IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 26, 2019, accepted June 10, 2019, date of publication June 13, 2019, date of current version July 11, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2922692

A Simple Recurrent Unit Model Based Intrusion Detection System With DCGAN

JIN YANG[1], TAO LI[1], GANG LIANG[1], WENBO HE[2], AND YUE ZHAO[3]
[1]Department of Computing and Software, College of Cyber Security, Sichuan University, Chengdu 610017, China
[2]Science and Technology on Communication Security Laboratory, McMaster University, Hamilton, ON L8S 4L8, Canada
[3]Science and Technology on Communication Security Laboratory, Chengdu 610041, China

Corresponding author: Gang Liang (lianggang@scu.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2016YFB0800600, in part by the Natural Science Foundation of China under Grant 61872254 and Grant U1736212, in part by the Fundamental Research Funds for the Central Universities under Grant YJ201727, and in part by the Application Foundation Project of Sichuan Province Science and Technology Department under Grant 2018JY0913.

ABSTRACT Due to the complex and time-varying network environments, traditional methods are difficult to extract accurate features of intrusion behavior from the high-dimensional data samples and process the high-volume of these data efficiently. Even worse, the network intrusion samples are submerged into a large number of normal data packets, which leads to insufficient samples for model training; therefore it is accompanied by high false detection rates. To address the challenge of unbalanced positive and negative learning samples, we propose using deep convolutional generative adversarial networks (DCGAN), which allows features to be extracted directly from the rawdata, and then generates new training-sets by learning from the rawdata. Given the fact that the attack samples are usually intra-dependent time sequence data, we apply long short-term memory (LSTM) to automatically learn the features of network intrusion behaviors. However, it is hard to parallelize the learning/training of the LSTM network, since the LSTM algorithm depends on the result of the previous moment. To remove such dependency and enable intrusion detection in real time, we propose a simple recurrent unit based (SRU)-based model. The proposed model was verified by extensive experiments on the benchmark datasets KDD'99 and NSL-KDD, which effectively identifies normal and abnormal network activities. It achieves 99.73% accuracy on the KDD'99 dataset and 99.62% on the NSL-KDD dataset.

INDEX TERMS Network security, deep learning, intrusion detection system (IDS), simple recurrent unit, deep convolutional generative adversarial networks.

I. INTRODUCTION

The Network Intrusion Detection System (NIDS) is an indispensable component in the security infrastructure in networked computing and communication systems, providing defense against Internet-based attacks, misuse, or negligent practices. Nowadays, with the increasing number of measurements on networked systems, learning based NIDS systems have been developed [1]–[5] to identify various legitimate network activities and the potential hidden threats. However, there are several major challenges in the design and implementation of the learning-based NIDS:

(1) To handle the complexity when dealing with large-scale, high-dimensional data points, traditional network intrusion detection approaches tend to apply dimension reduction, compression, and filtering techniques to remove noise in measurements. Consequently, it is likely to remove hidden but significant information when extracting features for intrusion behaviors. This may cause high false detection rate.

(2) Learning based NIDS usually requires a large number of labeled data samples to obtain accurate features of intrusion behaviors. The performance of the NIDS heavily depends on the quality and size of the labeled training sets. Traditionally, we label the training samples manually, which is labor intensive and error prone, thus it is difficult to generate high-quality large-scale training samples.

(3) A NIDS needs to respond to intrusion behaviors in real time to reduce the loss under an attack. For example, the Overflow attacks are often hidden in the network traffic

The associate editor coordinating the review of this manuscript and approving it for publication was Chenguang Yang.

that can pass through the firewall. If such attacks cannot be detected and blocked in time, the attacker can use it as a springboard to send a large number of aggressive messages to the intranet and leave a back door in the system that has been attacked. This requires a real time intrusion detection system which is implemented with parallelization design and system optimization to achieve time efficiency.

To address the above challenges, we propose a simple recurrent unit model for intrusion detection and discuss the design and implementation of real-time NIDS.

In recent years, deep learning model has been gradually applied to network security applications [6] because of its excellent performance in high dimension data processing, automatic feature extraction, and evolutionary learning ability. Given the fact that the attack samples are usually intra-dependent on time sequence data, we apply LSTM (Long Short-Term Memory) to automatically learn the effective features of time sequence measurements. The Naive LSTM establishes a structure with a gate, or cell, to decide whether to keep or discard the information. In this paper, we adopt the LSTM algorithm to learn the intrusion behavior with intra-sequence correlation and automatically extract the hidden information that cannot be captured by the traditional method.

To alleviate the shortage of correctly labeled training samples, we employ a deep convolutional generative adversarial network (DCGAN) [7] to generate satisfactory training data from original samples. Since DCGAN was originally designed for and more suitable for image processing, we use Mahalanobis Distance to map one-dimensional measurement into two-dimensional data, therefore, we can apply DCGAN to the measured threat samples.

For the sake of efficiency, we apply a simple recurrent unit (SRU) algorithm, so that the proposed NIDS can be easily parallelized on GPU servers. This enables real-time NIDS.

The three contributions of this paper are summarized as below:

1. A multichannel SRU-based network intrusion detection model is proposed. It extracts the features automatically through repeated multi-level learning and outperforms the LSTM algorithm in terms of speeding up the classification process. This model also improves the classification efficiency and detection accuracy against network threat behaviors.

2. A generative adversarial model for generating cyber threat samples is proposed. This method is used to generate new training samples. It solves the common problem faced by traditional intrusion perception methods, such as insufficient and unbalanced samples. It also improves the system detection rate and reduces false alarm rates.

3. A preprocessing method of mapping the network data into a 2-D matrix using Mahalanobis Distance is proposed to ensure that the data can be processed by our model with high efficiency, it offers high-quality data input for the proposed deep learning model.

Our proposed model achieved a detection accuracy of 99.73% on KDD datasets and with 99.62% on NSL-KDD datasets. As contrast, other six popular learning methods (including SVM, Random-Tree, Random Forest, NB Tree, Naive, Bayes, and J48) can only achieve less than 94% of detection accuracy on KDD'99 datasets, and less than 83% of accuracy on NSL-KDD datasets. In addition, the network intrusion detection based on multi-channel SRU has greatly improved the real-time performance of the IDS. It is 10 times faster than the Naive LSTM based method. The proposed method significantly improves the capabilities of network intrusion detection, especially in increasingly complex scenarios with severe cyber threats.

The remainder of this paper is structured as follows. Section II presents relevant background information and existing research. Section III specifies our proposed solution. Section IV discusses our experimental results. Finally, the paper concludes in Section V.

