KL(\rho \parallel \hat{\rho}_j). \tag{6}
\]

where \( \mu \) is the weight used to control the sparsity penalty term.

B. LONG SHORT-TERM MEMORY

Long Short-Term Memory (LSTM) is a variant of RNN, which overcomes the problems of gradient vanishing and gradient exploding [28]. LSTM has been widely used in Natural Language Processing [29], speech recognition [30] and time series prediction [31]. The structure of a LSTM cell is shown in Fig. 4. The correlation processing of LSTM are updated as follows:

\[
f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f). \tag{7}
\]

\[
i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i). \tag{8}
\]

\[
\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C). \tag{9}
\]

\[
C_t = f_t \ast C_{t-1} + i_t \ast \tilde{C}_t. \tag{10}
\]

\[
o_t = \sigma(W_o[h_{t-1}, x_t] + b_o). \tag{11}
\]

\[
h_t = o_t \ast \tanh(C_t). \tag{12}
\]

where \( f_t \) is forget gate, \( i_t \) is input gate, \( o_t \) is output gate, \( \tilde{C}_t \) represents the state of the cell at the previous moment, \( C_t \) represents the current state of the cell, \( h_t \) represents the output of the current cell, and \( h_{t-1} \) represents the output of the cell at the previous moment. \( \sigma \) and \( \tanh \) are activation functions. \( W \) is weight matrix and \( b \) is bias.

IV. THE PROPOSED DETECTION MODEL

A. STACK SPARSE AUTOENCODER

The model of an sSAE is formed by stacking multiple single SAE [32]. It can extract the features efficiently and reduce the dimensionality of the features [33]. As shown in Fig. 3, training an sSAE mainly includes two phases: pre-training and fine-tuning.

1) PRE-TRAINING

A large number of unlabeled data are used to conduct individual training for each SAE with an unsupervised layer-by-layer
greedy learning method. The number of neurons in the output layer is consistent with the input layer. The process is as follows:

Step 1: use unlabeled data as input, randomly initialize the weight matrix $W$ and bias vector $b$.

Step 2: set hyperparameters, such as epoch $n$, sparsity penalty term $\mu$, learning rate $\lambda$, and dropout rate $\eta$, etc.

Step 3: train SAE in unsupervised and calculate the average activation degree $\hat{p}_j$ and loss function $J_{\text{sparsity}}$.

Step 4: update the weight $W_j$ and bias $b$ in each epoch until the loss is within the set threshold.

Step 5: take the output of the hidden layer of the previous SAE as the input of the next SAE, and repeat step 4 with layer-by-layer greedy learning method until the training of all the SAEs is completed.

2) FINE TUNING

A little labelled data was used for supervised learning, and the backpropagation algorithm was used to optimize the initial parameters. The process is as follows:

Step 1: connect the trained hidden layer of each SAE in an orderly manner to form the whole sSAE, set the weight matrix $W$ and bias vector $b$ obtained from the pre-training phase.

Step 2: set the initialization hyperparameters.

Step 3: add the softmax activation function to the last layer of sSAE for supervised classification.

Step 4: compare the predicted value with the label value to obtain the reconstruction loss, and use the backpropagation algorithm to fine-tune the whole sSAE network parameters.

B. SIMPLE RECURRENT UNIT

Much of the current progress in deep learning has come from increased modelling power and associated computation, which often involves larger, deeper neural networks [34]–[36]. However, although the deep neural network brings obvious improvement, it also takes a great deal of training time.

In order to solve the computational power of the training model, the parallelization method of using GPU for accelerated training has been widely used in the field of deep learning. Using GPU to accelerate the Convolutional Neural Network (CNN) has obvious improvement in training speed [37]. However, LSTM cannot implement the parallelization, because the calculation of $h_t$ must wait until the previous $h_{t-1}$ calculation is completed [38]. SRU is proposed to remove this limitation. In the existing deep learning library, an SRU implementation is about five times faster than an LSTM implementation [23]. In SRU, $h_t$ calculation is no longer dependent on the calculation of the previous moment, and the training speed after parallel processing is faster than LSTM. The SRU structure is shown in Fig. 5.

The complete architecture of a SRU decomposes to two sub-components: light recurrence and highway network. Due to enhanced parallelization and gradient propagation, the combination of two components makes the entire architecture simple and expressive, and easy to expand. First, a linear transformation is performed on the input $x_t$:

$$\hat{x}_t = W x_t.$$  \hspace{1cm} (13)

The calculation process of highway network is similar to LSTM. It reads the input vector $x_t$ in sequence and calculates the state vector $C_t$ to obtain sequence information. Specifically, a forget gate $f_t$ controls the information flow as follows:

$$f_t = \sigma(W_f x_t + v_f \odot C_{t-1} + b_f).$$ \hspace{1cm} (14)

where $\odot$ denotes an element-wise multiplication. The state vector $C_t$ is determined based on the adaptive average of $f_t$ for the previous state vector $C_{t-1}$ and the current observation $W x_t$ is calculated as:

$$C_t = f_t \odot C_{t-1} + (1 - f_t) \odot \hat{x}_t.$$ \hspace{1cm} (15)

In the previous gate recurrent architecture, $C_{t-1}$ is multiplied by a parameter matrix to calculate $f_t$, e.g., $f_t = \sigma(W_f x_t + v_f \odot C_{t-1} + b_f)$. However, including $v_f C_{t-1}$ makes it difficult to parallelize state calculations, because each dimension of $C_t$ and $f_t$ depends on all items of $C_{t-1}$, and the calculation must wait until $C_1$ is completely calculated. Use $v_f \odot C_{t-1}$ to simplify substitution in light recurrence components. Each dimension of the state vector becomes independent to achieve parallelization.

