Cybersecurity

RESEARCH Open Access

On the combination of data augmentation method and gated convolution model for building effective and robust intrusion detection

Yixiang Wang[1†], Shaohua lv[1†], Jiqiang Liu[1], Xiaolin Chang[1*] and Jinqiang Wang[2]

Abstract

Deep learning (DL) has exhibited its exceptional performance in fields like intrusion detection. Various augmentation methods have been proposed to improve data quality and eventually to enhance the performance of DL models. However, the classic augmentation methods cannot be applied to those DL models which exploit the system-call sequences to detect intrusion. Previously, the seq2seq model has been explored to augment system-call sequences. Following this work, we propose a gated convolutional neural network (GCNN) model to thoroughly extract the potential information of augmented sequences. Also, in order to enhance the model's robustness, we adopt adversarial training to reduce the impact of adversarial examples on the model. Adversarial examples used in adversarial training are generated by the proposed adversarial sequence generation algorithm. The experimental results on different verified models show that GCNN model can better obtain the potential information of the augmented data and achieve the best performance. Furthermore, GCNN with adversarial training can enhance robustness significantly.

Keywords: Data augmentation, Intrusion detection system, Machine learning algorithms, System call

Introduction

An intrusion detection system (IDS) is a kind of active defense technique with the aim of resisting malware and sensitive activities. It mainly identifies malicious intrusions by monitoring network traffic or user behaviors. There are two types of detection systems, misuse-based and anomaly-based. The former type works by constructing a known attack pattern database and then identifying intrusion behaviors according to the pre-defined matching rules. The latter type focuses on normal behaviors, and when the system finds a behavior deviating from the pre-defined rules, it is determined to be an intrusion event. The ability to identify intrusion is a crucial factor in evaluating an intrusion detection system. However, current intrusion detection systems have some limitations. The misuse-based intrusion detection system needs a large number of attack pattern libraries and cannot identify unknown attacks, which will cause a high rate of false negatives. Due to the variability of user behavior habits, anomaly detection algorithms always have a high false-positive rate.

In recent years, there has been a growing number of publications focusing on the analysis of system-call sequences in the anomaly-based IDS. Forrest et al. (2008) overviewed the methods of analyzing the system-call sequences, including Hidden Markov Models (HMM), Bayes model and so on. Unfortunately, these methods delt with frequent system-call sequences without considering the semantic meanings of system-call sequences. Just as people express their thoughts through sentences and grammar, processes can also reach their goals through

*Correspondence: xlchang@bjtu.edu.cn
†Yixiang Wang and Shaohua lv contributed equally to this work.
[1]Beijing Key Laboratory of Security and Privacy in Intelligent Transportation, Beijing Jiaotong University, 3 Shangyuancun, 100044 Beijing, China
Full list of author information is available at the end of the article

specific system-call sequences. A system-call sequence can be considered as a language sentence which can interact with the operating system (Lv et al. 2018). Considering the high similarity between sentences and system-call sequences, using the language model to model the system-call sequences is an intuitive way to extract latent semantic information.

In order to alleviate the above limitations, researchers try to find answers in machine learning (ML) or deep learning (DL) techniques. DL techniques have been changing the landscape of mainstream research fields such as computer vision (Wang et al. 2019; Karras et al. 2019), natural language processing (Devlin et al. 2018; Lan et al. 2019), and also revolutionizing the way to solve the problems that traditional algorithms cannot handle. With the drive of big data and no need for manual feature extraction, DL makes it possible to address the issues of IDS (Xu et al. 2018; Liu et al. 2017; Singla et al. 2019).

As expected, a large number of results indicate that DL techniques are the key determinant of improving the classification results of intrusion detection (Ustebay et al. 2018; Al-Qatf et al. 2018; Hao et al. 2019). Meanwhile, the seq2seq model can extract the semantic information in the system-call sequences, and the predicted sequence from seq2seq model can be the augmented sequence to enhance the IDS model's performance (Bahdanau et al. 2015; Lv et al. 2018). There are a few researches on augmenting the system-call sequences such as adding some system calls into the sequences, but these previous works only concentrated on the generation of augmented sequences without considering how to apply augmented sequences to improve the robustness of the system-call model.