II. RELATED WORKS

The defects of traditional network intrusion detection methods have urged people to re-evaluate the existing cyber security framework and its technologies. There has been increasing interest in recent years in researching intrusion perception technologies equipped with capabilities of high-dimensional data processing, automatic feature extraction, and evolutionary learning. In addition, deep learning has gradually been applied in the field of cyber security. In 2015, at the USENIX conference, Shin et al. [8] proposed a deep learning RNN (Recurrent Neural Network) method to analyze the start and end of functions in Windows binary files as a manner to detect the malware. In the same year, Pascanu et al. [9] applied deep learning RNN to malware detection, and they established a two-layer malware detection system architecture based on dynamic analysis. The first layer is used by RNN to learn the feature representation of API events, and the second layer is a logistic regression classifier, which performs classification using features learned by RNN. In 2016, Huang and Stokes [10] adopted a deep learning DNN method to extract the features of malware that reduced the classification false rate of malware significantly. In the same year, Kolosnjaji et al. [11] constructed a deep learning network with convolutional and circular layers, to detect and analyze the system call of malware dynamically. Bauer et al. [12] implemented a lightweight password cracker by using the deep learning LSTM algorithm which increased the speed of password cracking. In 2017, Rhodea et al. [13] applied a deep learning RNN method to predict malicious behavior in the early stage of malware execution and reached an accuracy of 98%. This greatly exceeds the highest accuracy of 90% that was achieved by other traditional machine learning algorithms. In addition, Putchala [14] applied the deep learning GRU method to intrusion detection in the field of Internet of Things (IoT). Also, Aygun et al. [15] adopted the deep learning random noise reduction self-encoder to detect intrusions and reached the accuracy rate of 88.65%. Godefroid et al. [16]

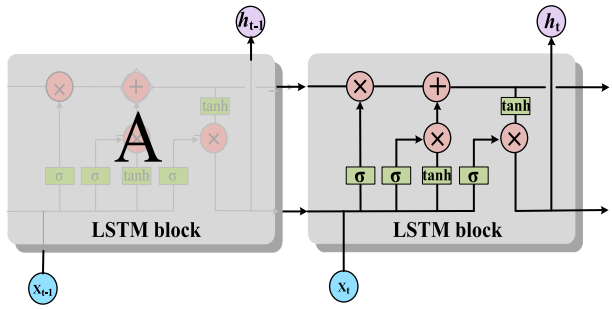


FIGURE 1. The structure of LSTM.

applied deep learning to a vulnerability mining program and he adopted the seq2seq architecture to generate fuzzing test cases for files vulnerability mining.

Long Short-Term Memory (LSTM) was proposed by Hochreiter and Schmidhuber [17] as a special type of RNN. In this model, the normal neurons (i.e., the unit that applies the sigmoidal activation function to the linear combination of its input) are replaced by the memory cells to avoid the long-term dependency issue. Naive LSTM establishes a structure known as a gate, or cell, to decide whether to keep or discard the information. It outperforms ordinary RNNs in terms of storing and accessing the historical sequences, and it allows the information to be kept over a long period of time by selectively storing or discarding the information [17]. LSTM has been widely applied to solve various problems in a wide variety of domains such as handwriting recognition, semantic recognition, natural language understanding, disease prediction, and weather forecasting. The structure of the LSTM is shown in Figure 1. The information is processed, discarded, or controlled by operations of sigmoid function and a point-by-point multiplication.

Researchers have constantly adjusted and optimized LSTM. In 2000, Gers and Schmidhuber [18] proposed a neural network with peephole connections as a variation of LSTM. This means the gate connection layer can receive the state of the cell. In 2009, Martin and Eyben [19] proposed a new technique for robust keyword spotting that uses bidirectional long short-term memory (BLSTM) recurrent neural nets to incorporate contextual information in speech decoding. In 2011, Wollmer and Blaschke [20] proposed a novel technique for online detection of driver's distraction based on a variation of LSTM, modeling the long-range temporal context of driving and head tracking data. In 2013, Derek presented [21] a new technique Recurrent Neural Collective Classification-RNCC which avoided arbitrary summarization, and learned to leverage larger relational neighborhoods around each entity. In 2014, Cho et al. [22] proposed the Gated Recurrent Unit (GRU) model, an important variation of LSTM. It combines the forget gate and input gate into a single update gate that consists of the cell state and hidden state. In 2014, Yao K et al. [23] proposed the Depth Gated RNNs model, and Koutnik J. et al. [24] proposed the Clockwork RNNs model which aims to solve long-term dependency problems.

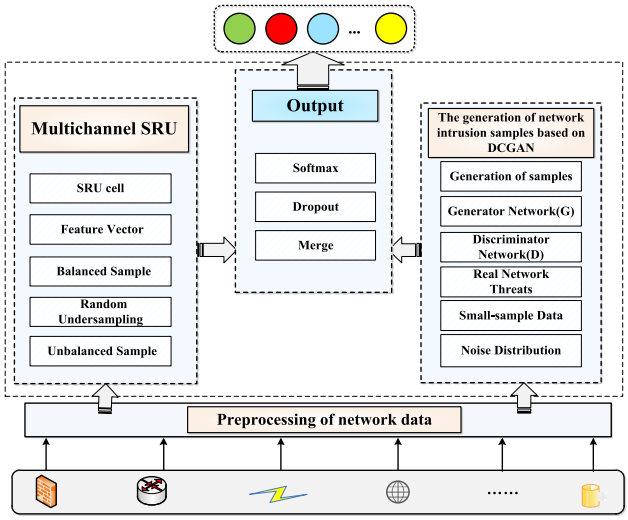


FIGURE 2. The structure of our model.

In brief, Deep learning [25]–[27] has drawn wide attention from universities, research institutes, and industries. It is of paramount importance to conduct research on network intrusion detection based on deep learning for developing the field of cyber security.

III. SRU-DCGAN MODEL

The framework of SRU-DCGAN model is shown in Figure 2. The complete architecture decomposes to three main subcomponents: the Multichannel SRU-based network intrusion detection component (Subsection A), the generation of network intrusion training-samples based on DCGAN component (Subsection B), and the preprocessing of network data component (Subsection C) as following. Moreover, it should be noted that the *output* component of the whole model is introduced in Subsection A.

A. THE MULTICHANNEL SRU-BASED NETWORK INTRUSION DETECTION

Due to the lack of automatic feature extraction capability from complex and high dimensional data, the performance of existing approaches is not satisfactory. Particularly, judging an intrusion behavior depends on a sequence of measurement samples, such as the counterfeit legitimate user and the behavior of unauthorized use of system resources. This implies that a deep neural network with memory units such as LSTM is well-suited for intrusion detection. Because a LSTM network usually has a deep structure and each layer contains many units, it requires significant time to train the network.

To accelerate the training, we prefer to parallel the training procedure using the GPU. Unfortunately, naive LSTM does not support the implementation of parallelization, because in the calculation, the model state at time t depends on its previous state at $t-1$. To enable the parallelization, we introduce a Simple Recurrent Unit (SRU) structure as a variation of LSTM, which accelerates the training speed 10 times than the optimized LSTM [28].