The highway network is helpful for gradient training. It uses reset gate $r_t$ to adaptively combine input $x_t$ and state $C_t$ from the light recurrence:

$$r_t = \sigma(W_r x_t + v_r \odot C_{t-1} + b_r).$$ \hspace{1cm} (16)

$$h_t = r_t \odot g(C_t) + (1 - r_t) \odot x_t.$$ \hspace{1cm} (17)

where $(1 - r_t) \odot x_t$ is a skip connection [39] that allows the gradient to propagate directly to the previous layer.

To solve a complex computation bottleneck of matrix multiplication, it can be batched in all time step, which significantly improves the calculation intensity and GPU utilization. The matrix multiplication is combined into one, and subsequent processing can be searched according to the index, as follows:

$$U^T = \begin{pmatrix} W & W_f & W_r \end{pmatrix} [x_1, x_2, \ldots, x_n].$$ \hspace{1cm} (18)
where \( U \in \mathbb{R}^{n \times 3d} \) is the calculated matrix, \( d \) is hidden state size and \( n \) is input data length. The operations between the elements in the sequence can be compiled and merged into a kernel function and parallelized in the hidden dimension.

\[
\begin{align*}
\bar{x} &= (x_1, x_2, \ldots, x_n), n \in N^* \text{ and its reconstruction } x_n'. 
\end{align*}
\]

After obtaining the optimal parameter \( \theta \), sSAE generates a function \( f(Wx_n + b) = x_n' \approx x_n \), which converts the input network traffics into a new feature representation.

![Algorithm 1 Stack Sparse Autoencoder (sSAE) for Feature Dimensionality Reduction](image)

**Algorithm 1 Stack Sparse Autoencoder (sSAE) for Feature Dimensionality Reduction**

**Input:** Traffic examples \( \bar{x} = (x_1, x_2, \ldots, x_n) \)

**Output:** High-level features encoded traffic data \( x_n' = f(Wx_n + b) \)

1) **Initialize:** the number of hidden neurons \( N_h \)

2) **for each training epoch** \( n = 1, 2, \ldots, N \) **do**

3) **for each hidden unit** \( j = 1, 2, \ldots, N_h \) **do**

4) Execute the forward propagation and compute the average activation degree of hidden units

5) **for each hidden layer** \( i = 1, 2, 3 \) **do**

6) Execute the Batch Normalization

7) Compute the loss function

8) Execute the back propagation to obtain parameters

9) **end**

10) **end**

11) **end**

In order to train a sSAE feature dimensionality reduction model, we chose to stack three single SAEs. The motivation for this is that the deep-level network structure can more accurately reconstruct the input data, ensuring that the high-level features of the encoded data are not lost while features dimensionality reduction. The key steps of the proposed sSAE are illustrated in Algorithm 1. It is worth noting that we use Batch Normalization (BN) processing [40] behind each hidden layer of sSAE. BN makes the distribution of input data in each layer of the network relatively stable, accelerating the model learning speed. More importantly, it can make the input data of the activation function fall in the gradient unsaturated zone through the normalize operation, alleviating the gradient vanishing problem caused by the network depth and complexity. Suppose \( m \)-size batch sample \( x \), do all the features in it as follows:

\[
\begin{align*}
\hat{x} &= \frac{x - \bar{x}}{\sqrt{\text{var}(x)}} \\
BN(x) &= \gamma \hat{x} + \beta.
\end{align*}
\]

where \( \bar{x} = \frac{1}{m} \sum_{i=1}^{m} x_i, \text{var}(x) = \frac{1}{m} \sum_{i=1}^{m} (x_i - \bar{x})^2 \).

2) **BiSRU for Classification**

We propose a BiSRU model based on SRU, which is more efficient for abnormal traffic detection. We take the network
traffic after feature dimensionality reduction as the high-level feature representation of the original data, and use them as the input vector of the BiSRU based abnormal traffic detection model, which is expressed as \( \tilde{X}' = (x'_1, x'_2, \ldots, x'_n) \).