Inspired by the above discussions, we propose a gated convolutional neural network (GCNN) model, specially designed to deal with the augmented system-call sequences and capture the latent information from the augmented sequences. Moreover, we conduct adversarial training on the proposed GCNN model in order to enhance its robustness against adversarial example attacks. Concretely, our major contributions are summarized as follows:

- We propose a gated convolution intrusion detection model (GCNN) to unearth the prediction sequence based on the system-call sequence augmentation method (Lv et al. 2018). On the premise of the reliability of augmented sequences, the practical information is extracted from the prediction sequence through the gated mechanism to help the model capture the characteristics of latent program behaviors. We observe that compared with training on the augmented data directly, our model can produce better results.
- We propose a system-call adversarial sequence generation algorithm based on Fast Gradient Sign Method (FGSM) (Goodfellow et al. 2015) and we employ the adversarial examples generated by the proposed algorithm for adversarial training to better defend the attack of adversarial examples on GCNN model. The results show that the adversarial-trained GCNN can maintain 60.7% accuracy under adversarial example attack.

The rest of the paper is organized as follows. "Background and related work" section presents background knowledge and related work. "Methods" section describes the details of GCNN model, adversarial sequence generation algorithm and adversarial training. Experimental results and discussions are in "Experiment and analysis" section . Conclusion and future work are given in "Conclusion and future work" section.

Background and related work

This section first introduces the language model and seq2seq model with the attention mechanism in order to explain how the seq2seq model works. Then, we outline adversarial examples and adversarial training. Finally, we overview the achievements and urgent issues in the field of intrusion detection.

Language model and seq2seq model

A language model (Chung et al. 2014) is a model that can calculate the probability that a sequence is a sentence from the perspective of linguistics. It can be formalized as the conditional probability of the occurrence of the next word given all the previous words, which is shown below.

$$p\left(w_1, w_2, \cdots, w_{L-1}, w_L\right) = \prod_{l=1}^{L} p\left(w_l \left| w_1, w_2, \cdots, w_{l-1}\right.\right)$$

where L is the number of words in the sentence, and w_l is the l_{th} word in the sentence. Recurrent neural network (RNN) is one of the language models, denoted as the RNN based language model (RNNLM)(Mikolov et al. 2010). RNNLM uses the previous input to determine the output of the current location. In practice, the RNNLM determines the semantic rationality of the predicted word. We apply RNNLM to the generation of system calls based on its generality. Here, in order to deal with input sequences and output sequences, we choose a sequence-to-sequence structure described in Sutskever et al. (2014) and Bahdanau et al. (2015), where the structure is *RNN Encoder* − *Decoder*.

Concretely, the structure of the seq2seq model is shown in Fig. 1. h denotes the hidden state of the encoder (E) and H is the hidden state of the decoder (D). As implied by the name, the seq2seq model takes the sequences as the