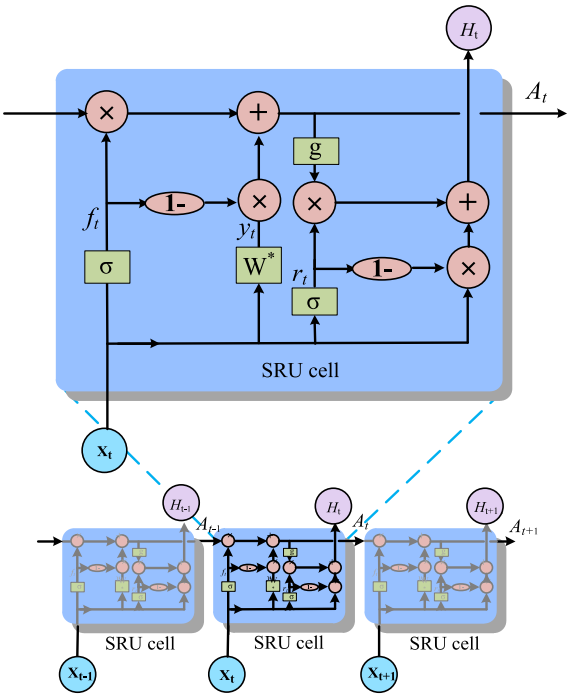


FIGURE 3. The structure of SRU cell. where, x_t represents the input at time t; W and b represent weight and bias, respectively;f_t represents the forget gate at time t;r_t represents the reset gate ate time t; A_t and H_t represent the state and the final output at time t, respectively; σ and g represent Sigmoid and the Activation function (tanh or relu); $\odot$ represents the operation of the matrix corresponding to the elements.

Figure 3 is the structure of the SRU cell. The SRU model uses a complete drop connection, which removes the dependence of the input H_{t-1}. The following is the model of SRU:

$$f_t = \sigma(W_f x_t + b_f) \tag{1}$$
$$\mathrm{r_t} = \sigma(\mathrm{W_r x_t} + \mathrm{b_r}) \tag{2}$$
$$\mathrm{y_t} = \mathrm{W^* x_t} \tag{3}$$
$$\mathrm{A_t} = \mathrm{f_t} \odot \mathrm{A_{t-1}} + (1 - \mathrm{f_t}) \odot \mathrm{y_t} \tag{4}$$
$$\mathrm{H_t} = \mathrm{r_t} \odot \mathrm{g(A_t)} + (1 - \mathrm{r_t}) \odot \mathrm{x_t} \tag{5}$$

where W_f, W_r^* and W^* are parameter matrices and b_f, b_r are parameter vectors to be learnt during training.

The complete architecture contains two components: *light recurrence* (Equation 1, 3 and 4) and *highway network* (Equation 2 and 5). From Equation 1 to 3, we can see that the dependency of H_{t-1} has been removed, and formulas Equation 4, 5 can be implemented quickly and concisely, because the output is no longer dependent on input from the H_{t-1}.

In our implementation of SRU-based NIDS, we adopt a *Highway Network component* between recurrent layers to alleviate the vanishing gradient or exploding gradient problem. In addition, we use Adamax optimizer and variational dropout during training. The connection between the Gate of t is removed to **accelerate** the recurrence calculation. Therefore, we enable parallel training for the proposed NIDS.

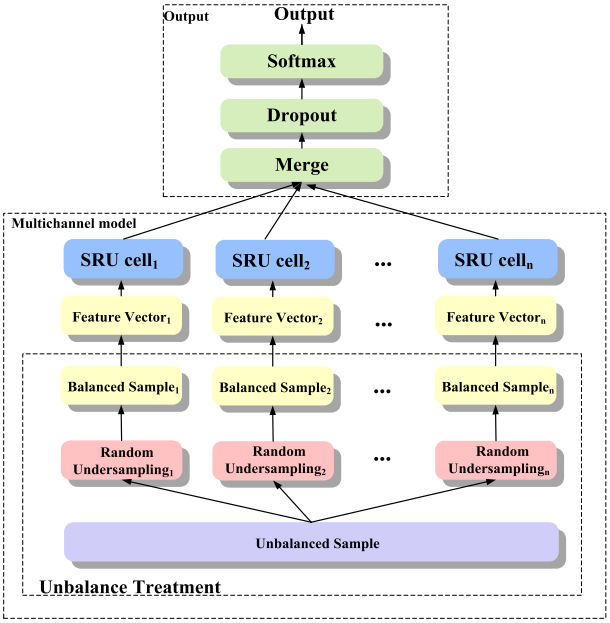


FIGURE 4. The structure of the multichannel SRU Model.

The performance of deep neural networks depends on the quality and size of the labeled samples. It is usually hard to capture positive samples (i.e., samples in attack scenarios), because positive threat samples are usually submerged into a large amount of normal data packets in reality. In this paper, two approaches are used to address the problem with unbalanced samples: One is to generate more specific samples and supplement them in the training set. This approach will be discussed in the later section. Another is to adopt the multi-channel SRU model, which collects several balanced sample sets by sampling the unbalanced data multiple times and train the SRU model using multiple balanced samples.

The framework of the multi-channel SRU neural network classifier is illustrated in Figure 4. Firstly, we under-sampled the training sample n times for each category. The number of under-sampled samples is the number of the minimum training samples every time. In the n groups of balanced samples obtained, the network threat categories of each group of corresponding location samples are the same. The n groups of different samples can be regarded as n groups of different feature representations corresponding to m kinds of Network Threat categories. We use all of the eigenvectors of the n training samples as the input of the model simultaneously, and each group of input is used to train every single-channel SRU cell. As shown in Figure 4, the SRU $\mathrm{cell_n}$ was learned from the n^{th} set of training samples. The output of the model can represent the input features in a better way. In our experiment, we let $n = 4$, to ensure that most of the input features in the labeled samples can be selected.

The output features of the above-mentioned n sets of SRU models are merged at the Merge layer. The Dropout layer takes the output of the Merge layer as the input, and

its function is the same as that in the single-channel SRU neural network. The back propagation method is adopted to update the network parameters. The softmax output layer is the last layer of the network. During model training, it requires the mean square error of the predicted and actual values to be minimized. Given a set of training samples, $X = \{x^{(1)}, x^{(2)}, \ldots, x^{(i)}, \ldots, x^{(m)}\}$, the true tags are $\{y = y^{(1)}, y^{(2)}, \ldots, y^{(i)}, \ldots, y^{(m)}\}$, then $f(x, \Theta)$ gives the prediction value of sample x, and the loss function can be defined as shown in Equation (6):

$$L(X, y) = \frac{1}{2m} \sum_{i=1}^{m} \left\| f\left(x^{(i)}, \Theta\right) - y^{(i)} \right\|^2 \quad (6)$$

The back-propagation algorithm [29], based on the gradient descent method, is applied to learn the model parameters during the training process, and eventually determine the category of attacks.

B. THE GENERATION OF NETWORK INTRUSION TRAINING-SAMPLES BASED ON DCGAN

In addition to multi-channel SRU, we exploit using a generative adversarial network model, DCGAN to generate adequate and balanced data samples. We build the dynamic equilibrium (Nash equilibrium) in the high dimensional and non-convex continuous game by constructing the competitive adversarial relationship. The DCGAN model significantly reduces the false positive detection rate.