Assuming that the predicted result vector is \( y(x) \in \{0, 1\} \), the detection model is trained by comparing the loss between the predicted value \( y(x) \) and the actual label value \( Y \). The input data of the model must be time series data, so we use TimeseriesGenerator to generate time-series for the input data first so that the input data becomes a sequence data with a time step. The key steps of BiSRU are shown in Algorithm 2. After calling TimeseriesGenerator to generate time series traffic, we set the learning rate decay to control the learning rate \( \eta \) in sections, and the purpose is to achieve more efficient learning and to train a neural network in different stages. In particular, BiSRU is trained by using mini-batch Stochastic Gradient Descent (SGD) [41]. It summarizes the forward information \( \tilde{h}_t \) and the backward information \( \tilde{h}_t \) to enhance feature extraction abilities.

\[
\tilde{h}_t = \text{SRU}(x_t), \quad t \in [1, T].
\]

\[
\tilde{h}_t = \text{SRU}(x_t), \quad t \in [1, T].
\]

For the binary classification problem, the label of the \( k \)-th network traffic sample is \( y(k) \in \{0, 1\} \), where 1 and 0 represent abnormal traffic and normal traffic, respectively. After completing the advanced feature learning process, for output layer a sigmoid classifier is utilized to encode reconstructed data as:

\[
f_{W^{(i)}}(x') = \frac{1}{1 + e^{-W^{(i)}x'}}.
\]

where \( f_{W^{(i)}} \) is a sigmoid activation function with parameters \( W^{(i)} \).

V. EXPERIMENT RESULTS AND ANALYSIS

A. DATASET DESCRIPTION

In order to evaluate our proposed method for abnormal traffic detection, the commonly used public standard intrusion detection UNSW_NB15 dataset is used for verification. The UNSW_NB15 is a new public data for intrusion detection. It was created by the Cyber Security Research Team of the Australian Cyber Security Centre (ACCS) to solve the data redundancy problem in the KDDCup99 and NSL_KDD datasets [42]. UNSW_NB15 has a reasonable number of records in training and testing datasets. The total number of records in the training dataset is 175341, and the testing dataset has 82332 records. Each traffic record contains 42 features and one class label. As shown in Table 1, the features can be categorized into five types: basic, flow, content, time, additional generated.

According to all features, UNSW_NB15 dataset has nine types of attacks, Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms, as shown in Table 2. We label all attack type records with abnormal traffic and normal records as normal traffic samples. The training dataset is used as model training, and the testing dataset is used as model verification.

B. DATA PREPROCESSING

1) ONE-HOT ENCODING

The original dataset contains 3 symbolic (non-numeric) features and 39 continuous (numeric) features. Non-numeric features cannot be directly used as model input, so non-numeric features need to be converted into numerical features first. Hence, we apply an one-hot encoding for non-numeric features, such as “proto”, “state” and “service”. We convert the “proto” into a numeric feature. The “proto” has three attributes, tcp, udp, and icmp. They can be encoded as \((1,0,0), (0,1,0), (0,0,1)\) respectively as follows:

\[
\text{one} - \text{hot encoding} = \begin{cases} 
(0, 0, 1), & \text{if the protocol is tcp.} \\
(0, 1, 0), & \text{if the protocol is udp.} \\
(1, 0, 0), & \text{if the protocol is icmp.}
\end{cases}
\]

We also convert “state” and “service” in the same way. After all the transformation, the 42-dimensional features of the UNSW_NB15 dataset are mapped into 196-dimensional features.
TABLE 1. Features detail of the UNSW_NB15 dataset. The features can be categorized into five types: basic, flow, content, time, additional generated.

| Feature type | No. | Feature name | Data type   |
|--------------|-----|--------------|-------------|
| Flow         | 1   | state        | Symbolic    |
|              | 2   | durt         | Continuous  |
|              | 3   | sztys        | Continuous  |
|              | 4   | dbyte        | Continuous  |
|              | 5   | stl          | Continuous  |
|              | 6   | dtl          | Continuous  |
|              | 7   | sloss        | Continuous  |
|              | 8   | dloss        | Continuous  |
|              | 9   | service      | Symbolic    |
|              | 10  | sload        | Continuous  |
|              | 11  | dload        | Continuous  |
|              | 12  | spkts        | Continuous  |
|              | 13  | dpkts        | Continuous  |
| Basic        | 15  | svin         | Continuous  |
|              | 16  | dvin         | Continuous  |
|              | 17  | stepb        | Continuous  |
|              | 18  | dtpcb        | Continuous  |
|              | 19  | smean        | Continuous  |
|              | 20  | dmean        | Continuous  |
|              | 21  | trans_depth  | Continuous  |
|              | 22  | response_body_len | Continuous |
| Content      | 23  | sjit         | Continuous  |
|              | 24  | djit         | Continuous  |
|              | 25  | rate         | Continuous  |
|              | 26  | sinpak       | Continuous  |
|              | 27  | dinpak       | Continuous  |
|              | 28  | tcprr         | Continuous  |
|              | 29  | syncack      | Continuous  |
| Time         | 30  | ackdat       | Continuous  |
| Additional Generated | 31  | ct_srvsrc    | Continuous  |
|              | 32  | ct_svdst     | Continuous  |
|              | 33  | ct_dstl      | Continuous  |
|              | 34  | ct_src      | Continuous  |
|              | 35  | ct_srcdportl|^ Continuous  |
|              | 36  | ct_dstportl  | Continuous  |
|              | 37  | ct_dstsrc   | Continuous  |
|              | 38  | ct_srcports | Continuous  |
|              | 39  | ct_state_ttl| Continuous  |
|              | 40  | ct_flwttpmtd | Continuous  |
|              | 41  | is_flplogin | Continuous  |
|              | 42  | ct_flpc   | Continuous  |

TABLE 2. Description of attacks in UNSW_NB15 dataset.