In 2016, Goodfellow [30] proposed the Generative Adversarial Networks (GAN), aimed at solving the problem of generating new samples from the training samples. In the GAN model, the generator G continuously generates fake samples close to the real samples to confuse the discriminator, and the discriminator D filters out the fake samples generated by the generator G. In such a process, the discriminator D is trained and continuously improving the discriminating ability, so the generated samples which are close to real-world measurements are left and involved in the training set. There are a few mainstream generation models, including DCGAN (Deep Convolutional Generative Adversarial Networks)[7], WGAN(Wasserstein GAN), and Least Squares GAN (LSGAN), etc. Among those DCGAN is the most suitable to the network intrusion detection because it has relatively less training time and higher accuracy. With DCGAN we obtain adequate training samples for NIDSes, therefore the threat detection rate can be significantly improved.

We utilize the idea of DCGAN to generate new samples through Generative-Adversarial process, which achieves the dynamic equilibrium in the high-dimensional and non-convex continuous adversary process (NASH equilibrium). The new samples can be generated by learning from existing attack samples and keep as much information as possible that appeared in the raw sample data. The outstanding features of DCGAN can extract the correlations of data in high-dimensionality instantly, without labeling the target categories. Moreover, it can generate training samples by automatically capturing the distribution of sample data.

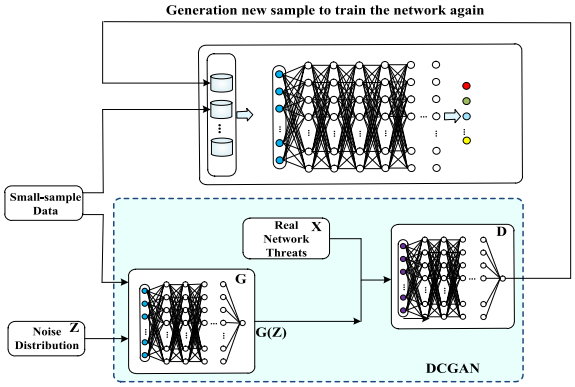


FIGURE 5. Sample generation and discrimination network model based on DCGAN.

The diagram of sample Generation and the Discrimination model is shown in Figure 5.

DCGAN is formalized as shown in Equation (7):

$$\mathrm{V}(\mathrm{D}, \mathrm{G}) = \mathrm{E}_{x \sim p_{data}(x)} [logD(x)] + \mathrm{E}_{z \sim p_z(z)} \left[\log(1 - D(G(z)))\right] \quad (7)$$

where $x \sim p_{data}(x)$ shows that x is from the real distribution, and $z \sim p_z(z)$ shows that z is from the simulated distribution. G indicates the generation model, and D represents the classification model.

1) GENERATOR NETWORK

The generator continuously generates data, aiming to generate the samples that can "cheat" the discriminator and make it appear real. We construct generator G with the CNN, and replace the pooling layer with the fractional srided convolutions. Then, we remove the fully connected layer, apply the ReLU for all layers except the output layer, and use tanh at the output layer. We also use batchnorm to solve the poor initialization problem and propagate the gradient to each layer. G is optimized to minimize V(D, G), that is $\min_G(\mathrm{V}(\mathrm{D}, \mathrm{G}))$.

2) DISCRIMINATOR NETWORK

The discriminator continually receives data and judges whether the data comes from real samples or the generated ones. The discriminator is also constructed by using CNN. And the CNN layer contains pooling layer, but does not include a fully connected layer. Batchnorm is used to propagate the gradient to each layer to avoid the generator converging all samples to the same point. LeakyReLU is used at all layers. During the process of training the discriminator, the generator needs to be paused, i.e., stopping the training of the generator (no backpropagation is performed). D is optimized to maximize V(D,G), i.e., $\max_D(\mathrm{V}(\mathrm{D}, \mathrm{G}))$.

3) GENERATION OF SAMPLES

There are two phases involved in the process of sample generation: One phase continuously optimizes G to make the samples confused with D as much as possible. In the other

phase, D is optimized to make it recognize the "fake" samples as much as possible. Next, as D can no longer distinguish whether the sample is a real threat or a fake sample generated by G, and G can no longer generate a different sample, comes the moment when these two sides have achieved the Nash equilibrium. Then, it is deemed that the new samples generated by G have the same distribution of the raw samples. In this condition, the adversarial between the generator and the discriminator can be terminated and the DCGAN training is finished. The generated samples are inserted into the sample base as training samples.

4) THE WHOLE PROCESS OF GENERATING DATA

The detail of data reproduced by the model proposed in this paper is listed below:

- *Step 1:* select benign and malicious examples from the KDD'99 randomly;
- *Step 2:* Obtain G(Z) by generative network G and obtain an example Z by mask;
- *Step 3:* Label x or z according to the discrimination result of T;
- *Step 4:* Calculate loss according to the D's discrimination result;
- *Step 5:* Update weights of D;
- *Step 6:* Update weights of G;
- *Step 7:* Loop step 2 to step 6 if the authentication module is failed.

C. PREPROCESSING OF NETWORK DATA

In order to speed up the training speed and reduce the processing time of the system, we propose a data preprocessing method suitable for deep learning processing.

By collecting the network flow data (firewalls, vulnerability scanning systems, terminal security management systems, security management platforms, and the security operation centers) in real-time, some of the essential features are selected. These features include the source IP address, destination IP address, source port number, destination port number, protocol type, domain name, total number of data packets, the interval of packet arrival, stream duration, number of concurrent streams, number of reconnections, the length of the first packet, the total number of bytes, the average number of bytes per packet, the number of empty packets, the ratio of the number of incoming and outgoing packets, the average interval of packets arrival, average packet length, average number of bits per second, average number of packets per second, etc. However, in real scenarios, the optional features are much more numerous than the above-mentioned basic ones.

To satisfy the training requirements of deep learning model, a network data mapping method is proposed during the stage of data preprocessing after collecting the data. In this paper, n network flow eigenvectors are defined as $X = [x_1, x_2, \ldots, x_j, \ldots, x_n]$ (e.g., if the KDD'99 or NSL-KDD dataset is used for training, and the 41 attributes of raw data are regarded as the essential features, then n = 41. Different values of n correspond to different data sources.).

$$\text{Then } X = \begin{bmatrix} w_1^1 & w_1^2 & \cdots & w_1^n \\ w_2^1 & w_2^2 & \cdots & w_2^n \\ \vdots & \vdots & \vdots & \vdots \\ w_m^1 & w_m^2 & \cdots & w_m^n \end{bmatrix}$$

where w_k^j is the value of the k^{th} feature of the j^{th} network flow eigenvector, $j = 1, 2, \ldots, m$; $k = 1, 2, \ldots, m$. Next, the x_j is converted to a matrix $x_j^{'}$ with m rows and m columns, to take advantage of the correlation between different features, as shown in Equation (8).

$$\begin{aligned} x_j^{'} = x_j^T I &= \left[w_1^j, w_2^j, \ldots, w_m^j\right] \cdot \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}_{m\times m} \\ &= \begin{bmatrix} w_1^j & 0 & \cdots & 0 \\ 0 & w_2^j & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & w_m^j \end{bmatrix}_{m\times m} \end{aligned} \tag{8}$$

The diagonal elements of $x_j^{'}$ represent the values of the m-dimensional feature, as shown in Equation (9), where each column of $x_j^{'}$ is demonstrated as the m-dimensional vector W_p^j.