| Class name   | Train size | Distribution(%) | Test size  | Distribution(%) |
|--------------|------------|-----------------|------------|----------------|
| Normal       | 56000      | 31.94           | 37000      | 44.94          |
| Backdoor     | 1746       | 1                | 583        | 0.7            |
| Analysis     | 2000       | 1.14            | 677        | 0.82           |
| Puzzers      | 18184      | 10.37           | 6062       | 7.36           |
| Shellcode    | 1133       | 0.65            | 378        | 0.46           |
| Reconnaissance | 10491    | 5.98            | 3496       | 4.24           |
| Exploits     | 33393      | 19.94           | 11132      | 13.52          |
| DoS          | 12264      | 6.99            | 4089       | 4.97           |
| Worms        | 130        | 0.07            | 44         | 0.053          |
| Generic      | 40000      | 22.81           | 18871      | 22.92          |
| Total        | 175343     | 100             | 82332      | 100            |

2) NORMALIZATION
UNSW_NB15 dataset has multiple features with maximum and minimum ranges. These features are incomparable and unsuitable for processing. We normalize the data and scale it to a small specific interval. Min-Max normalization is used for mapping all features into the range $[0,1]$ by follow equation:

$$x_i' = \frac{x_i - \min(x_i)}{\max(x_i) - \min(x_i)}.$$  \hspace{1cm} (23)

C. PERFORMANCE METRICS
The performance metrics for abnormal traffic detection depend on the confusion matrix. [43]. The detailed description of the values in confusion matrix can be defined as follows:

- True positive ($TP$): anomaly traffic instances correctly classified.
- False positive ($FP$): normal traffic instances wrongly classified.
- True negative ($TN$): normal traffic instances correctly classified.
- False negative ($FN$): anomaly traffic instances wrongly classified.

These four items are used to generate the following performance evaluation metrics:

- **Accuracy**: evaluate the overall success rate of the model in detecting normal records and abnormal traffic, and is calculated as:
  $$\text{Accuracy} = \frac{TN + TP}{TP + FP + TN + TP}. \hspace{1cm} (24)$$
- **Precision**: evaluate the proportion of the actual normal samples to the sample divide into normal, The formula is:
  $$\text{Precision} = \frac{TP}{TP + FP}. \hspace{1cm} (25)$$
- **Recall**: evaluate the ratio of correctly classified abnormal traffic instances to the total number of abnormal traffic instances. The calculation formula is:
  $$\text{Recall} = \frac{TP}{FN + TP}. \hspace{1cm} (26)$$
- False Positive Rate ($FPR$): evaluate the proportion of normal instances that are misclassified as abnormal traffic in the total number of normal instances. The formula is:
  $$\text{FPR} = \frac{FP}{FP + TN}. \hspace{1cm} (27)$$

D. PERFORMANCE COMPARISON
1) COMPARISON METHODS AND PARAMETERS SETTING
All methods are implemented in Python 3.6 and Keras. The experiments are performed on a machine with Intel Core i7-9750H CPU, NVIDIA RTX2060 GPU, and 8 GB RAM. In order to evaluate the performance of our proposed model, we performed experiments on UNSW_NB15 dataset. The comparison machine learning and deep learning methods selected are as follows:

- Support Vector Machine (SVM) [11]: Linear SVM.
- Naive Bayes (NB) [12]: NB for Classification.
- Decision Tree (DT) [13]: IDS based on DT (DTB-IDS).
- DT (J48) + GR [18]: DT (J48) combined with Gain Ratio (GR) filter model. GR is used for feature selection and DT (J48) is used for intrusion detection.
- Autoencoder + SVM [19]: A deep learning method Autoencoder is used to reduce features dimensionality. SVM is used as decision engine with the help of the Artificial Bee Colony algorithm for finding the best parameter. The number of neurons in each layer of the Autoencoder is set as 42, 128, 64, 10, 64, 128, 42.
- RNN-IDS [21]: RNN for IDS. Referring to the original parameter with the best result in the original paper, the number of epoch is 100, 80 hidden nodes and the learning rate is set to 0.1.
- BLSTM RNN [22]: Bidirectional Long Short-Term Memory RNN for intrusion detection. Referring to the parameters in the original paper, the number of hidden layers is 3, the batch-size is 132, the time step is 60, the dropout rate is 0.8, the learning rate is 0.001, and the epoch is 100.

**TABLE 3. Initialization parameter setting.**

| Parameter         | Value          |
|-------------------|----------------|
| **sSAE**          |                |
| Number of input neurons | 196            |
| Number of hidden layers | 2              |
| Number of first hidden neurons | 128           |
| Number of second hidden neurons | 32            |
| \( \rho \)       | 0.04           |
| Dropout rate      | 0.1            |
| Activation function | ReLU           |
| **BiSRU**         |                |
| Dropout rate      | 0.5            |
| Activation function | ReLU           |
| Optimizer         | mini-batch SGD |
| time step         | 10             |

- Our proposed method: As shown in Table 3, the nodes in each layers is set to \( \{196, 128, 32\} \). \( \rho \) and learning rate are set to 0.04 and 0.001 respectively. In BiSRU, time step = 10, the last decoder layer adopts sigmoid activation function and the rest layes are followed by the ReLU activation function and the optimizer uses mini-batch SGD.

2) METHOD COMPARISON
In order to evaluate the performance of our proposed model, we conduct all experiments on the UNSW_NB15 dataset. Table 4 lists the performance comparison between our proposed method and the state-of-the-art methods. All models are trained on the training dataset, and the model is verified on the testset. In each set of experiments, we evaluated eight methods, including our proposed method on four performance indicators (Accuracy, Precision, Recall and FPR). The experimental results in Table 4 show that, compared with other traditional machine learning and deep learning methods, our proposed method can achieve the highest accuracy of 0.9941, Precision of 0.9904, Recall of 0.9964, and FPR of 0.0077, which means our proposed method has better performance than other methods in detecting abnormal traffic. Fig. 7 is a comparison of the training accuracy and testing accuracy and loss convergence between our proposed sSAE+BiSRU and two deep learning methods. All models have been trained for 100 epochs, and performance indicators have been evaluated after each epoch. In contrast, the proposed method has a much faster loss convergence rate during training and testing and can obtain the best results faster to improve accuracy, which is obviously better than other methods.

We further plot the Receiver Operating Characteristic (ROC) curve of our proposed sSAE+BiSRU and state-of-the-art methods on UNSW_NB15 as Fig. 8. The ROC curve can easily detect the performance of any threshold value and select the best diagnostic threshold value. The closer the ROC curve is to the upper left corner, the more accurate the testing will be. The point on the ROC curve closest to the upper left corner is the best threshold with the fewest errors and the fewest total false positives and false negatives. The ROC curve of sSAE+BiSRU is the closest one to the upper left corner, indicating a better generalization ability against the other methods. All the results reported above demonstrate that sSAE+BiSRU outperforms its competitors. We can conclude that sSAE+BiSRU is capable of efficient intrusion detection.

In the following, we further examine the robustness of sSAE+BiSRU with different training ratios based on UNSW_NB15 dataset. Fig. 9 reports the detailed performance of the model under different training ratios. We can see that sSAE+BiSRU achieve the highest Accuracy and Recall when the training ratio is 80%. With a training ratio of 70%, the highest Precision and lowest FPR are obtained. In short, it is indeed necessary to control the training ratio to bring performance improvement to abnormal traffic detection. For our proposed sSAE+BiSRU model, it can be found that the results obtained with different training rates are different, but the results within this range are still relatively excellent. This also means that the proposed method has a certain robustness to the data set, and shows high detection accuracy and low FPR in different scale training dataset.

3) COMPUTATION COMPARISON
To deepen this investigation, Table 5 reports the number of training parameters and running time required for both the proposed method and state-of-the-art methods. We use GPU to accelerate the training speed of all models. It can be noticed that, when training on the UNSW_NB15 dataset, the proposed method has fewer trainable parameters and lower training time and testing time. We use a small number of parameters as possible in the structure, which can realize efficient parallel computation.

E. RESULT OF DIMENSIONALITY REDUCTION
In the first stage, to further improve the Accuracy of abnormal traffic detection and effectively reduce the feature dimensionality. The relevant results of the dimensionality reduction
The performance comparison of our proposed method and other methods is shown in Table 4. The proposed method outperforms other methods in terms of accuracy, precision, and recall. The key factors that affect the performance of sSAE are analyzed. Four sSAE models with different structures are used to change the number of stack layers, nodes, and ρ. The performance comparison of different structures is obtained, as shown in Fig. 10. The parameter of sSAE has an important influence. After data preprocessing, two layers and a three layers sSAE are set for data dimensionality reduction. It is the first reduction of the original data, which determines the ability to extract the features of the input layer. On the one hand, the increase in the number of nodes can strengthen the learning ability of the network. On the other hand, it will affect the generalization ability of the network, which causes over-fitting, and the calculation will also be greatly increased. Therefore, set the number of nodes in the first hidden layer is set to 128. The softmax activation function is combined with an output layer to estimate the probability. When the number of hidden neurons is large, the compressed representation of the input cannot be obtained without sparseness. The sparse factor ρ can make the average activation of hidden neurons extremely small. ρ is usually a small value close to 0, so we set it to {0, 0.02, 0.04, 0.06} respectively. Finally, the performance indexes of sSAE with different structures are compared under different ρ situations.

In order to prevent over-fitting and reduce computational overhead, we obtain results by setting different numbers of layers (two and three layers) and the number of hidden nodes to obtain different dimensionality reduction effects. By comparing the sSAE performance indicators of four different structures, we found that: (1) when the number of stacked layers of sSAE is three, the overall performance of the four indicators is better than that of two layers; (2) compared with 64 nodes in the second hidden layer, a better result can be achieved when the number of nodes is 32; (3) among the four model results, the best results are obtained when the structural model is {128, 32, 32}.