$$W_p^j = \begin{bmatrix} \alpha_{p,1}^j \\ \alpha_{p,2}^j \\ \cdots \\ \alpha_{p,m}^j \end{bmatrix}, \quad \text{where } \alpha_{p,t}^j = \begin{cases} w_p^j, & t = p \\ 0, & t \neq p \end{cases} \tag{9}$$

So, $x_j^{'} = \left[W_1^j, W_2^j, \ldots, W_m^j\right]$

To define the correlation between different features of the network flow eigenvectors, the Mahalanobis Distance is adopted in this paper as shown in Equation (10):

$$\beta_{p,k}^j = \begin{cases} \sqrt{(W_p^j - W_k^j)^T S^{-1} (W_p^j - W_k^j)}, & t \neq p \\ 0, & t = p \end{cases} \tag{10}$$

The advantage of using the Mahalanobis distance is that it is not related to dimension, and it allows a balance between the probability of the two categories by the introduced covariance parameters. This parameter illustrates the degree of point density and can eliminate the interference caused by the correlation between variables. As a result, the j^{th} flow data is converted into a symmetric Hallow matrix E with m rows and m columns, as shown in Equation (11), where

$$E_{x_j} = \begin{vmatrix} \beta_{1,1}^j & \beta_{1,2}^j & \cdots & \beta_{1,m}^j \\ \beta_{2,1}^j & \beta_{2,2}^j & \cdots & \beta_{2,m}^j \\ \vdots & \vdots & \vdots & \vdots \\ \beta_{m,k}^j & \beta_{m,k}^j & \cdots & \beta_{m,m}^j \end{vmatrix} \tag{11}$$

Whether it is a normal or abnormal network flow, the Mahalanobis distance is used to map to E, which can be used in the next step as the input of the SRU-based intrusion detection model.

IV. EXPERIMENTAL RESULTS

A. DATASET AND EXPERIMENTAL ENVIRONMENT

The experiment involves 3 datasets, the KDD'99 [31] dataset, the NSL-KDD [32] dataset and CICDS2017 dataset [33].

The KDD'99 dataset is guided by the ACM Special Interest Group on Data Mining and Knowledge Discovery, and counts as a tool to collect data in diverse types publicly. It is the leading data mining competition in the world. This dataset in this study involves numerous and various intrusions that are simulated under the condition of a military network, and it is virtually a dataset to evaluate and benchmark IDS tools [34]. Each record in the KDD'99 dataset consists of 41 characteristics and is labeled with either normal or a particular kind of attack. The raw data collected by these TCP-Dump are divided into two parts: training data for 7 weeks which probably contains more than 5,000,000 network connection records, and the remaining 2 weeks of test data probably containing 2,000,000 network connection records. There are two kinds of datasets in KDD'99, the original full dataset, and a section containing 10% of the original dataset samples.

NSL-KDD is a new data set which is to solve some of the inherent problems of the KDD'99 data set. NSL-KDD counts as a dataset proposed to reckon with some underlying problems existing in the KDD'99 dataset. The most significant weakness in the KDD'99 dataset refers to the considerable amount of residual records [32]. To reckon with the foregoing problems, a new dataset NSL-KDD was designed by Tavallaee et al. [35], not involving any residual records in the training set and the test set. In this regard, the classifiers shall not have biased towards records being more frequent, which increases the efficiency for attaining an accurate evaluation of diverse learning technologies. Consequently, the evaluation results of diverse research work shall turn out to be comparable and consistent [36]. This paper adopts the first 20% of the records in KDDTrain+ as the set of training, and adopts the KDDTest-21, KDDTest+ and KDDTest as sets of test.

Although the above-mentioned two data sets are the benchmarks in the research of Intrusion Detection techniques. These two data sets suffer several weaknesses including their age, highly skewed targets, non-stationarity between training and test datasets, pattern redundancy, and irrelevant features [35]. So, a state of art dataset named CICIDS2017 which is provided by Canadian Institute of Cybersecurity was introduced in this paper. This new dataset not only contains up-to-date network attacks but also fulfills all the criteria of real-world attacks [37]. The CICIDS2017 contains data capturing period started at 9 a.m., Monday, July 3, 2017 and ended at 5 p.m. on Friday July 7, 2017, for a total of 5 days. The data of Monday is the normal and only includes the benign traffic. The data of Thursday working hour afternoon and Friday is fit for binary classification. And, the data of Tuesday, Wednesday and Thursday morning is well suited for designing multiclass detection model.

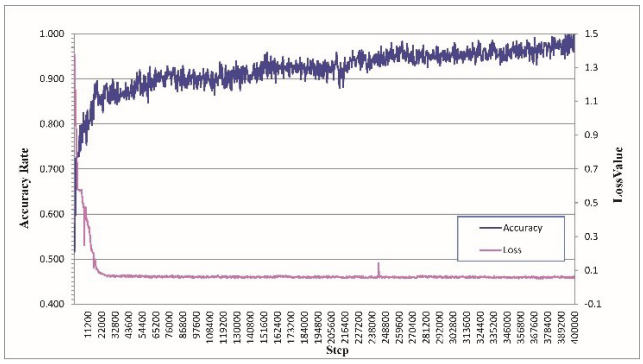


FIGURE 6. The accuracy and loss of SRU-DCGAN model.

The specifications of the hosts adopted in the experiments are Core i7 6700k 4.2 GHz CPU, 64 GB RAM, Ubuntu 16.04, NVidia GTX 1080 Ti Device Driver*10, CUDA8.0, CUDNN 5.1, ANACONDA3, and Tensorflow 2.0 were adopted in each of the hosts.

B. PERFORMANCE OF SRU-DCGAN MODEL IN KDD'99 DATASET

The first set of tests was designed to ascertain the accuracy of proposed model, and the final result showed the recognition accuracy is 99.73% (see Figure 6). The Hyper-parameter settings were as follows: hidden units = [30, 70, 30, 30], steps = 400000, batch-size = 20000, epoch = 500, Learning Rate $\eta = 0.001$.

As shown in Figure 6, the detection accuracy increased from 25% to about 65% at the initial stages of the 1,000 steps, which is related to the initial stage of learning. The detection rate was also relatively low in the initial stages. Since the learning had just started, the accuracy was quite low, but it increased dramatically. During this period, the value of loss drastically dropped from 1 to 0.5. This is because the multi-channel parallel SRU could accelerate processing and improve learning efficiency. From step 1,000 to 10,000, the learning rate steadily increased, and the detection rate increased from 65% to 95%. Meanwhile, the training process gradually stabilized as the value of loss decreased from 0.5 to about 0.08. From step 10,000 to 38,000, the accuracy rate increased from 95% to approximately 99.7%, while the detection rate increased quite slowly. Moreover, the value of loss remained at around 0.05, indicating that the learning process had been completed. Eventually, we terminated the training at step 400,000.

Figure 7 illustrates the comparison results between the proposed model as well as the classical algorithm with dataset KDD'99. Literature[1] describes the performance adopted by the six learning methods, including SVM, Random-Tree, Random Forest, NB Tree, Naive, Bayes, and J48. It was found that SRU-DCGAN model had the highest accuracy than any of the six methods.