**F. EVALUATION FOR BiSRU**

In the second stage, the encoded data which from sSAE dimension reduction is input to the time-related model based on RNN variants. The four different variants include SimpleRNN, LSTM, GRU and SRU. We use tanh activation function and Adam optimizer. The Adam optimizer can be used to adapt the learning rate, reduce the parameter setting steps and have an efficient calculation way. In order to verify the validity of the time series model, all samples in the dataset are used to conduct experiments on different time step. In Fig. 11 (a), it is clear that as the time step increases, the training FPR on the training dataset gradually decreases. Compared with other models, SRU has the fastest training.
FPR decline, which means that as the time step increases, the SRU model can converge faster to achieve a higher Accuracy and reduce the FPR. As shown in Fig. 11 (b), the model training time increases with time step. Among them, the training time of LSTM and GRU is much more than SRU. It is worth noting that the SimpleRNN has the least training time due to its simplest internal structure. However, SimpleRNN is comfortable to appear gradient vanishing and gradient explosion. It can be summarized from the above results that as time step increases, SRU can obtain the lowest training FPR compared to other models and the training time is much less than LSTM and GRU. Fig. 11 (c) and (d) show the FPR and time of the testing dataset as the time step increases. Under the same time step, the SRU model obtains the lowest FPR, and the testing time is much less than that of LSTM and GRU models.

G. ABLATION STUDY
In order to verify the effectiveness of our proposed two-stage combination method, we conduct a model ablation study. Specifically, to verify that the improvement comes from each component. We remove the component of each stage in turn from the proposed sSAE+BiSRU and compare it with the complete sSAE+BiSRU. For the individual sSAE, we add

| Method     | Trainable parameters | Training time (s) | Testing time (s) |
|------------|----------------------|-------------------|------------------|
| RNN IDS[21]| 22.7k                | 173.8             | 2.5              |
| BLSTM RNN[22]| 42.7k              | 266.1             | 3.5              |
| Proposed   | 6.5k                 | 144.3             | 2.39             |
softmax activation function to the classification test. The BiSRU we proposed is a bidirectional structure based on SRU. In order to verify the superiority of BiSRU, we compare it with a single SRU. We name these models as follows:

- Model w/o sSAE: The model without the sSAE component.
- Model w/o SRU: The model without the SRU component.
- sSAE + SRU: The model replaces BiSRU with an SRU cell.

### TABLE 6. The ablation study results on testing dataset.

| Model                | Accuracy | Precision | Recall | FPR  |
|----------------------|----------|-----------|--------|------|
| Model w/o sSAE       | 0.9684   | 0.9349    | 0.9914 | 0.0501|
| Model w/o SRU        | 0.8898   | 0.7824    | 0.9660 | 0.1538|
| sSAE + SRU           | 0.9923   | 0.9873    | 0.9955 | 0.0102|
| sSAE + BiSRU         | 0.9941   | 0.9904    | 0.9964 | 0.0077|

The ablation study results are shown in Table 6. It can be seen that no matter whether two-stage components are deleted from our proposed model, the last four performance indicators will decline. Among them, the Accuracy of our proposed sSAE+BiSRU model is 0.9941, Precision is 0.9904, Recall is 0.9964, and FPR is 0.0077. Compared with conventional SRU, the performance has been slightly improved. When without the sSAE component, the effects of various performance indicators have become worse. When the BiSRU part was deleted, the result dropped significantly and became the worst, only reaching Accuracy of 0.8898. This shows that the use of BiSRU can significantly improve the Accuracy of abnormal traffic detection, which comes from its efficient automatic extraction of bidirectional features of network traffic. BiSRU can learn the bidirectional structural features of network traffic through the input forward and backward, and achieve accurate detection with less model training time.

### VI. CONCLUSION

In this article, an efficient abnormal traffic detection method based on bidirectional compression features extraction using BiSRU and sSAE was proposed. Our main contribution is to introduce an efficient in-depth features extraction method combined with dimensionality reduction. Firstly, we propose an sSAE to perform feature dimensionality reduction on the high-dimensional network traffic and extract the compressed high-level features. Then a BiSRU with parallel computing is proposed for abnormal traffic detection, which adopts a bidirectional structure to extract the context feature of network traffic better. This leads to a significant speedup for model training compared to LSTM and GRU. The experimental result shows that our method maintains a competitive performance compared to the state-of-the-art methods in terms of accuracy and computational efficiency. In general, this article analyzes the shortcomings of modern abnormal traffic detection technology. Based on deep learning technology,
we propose BiSRU combined with feature dimensionality reduction algorithm for intrusion detection system. The network intrusion behavior can be detected correctly from the traffic data in the intrusion detection system to further ensure the security of network environment.

As future work, we will jointly utilize a swarm intelligence based on a combination of deep and shallow learning, "A review of current security issues in Internet of Things," in Recent Trends and Advances in Wireless and IoT-enabled Networks. Cham, Switzerland: Springer, 2019, pp. 11–23.

J. Niu, Y. Zhang, D. Liu, D. Guo, and Y. Teng, "Abnormal network traffic detection based on transfer component analysis," in Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops), May 2019, pp. 1–6.

C. Yang, "Anomaly network traffic detection algorithm based on information entropy measurement under the cloud computing environment," Cluster Comput., vol. 22, no. S4, pp. 8309–8317, Jul. 2019.

L. Nie, Y. Li, and X. Kong, "Spatio-temporal network traffic estimation and anomaly detection based on convolutional neural network in vehicular ad-hoc networks," IEEE Access, vol. 6, pp. 40168–40176, 2018.

M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," Comput. Secur., vol. 86, pp. 146–177, Sep. 2019.

R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: A survey," 2019, arXiv:1901.03407. [Online]. Available: http://arxiv.org/abs/1901.03407

G. Fernandes, J. J. P. C. Rodrigues, L. F. Carvalho, J. F. Al-Muhtadi, and M. L. Proença, "A comprehensive survey on network anomaly detection," Telecommun. Syst., vol. 70, no. 3, pp. 447–489, Mar. 2019.

A. Sahasarabuddhe, S. Naikade, A. Ramazwamy, B. Sadiwala, and P. Futane, "Survey on intrusion detection system using data mining techniques," Int. Res. J. Eng. Technol., vol. 4, no. 5, pp. 1780–1784, 2017.

R. R. Reddy, Y. Ramadevi, and K. V. N. Sunitha, "Effective discriminant function for intrusion detection using SVM," in Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI), Sep. 2016, pp. 1148–1153.

M. O. Mughal and S. Kim, "Signal classification and jamming detection in wide-band radios using Naive bayes classifier," IEEE Commun. Lett., vol. 22, no. 7, 1398–1401, Jul. 2018.

D. Moon, H. Im, J. Kim, and J. H. Park, "DTB-IDS: An intrusion detection system based on decision tree using behavior analysis for preventing APT attacks," J. Supercomput., vol. 73, no. 7, pp. 2881–2895, Jul. 2017.

M. Panda and M. R. Patra, "Network intrusion detection using naive Bayes," Int. J. Comput. Sci. Netw. Secur., vol. 7, no. 12, pp. 258–263, 2007.

N. Ashraf, W. Ahmad, and R. Ashraf, "A comparative study of data mining algorithms for high detection rate in intrusion detection system," Ann. Emerg. Technol. Comput., vol. 2, no. 1, pp. 49–57, Jan. 2018.

D. Md. Farid, N. Harbi, and M. Zahidur Rahman, "Combining naive bayes and decision tree for adaptive intrusion detection," 2010, arXiv:1005.4496. [Online]. Available: http://arxiv.org/abs/1005.4496

K. Rai, M. S. Devi, and A. Guleria, "Decision tree based algorithm for intrusion detection," Int. J. Adv. Netw. Appl., vol. 7, no. 4, pp. 2828, 2016.

H. M. Anwer, M. Farouk, and A. Abdel-Hamid, "A framework for efficient network anomaly intrusion detection with features selection," in Proc. 9th Int. Conf. Inf. Commun. Syst. (ICICS), Apr. 2018, pp. 157–162.

Q. Tian, J. Li, and H. Liu, "A method for guaranteeing wireless communication based on a combination of deep and shallow learning," IEEE Access, vol. 7, pp. 38688–38695, 2019.

X. Liao, K. Li, X. Zhu, and K. J. R. Liu, "Robust detection of image operator chain with two-stream convolutional neural network," IEEE J. Sel. Topics Signal Process., vol. 14, no. 5, pp. 955–968, Aug. 2020.

C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," IEEE Access, vol. 5, pp. 21954–21961, 2017.

B. Roy and H. Cheng, "A deep learning approach for intrusion detection in Internet of Things using bi-directional long short-term memory recurrent neural network," in Proc. 28th Int. Telecommun. Netw. Appl. Conf. (ITNAC), Nov. 2018, pp. 1–6.

T. Lei, Y. Zhang, S. I. Wang, H. Dai, and Y. Artzi, "Simple recurrent units for highly parallelizable recurrence," 2017, arXiv:1709.02755. [Online]. Available: http://arxiv.org/abs/1709.02755

Y. Bakhti, S. A. Fiezza, W. Hamidouche, and O. Deforges, "DDSA: A defense against adversarial attacks using deep denoising sparse autoencoder," IEEE Access, vol. 7, pp. 160397–160407, 2019.

Y. Fan, C. Zhang, Z. Liu, Z. Qiu, and Y. He, "Cost-sensitive stacked sparse auto-encoder models to detect striped stem borer infestation on rice based on hyperspectral imaging," Knowl.-Based Syst., vol. 168, pp. 49–58, Mar. 2019.

J. Zhang, S.-L. Shen, A. Zhou, and Y.-S. Xu, "Investigation on performance of neural networks using quadratic relative error cost function," IEEE Access, vol. 7, pp. 106662–106652, 2019.

W. Jia, K. Muhammad, S.-H. Wang, and Y.-D. Zhang, "Five-category classification of pathological brain images based on deep stacked sparse autoencoder," Multimedia Tools Appl., vol. 78, no. 4, pp. 4045–4064, Feb. 2019.

Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: LSTM cells and network architectures," Neural Comput., vol. 31, no. 7, pp. 1235–1270, Jul. 2019.

P. Mishra, K. Khurana, S. Gupta, and M. K. Sharma, "VMAnalyzer: Malware semantic analysis using integrated CNN and bi-directional LSTM for detecting VM-level attacks in cloud," in Proc. 12th Int. Conf. Contemp. Comput. (IC3), Aug. 2019, pp. 1–6.