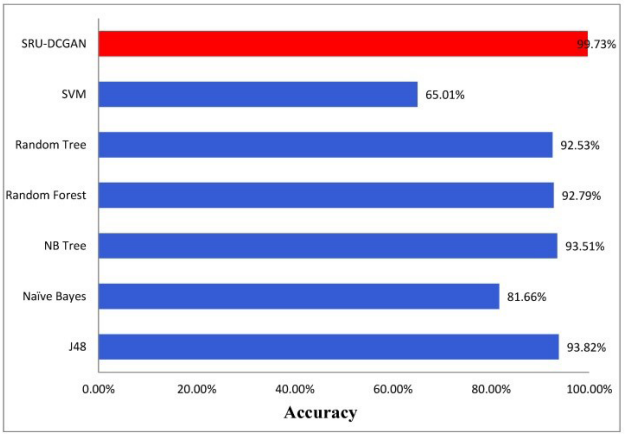


FIGURE 7. Comparing experiment with SRU-DCGAN model and the classical algorithm.

TABLE 1. Accuracy of SRU-DCGAN model and shallow model.

Model Type	2-Class			4-Class		
	Accuracy (%)	*DR (%)*	*FAR (%)*	*Accuracy (%)*	*DR (%)*	*FAR (%)*
SRU-DCGAN	99.73	99.79	0.56	99.37	98.92	2.13
Shallow Model	95.22	97.50	6.57	96.75	93.66	4.97

Furthermore, classification accuracy and other indicators were applied to demonstrate the effectiveness of SRU-DCGAN model and other models in the classification of network intrusions during the process of evaluation. The detection rates in Table 1 also illustrate that SRU-DCGAN model was superior to traditional machine learning for binary classifications and multiclass classifications. The accuracy improvement of the multi-channel model that we proposed is due to the sharing of collaborative learning parameters, which avoids overfitting of the local parameters. As a result, the accuracy benefits from this.

As shown in Table 1, the Accuracy (99.73%) is the closeness of a measured value to a standard or known value. And the FAR (False Alarm Rate) of 0.56% of the SRU-DCGAN model was much lower than the machine learning model (6.57%). As shown in Table 2, SRU-DCGAN model was superior to general machine learning models in terms of performance for each type of intrusion. In addition, the DR is the Detection Rate.

For the binary classification, the DR Rate of the SRU-DCGAN model was 99.34%, which is superior to the traditional model with a DR Rate of 97.50%. Similarly, the average DR Rate of the deep model was 97.7% in multi-class classification, which is also higher than the traditional model at 93.66%.

C. FURTHER EXPERIMENTS ON NSL-KDD DATASET

Figure 8 illustrates the comparison results between the proposed model as well as the classical algorithm with

TABLE 2. Performance of 2-class and 4-class.

Model type	2-Class			4-Class		
	Class	*Accuracy (%)*	*DR (%)*	*Class*	*Accuracy (%)*	*DR (%)*
SRU-DCGAN	Normal	99.86	99.26	Normal	99.57	97.62
	Attack	99.73	99.34	DoS	97.31	99.6
				Probe	98.97	99.3
				R2LU2R	83.5	94.3
Shallow Model	Normal	97.95	93.43	Normal	99.35	95
	Attack	92.1	97.50	DoS	96.55	99
				Probe	87.44	99.48
				R2LU2R	42	82.49

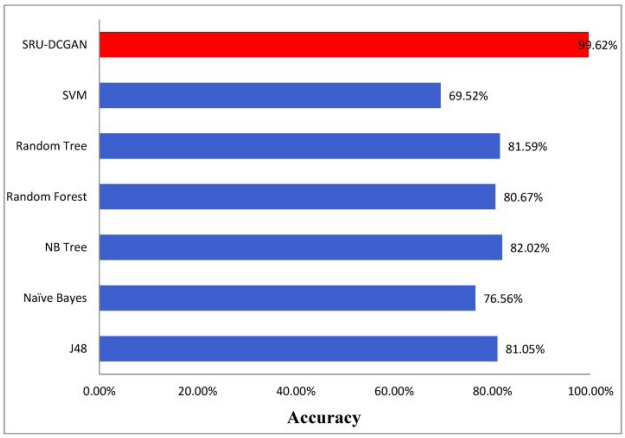


FIGURE 8. Comparing experiment with SRU-DCGAN model and the classical algorithm based on NSL-KDD dataset.

TABLE 3. Accuracy of SRU-DCGAN model and shallow model on NSL-KDD dataset.

Model Type	2-Class			4-Class		
	Accuracy (%)	*DR (%)*	*FAR (%)*	*Accuracy (%)*	*DR (%)*	*FAR (%)*
SRU-DCGAN	99.62	99.68	0.79	98.21	99.33	3.61
Shallow Model	94.13	96.78	8.37	94.13	92.51	6.32

dataset NSL-KDD. The classification accuracy and other indicators were applied to demonstrate the effectiveness of SRU-DCGAN model and other models in the classification of network intrusions during the process of evaluation on the NSL-KDD dataset, as shown in Table 3 and 4. And the SRU-DCGAN model also achieved a good performance on the NSL-KDD.

D. EXPERIMENTS ON CICIDS2017 DATASET

Table 5 demonstrates the experimental result of the proposed model in this paper. As shown in the Table 5, our model can detect most of intrusion correctly, which proves the validity of the model proposed in this paper. F-Measure (F1) is a

TABLE 4. Performance of 2-class and 4-class on NSL-KDD dataset.

Model type	2-Class			4-Class		
	Class	*Accuracy (%)*	*DR (%)*	*Class*	*Accuracy (%)*	*DR (%)*
SRU-DCGAN	Normal	99.12	96.26	Normal	98.24	96.72
	Attack	98.26	98.61	DoS	96.72	98.13
				Probe	94.27	98.23
				R2LU2R	82.43	93.26
Shallow Model	Normal	97.95	92.16	Normal	96.2	93.2
	Attack	89.15	94.20	DoS	94.85	97.6
				Probe	82.45	98.32
				R2LU2R	41	81.32

TABLE 5. The results of different evaluation metrics on CICIDS2017 dataset.

	Accuracy (%)	DR (%)	F1 (%)
Normal	99.3	92.2	95.6
Web Attack	65.1	89.2	75.3
PortScan	86.2	99.4	92.3
Web Attack	65.1	89.2	75.3
DoS	89.1	89.1	89.1
DDOS	84.5	98.6	91.0
BOT	71.1	79.8	75.2

TABLE 6. The comparative experiments between the Original kdd'99 dataset and the regenerated Kdd'99 dataset.

	Accuracy (%)	DR (%)	F1 (%)
Regenerated Kdd'99 dataset	99.73	99.79	99.76
Original Kdd'99 dataset	96.17	89.12	92.51

harmonic combination of the precision and recall into a single measure.