Y. Xie, R. Liang, Z. Li, and L. Zhao, "Attention-based dense LSTM for speech emotion recognition," IEEE Trans. Inf. Syst., vol. E102.D, no. 7, pp. 1426–1429, 2019.

M. Abdel-Nasser and K. Mahmoud, "Accurate photovoltaic power forecasting models using deep LSTM-RNN," Neural Comput. Appl., vol. 31, no. 7, pp. 2727–2740, Jul. 2019.

S. Li, H. Lei, F. Zhou, J. Gardezi, and B. Lei, "Longitudinal and multimodal data learning for Parkinson’s disease diagnosis via stacked sparse autoencoder," in Proc. IEEE 16th Int. Symp. Biomed. Imag. (ISBI), Apr. 2019, pp. 384–387.

L. Liu, Y. Wang, J. Peng, L. Zhang, B. Zhang, and Y. Cao, "Latent relationship guided stacked sparse autoencoder for hyperspectral image classification," IEEE Trans. Geosci. Remote Sens., vol. 58, no. 5, pp. 3711–3725, May 2020.

Y. Gan, Y. Wang, and L. Wei, "Video object forgery detection algorithm based on VGG-11 convolutional neural network," in Proc. Int. Conf. Intell. Comput. Autom. Syst. (ICICAS), Dec. 2019, pp. 575–580.

I. Hammad and K. El-Sankary, "Impact of approximate multipliers on VGG deep learning network," IEEE Access, vol. 6, pp. 60438–60444, 2018.

X. Xu, M. Xie, P. Miao, W. Qu, W. Xiao, H. Zhang, X. Liu, and T.-W. Tong, "Perceptual-aware sketch simplification based on integrated VGG layers," IEEE Trans. Vis. Comput. Graphics, early access, Jul. 24, 2019, doi: 10.1109/TVCG.2019.2935012.

A. S. Garea, D. B. Heras, and F. Argüello, "Caffe CNN-based classification of hyperspectral images on GPU," J. Supercomput., vol. 75, no. 3, pp. 1065–1077, Mar. 2019.

Q. Li, J. Zhong, Y. Tao, L. Li, and X. Miao, "Research of the processing technology for time complex event based on LSTM," Cluster Comput., vol. 22, no. 54, pp. 9571–9579, Jul. 2019.

H. Wu, J. Zhang, and C. Zong, "An empirical exploration of skip connections for sequential tagging," 2016, arXiv:1610.03167. [Online]. Available: http://arxiv.org/abs/1610.03167

S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015, arXiv:1502.03167. [Online]. Available: http://arxiv.org/abs/1502.03167

Q. Qian, R. Jin, J. Yi, L. Zhang, and S. Zhu, "Efficient distance metric learning by adaptive sampling and mini-batch stochastic gradient descent (SGD)," Mach. Learn., vol. 99, no. 3, pp. 353–372, Jan. 2015.

P. Ding et al.: Efficient BiSRU Combined With Feature Dimensionality Reduction
[42] N. Moustafa and J. Slay, “UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set),” in Proc. Mil. Commun. Inf. Syst. Conf. (MilCIS), Nov. 2015, pp. 1–6.

[43] M. Sokolova and G. Lapalme, “A systematic analysis of performance measures for classification tasks,” Inf. Process. Manage., vol. 45, no. 4, pp. 427–437, Jul. 2009.

PENGPENG DING received the B.S. degree from the College of Automation and Electronic Information, Sichuan University of Science and Engineering, China, in 2018. He is currently pursuing the M.S. degree in computer technology with the College of Computer Science and Technology, Shanghai University of Electric Power, China. His research interests include deep learning, information security, and smart grid.

JINGUO LI (Member, IEEE) received the B.S. degree in information security and the Ph.D. degree in computer science and technology from Hunan University, China, in 2007 and 2014, respectively. He is currently an Associate Professor with the College of Computer Science and Technology, Shanghai University of Electric Power. His research interests include information security and privacy, applied cryptography and smart grid.

MI WEN (Member, IEEE) received the M.S. degree in computer science from the University of Electronic Science and Technology of China, in 2005, and the Ph.D. degree in computer science from Shanghai Jiao Tong University, Shanghai, China, in 2008. She is currently an Associate Professor with the College of Computer Science and Technology, Shanghai University of Electric Power. From May 2012 to May 2013, she was a Visiting Scholar with the University of Waterloo, Canada. She serves as an Associate Editor of Peer-to Peer Networking and Applications (Springer). She keeps acting as the TPC member of some flagship conferences such as the IEEE INFOCOM, the IEEE ICC, the IEEE GLOEBECOM, and so on, since 2012. Her research interests include privacy preserving in wireless sensor networks, smart grid, and so on.

LIANGLIANG WANG (Member, IEEE) received the Ph.D. degree from Shanghai Jiao Tong University, in 2016. He is currently an Assistant Professor with the College of Computer Science and Technology, Shanghai University of Electric Power. His research interests include information security, smart grid, and so on.

HONGJIAO LI received the master’s degree from the Huazhong University of Science and Technology, in 2002, and the Ph.D. degree from Shanghai Jiaotong University, in 2008. She is currently an Associate Professor with the College of Computer Science and Technology, Shanghai University of Electric Power. Her main research interests include big data security and privacy, network security, and AI security.

* * *