E. THE COMPARATIVE EXPERIMENTS BETWEEN THE ORIGINAL KDD'99 DATASET AND THE REGENERATED KDD'99 DATASET

In order to prove the validation of regenerated samples by the SRU-DCGAN model, we make comparative experiments between the original kdd'99 dataset and the regenerated Kdd'99 dataset (by SRU-DCGAN model), and the experiment results are listed below in Table 6.

From Table 6, we can see the Accuracy is improved in the regenerated dataset which proves the effectiveness of the method proposed in our model.

F. EXPERIMENTS ON DIFFERENT HARDWARE

The training time is related to the GPU performance, the number of GPUs, the dataset, and the number of SRU channels.

TABLE 7. The training time of SRU-DCGAN model on different hardware's.

Hardware	Training Time/Seconds	
Intel core i7- Core i7 6700k	Serial	135.350276
	Parallel	93.3454011
Nvidia GeForce GTX 1080Ti	63.4059697	
Intel core i7- Core i7 6700k + Nvidia GeForce GTX 1080Ti	74.1890722	

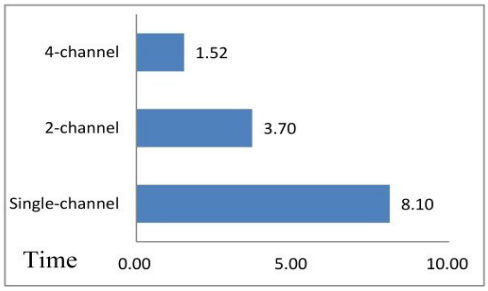


FIGURE 9. Comparing experiment with single-channel and multichannel SRU model.

After analyzing the accuracies, we measured the training time of our model on different platforms. Table 6 shows the training time needed in our model.

Firstly, we just use the Intel Core i7-6700k which is 4.2GHZ, quad core processors with 8 threads. From the table, we can see the training time of CPU in parallel mode is very fast when comparing to CPU in serial mode. Executed serially is very high when compared to parallel. The training time of CPU in parallel mode achieved nearly 4x faster in our model when compared to CPU in serial mode. Then we used GPU NVidia GTX 1080 Ti to train the model. The training time of the GPU was faster than that in CPU only execution. Finally, we combined CPU + GPU to train the model proposed in this paper. This combination did not gain much performance benefits than GPU only execution. The factor that performance of the combination mode is not achieved our expected is duo to the data type we used (Table 7).

G. EXPERIMENTS OF SINGLE-CHANNEL AND MULTICHANNEL SRU MODEL

The impact exerted by channel number on the processing time of the proposed model was ascertained in this experiment. Figure 9 shows the processing time of the Single-channel SRU, 2-channel SRU, and 4-channel SRU. By comparison, it indicates that 4-channel was almost 5 times faster than the Single-channel SRU model.

H. RESULTS AND ANALYSIS

The experimental result showed that our proposed model is superior to the traditional methods, primarily for two reasons

that are being considered. On the one hand, the proposed model is based on a variation of LSTM - the SRU method. SRU has the same training efficiency as CNN and is 10 times faster than the optimized LSTM. Moreover, unlike the LSTM algorithm, SRU is not dependent on the calculation performed at time ht-1 anymore. This design can realize the parallel execution of programs and is superior to LSTM. At the same time, the SRU method is more suitable than CNN to deal with issues related to network intrusion detection. The proposed multi-channel parallel SRU can accelerate processing under the premise of ensuring the detection efficiency. On the other hand, the existing methods of cyber-threat detection are increasingly ineffective. The core intrusion behaviors are often difficult to identify and are submerged in many normal data packets. Fewer intrusions make the threat detection samples relatively small, which can easily lead to inadequate training of models and short learning processes. Even worse, it is impossible to train the model when the training samples are insufficient. Normally, the collected attack samples have a serious imbalance, and models trained with these unbalanced datasets tend to fit the sample categories that have the largest amount, which causes serious bias in the model. Although many traditional methods have high overall detection rates on unbalanced datasets such as KDD'99, they can detect certain types of intrusions with a high detection rate, while giving a very low detection rate for other types of intrusions. The SRU-DCGAN model, however, shows good test results in both KDD'99 and NSL-KDD datasets with high detection rates.

V. CONCLUSION

This paper proposes a network intrusion detection model based on SRU and DCGAN. Firstly, a network data preprocessing method is established for complex, multidimensional cyber threats, which is suitable for deep learning. It can provide high-quality data input for the proposed deep learning model. In addition, an SRU-based intrusion detection model is proposed, which extracts features automatically through repeated multi-level learning by taking advantage of the outstanding features of deep learning. It outperformed the traditional LSTM algorithms in terms of processing efficiency and classification accuracy, and improved the classification efficiency and accuracy of cyber intrusion behaviors. Finally, a method for generating samples of network threats based on generative adversarial is presented which extracts features directly from the raw data and generates new samples from the training samples. It solves the common problems of inadequate training from unbalanced samples and small samples that are faced by the traditional threat perception methods, improves the system detection rate, and reduces the false positive rate.

ACKNOWLEDGMENT

The authors would like to convey their grateful appreciation to the corresponding author of this paper, Gang Liang, who has offered advice with huge values in all stages when writing this essay to them.

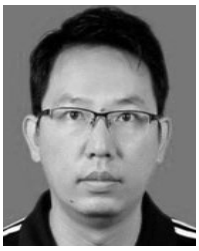
REFERENCES

[1] O. Gupta and R. Raskar, "Distributed learning of deep neural network over multiple agents," *J. Netw. Comput. Appl.*, vol. 116, pp. 1–8, Aug. 2018.
[2] P. K. Sharma, S. Singh, and J. H. Park, "OpCloudSec: Open cloud software defined wireless network security for the Internet of Things," *Comput. Commun.*, vol. 122, pp. 1–8, Jun. 2018.
[3] F. Salo, A. B. Nassif, and A. Essex, "Dimensionality reduction with IG-PCA and ensemble classifier for network intrusion detection," *Comput. Netw.*, vol. 148, pp. 164–175, Jan. 2019.
[4] B. Selvakumar and K. Muneeswaran, "Firefly algorithm based feature selection for network intrusion detection," *Comput. Secur.*, vol. 81, pp. 148–155, Mar. 2019.
[5] Z. Cui, F. Xue, X. Cai, Y. Cao, G.-G. Wang, and J. Chen, "Detection of malicious code variants based on deep learning," *IEEE Trans. Ind. Informat.*, vol. 14, no. 7, pp. 3187–3196, Jul. 2018.
[6] M. R. G. Raman, N. Somu, K. Kirthivasan, R. Liscano, and V. S. S. Sriram, "An efficient intrusion detection system based on hypergraph—Genetic algorithm for parameter optimization and feature selection in support vector machine," *Knowl.-Based Syst.*, vol. 134, pp. 1–12, Oct. 2017.
[7] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *Proc. Int. Conf. Learn. Represent.*, 2016, pp. 1–16.
[8] E. C. R. Shin, D. Song, and R. Moazzezi, "Recognizing functions in binaries with neural networks," in *Proc. USENIX Conf. Secur. Symp.*, 2015, pp. 611–626.
[9] R. Pascanu, J. W. Stokes, H. Sanossian, M. Marinescu, and A. Thomas, "Malware classification with recurrent networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Apr. 2015, pp. 1916–1920.
[10] W. Huang and J. W. Stokes, "MtNet: A multi-task neural network for dynamic malware classification," *Detection of Intrusions and Malware, and Vulnerability Assessment*. Cham, Switzerland: Springer, 2016, pp. 399–418.
[11] B. Kolosnjaji, A. Zarras, G. Webster, and C. Eckert, "Deep learning for classification of malware system call sequences," in *Proc. Australas. Joint Conf. Artif. Intell.*, 2016, pp. 137–149.
[12] W. Melicher, B. Ur, S. M. Segreti, S. Komanduri, L. Bauer, N. Christin, and L. F. Cranor, "Fast, lean, and accurate: Modeling password guessability using neural networks," in *Proc. USENIX Secur. Symp.*, 2016, pp. 1–18.
[13] M. Rhode, P. Burnap, and K. Jones, "Early-stage malware prediction using recurrent neural networks," *Comput. Secur.*, vol. 77, pp. 578–594, Aug. 2017.
[14] M. K. Putchala, "Deep learning approach for intrusion detection system (IDS) in the Internet of Things (IoT) network using gated recurrent neural networks (GRU)," Wright State Univ., Dayton, OH, USA, Tech. Rep., 2017.
[15] R. C. Aygun and A. G. Yavuz, "Network anomaly detection with stochastically improved autoencoder based models," in *Proc. IEEE 4th Int. Conf. Cyber Secur. Cloud Comput. (CSCloud)*, Jun. 2017, pp. 193–198.
[16] P. Godefroid, H. Peleg, and R. Singh, "Learn&Fuzz: Machine learning for input fuzzing," in *Proc. IEEE/ACM Int. Conf. Automated Softw. Eng.*, Nov. 2017, pp. 50–59.
[17] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[18] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," *Neural Comput.*, vol. 12, no. 10, p. 2451, 2000.
[19] M. Wollmer, F. Eyben, J. Keshet, A. Graves, B. Schuller, and G. Rigoll, "Robust discriminative keyword spotting for emotionally colored spontaneous speech using bidirectional LSTM networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Apr. 2009, pp. 3949–3952.
[20] M. Wollmer, C. Blaschke, T. Schindl, B. Schuller, B. Farber, S. Mayer, and B. Trefflich "Online driver distraction detection using long short-term memory," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 2, pp. 574–582, Jun. 2011.
[21] D. D. Monner and J. A. Reggia, "Recurrent neural collective classification," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 12, pp. 1932–1943, Dec. 2013.
[22] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," 2014, *arXiv:1406.1078*. [Online]. Available: https://arxiv.org/abs/1406.1078
[23] K. Yao, T. Cohn, K. Vylomova, K. Duh, and C. Dyer, "Depth-gated LSTM," 2015, *arXiv:1508.03790*. [Online]. Available: https://arxiv.org/abs/1508.03790

[24] J. Koutník, K. Greff, F. Gomez, and J. Schmidhuber, "A clockwork RNN," in *Proc. 31st Int. Conf. Mach. Learn.*, 2014, pp. 1863–1871.
[25] H. Manukian, F. L. Traversa, and M. Di Ventra, "Accelerating deep learning with memcomputing," *Neural Netw.*, vol. 110, pp. 1–7, Feb. 2019.
[26] J. Kim, A.-D. Nguyen, and S. Lee, "Deep CNN-based blind image quality predictor," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 1, pp. 11–24, Jan. 2019.
[27] S. Yao, Y. Zhao, A. Zhang, S. Hu, H. Shao, C. Zhang, L. Su, and T. Abdelzaher, "Deep learning for the Internet of Things," *Computer*, vol. 51, no. 5, pp. 32–41, 2018.
[28] T. Lei, Y. Zhang, S. I. Wang, H. Dai, and Y. Artzi, "Training RNNs as fast as CNNs," in *Proc. ICLR*, 2017.
[29] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 399–421, 1986.
[30] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016, pp. 528–566.
[31] *KDDCup99 Data*. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/
[32] *NSL-KDD Data*. [Online]. Available: http://nsl.cs.unb.ca/NSL-KDD/
[33] *CICDS2017 Dataset*. [Online]. Available: https://www.unb.ca/cic/datasets/ids-2017.html
[34] V. Katos, "Network intrusion detection: Evaluating cluster, discriminant, and logit analysis," *Inf. Sci.*, vol. 177, pp. 3060–3073, Aug. 2007.
[35] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. 2nd IEEE Symp. Comput. Intell. Secur. Defense Appl. (CISDA).*, Jul. 2009, pp. 1–6.
[36] S. K. Sahu, S. Sarangi, and S. K. Jena, "A detail analysis on intrusion detection datasets," in *Proc. IEEE Int. Adv. Comput. Conf.*, Feb. 2014, pp. 1348–1353.
[37] A. Divekar, M. Parekh, V. Savla, R. Mishra, and M. Shirole, "Benchmarking datasets for anomaly-based network intrusion detection: KDD CUP 99 alternatives," in *Proc. IEEE 3rd Int. Conf. Comput., Commun. Secur. (ICCCS)*, Oct. 2018, pp. 1–8.

JIN YANG received the M.S. and Ph.D. degrees in computer science from Sichuan University, Sichuan, China, in 2004 and 2007, respectively, where he is currently an Associate Researcher with the College of CyberSecurity. His main research interests include network security, knowledge discovery, and expert systems.

TAO LI received the Ph.D. degree in computer science from the University of Electronic Science and Technology of China, in 1994. He is currently a Professor with the College of Cyber-Security, Sichuan University, China. His main research interests include network security, and cloud computing and cloud storage.



GANG LIANG received the Ph.D. degree in computer science from the Sichuan University, in 2007, where he is currently an Associate Professor with the College of Cyber-Security, Sichuan University, China. His main research interests include network security and cloud computing.

WENBO HE received the Ph.D. degree in computer science from the University of Illinois at Urbana-Champaign, Urbana, IL, USA, in 2008. She was an Assistant Professor with the Department of Computer Science, University of New Mexico, Albuquerque, NM, USA, from 2008 to 2010. From 2010 to 2011, she was an Assistant Professor with the Department of Electrical Engineering, University of Nebraska-Lincoln, Lincoln, NE, USA. From 2011 to 2016, she was an Assistant Professor with the School of Computer Science, McGill University, Montreal, QC, Canada. She is currently an Associate Professor with the Department of Computing and Software, McMaster University. Her research interests include big data systems, mobile and pervasive computing, security and privacy, and cloud computing.

YUE ZHAO received the B.S. degree in communication engineering from the North China Institute of Science and Technology, Langfang, China, in 2006, and the Ph.D. degree in information and communication systems from Southwest Jiaotong University, Chengdu, China, in 2012. From 2010 to 2011, he was a Visiting Student with the Department of Electrical and Computer Engineering, University of Florida. He is currently a Senior Engineer with the Science and Technology on Communication Security Laboratory, Chengdu, China. His research interests include wireless networks and information security.