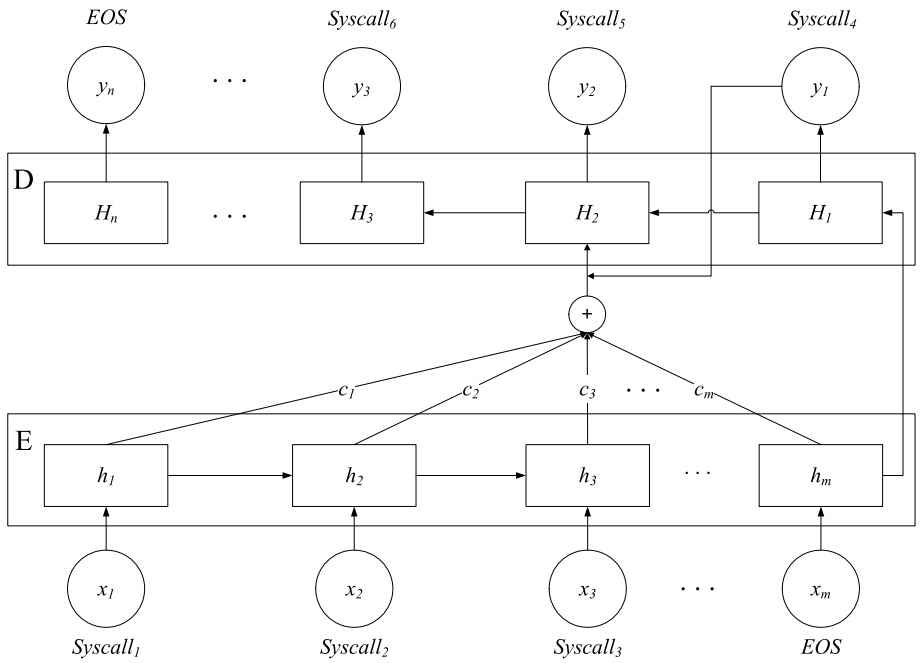

Fig. 1 The structure of the seq2seq model with the attention mechanism

input and produces the predicted sequences. It contains two parts, namely, encoder and decoder. The encoder is a Long Short-Term Memory (LSTM) model, which is a sophisticated RNN architecture to solve the gradient vanishing problem of vanilla RNN. The encoder is responsible for encoding a sequence into a context vector, which contains the latent semantic information. The decoder is also an LSTM RNN model that uses the context vector as the initial hidden state and parses the latent semantics from it to predict the subsequent system-call sequence. Since the decoder only relies on one context vector, which limits the relationship between the input sequence and output sequence, the attention mechanism is introduced. The attention mechanism combines each hidden state of the encoder in a probabilistic way to form a context vector. In this way, each predicted system call of the decoder is associated with each input element. Meanwhile, the results (Lv et al. 2018) show the seq2seq model with attention mechanism can obtain better predicted sequences. For the above reasons, we choose the seq2seq with the attention mechanism to generate predicted system-call sequences.

Adversarial attacks and defense

Adversarial examples are the model input that causes the model misclassification. Szegedy et al. (2014) was the first to discover the existence of adversarial examples and reveal the attribute of adversarial examples — transferability. Transferability means that adversarial examples generated by model A can misclassify model B. Subsequently, Goodfellow et al. (2015) introduced a white-box attack, Fast Gradient Sign Method(FGSM), and adversarial training, where FGSM is an effective and computation-less method to generate adversarial examples. White-box means that the attacker has access to the dataset, model parameters and any information. The core function of FGSM is shown below.

$$X^* = X + \varepsilon sign\left(\nabla_x J\left(\theta, X, y\right)\right)$$

where X is the original clean sample and y is the corresponding label. X^* is the adversarial sample. ∇ is the derivation calculation. ε is the distortion rate. J is an objective function and θ is the parameters of the attacked model. At present, the research on adversarial examples focuses on the images (Akhtar and Mian 2018). There are some researches on the adversarial system-call sequences, where Rosenberg et al. (2018) introduced a black-box adversarial sequence attack. In this paper, we will construct a white-box adversarial sequence generation algorithm based on FGSM to craft adversarial examples for adversarial training to enhance the model's robustness.

Adversarial training is a method that combines the original legitimate samples and adversarial samples to retrain the model, which can improve the robustness of the model. The adversarial training objective function based on the FGSM is shown below.

$$\widetilde{J}\left(\theta, X, y\right) = J\left(\theta, X, y\right) + J\left(\theta, X + \varepsilon sign\left(\nabla_x J\left(\theta, X, y\right)\right), y\right)$$

The goal of adversarial training is to improve the generalization ability of the model. First of all, the model is

trained on the original legitimate data set, and after training, the adversarial examples are generated on the basis of the model parameters and generation algorithm. Then, adversarial examples are used as the training data, and the optimization function of the model is modified, which is shown above, to train the second time. After adversarial training, the model has a strong defense ability against the adversarial examples.

Related work

The fact that the attacks are more sophisticated and even more frequent requires intrusion detection technology to develop towards standardization and intelligence. Deep learning technology has been applied to intrusion detection. Gao et al. (2014) applied a deep belief network(DBN) to the KDD CUP 1999, whose results showed that deep neural networks outperformed traditional machine learning algorithms. Ustebay et al. (2018) combined recursive feature elimination and deep multilayer perceptron to classify benign traffic and DDoS traffic. Kim and Kim (2015) was the first to use RNN to solve the classification on the DARPA dataset (available in the website). Then LSTM recurrent neural network (Hochreiter and Schmidhuber 1997; Kim et al. 2016; Hao et al. 2019) and gated recurrent unit (GRU) network (Xu et al. 2018) were applied to get better performance than vanilla RNN. At this point, the researchers considered migrating the relevant technologies of natural language processing to the field of intrusion detection (Cho et al. 2014). Lv et al. (2018) were the first to apply a seq2seq model to the prediction of the system-call sequence.

Deep learning models converge through continuous iterations of model parameters based on a large amount of data. Without enough data, the model can not perform well. Therefore, data augmentation methods are introduced to solve the problem. In the computer vision area, there are many augmentation methods such as random rotation, random flipping, and so on (Shorten and Khoshgoftaar 2019). Also, in natural language processing, randomly insert, randomly swap, or delete (Wei and Zou 2019; Kobayashi 2018), even GANs (Kusner and Hernández-Lobato 2016; Yu et al. 2019) can be used to augment a dataset. Lv et al. (2018) was the first to use the predicted sequence generated from the seq2seq model as the augmented sequence. But it only proposed an augmentation generation algorithm, there have been no methods to deal with augmented sequences efficiently. We introduce a model, GCNN, to fill the gap. Rosenberg et al. (2018) introduced a black-box attack to generate adversarial sequences. We prefer a white-box attack to generate adversarial sequences by adding perturbation in the padding part, and we make GCNN more robust by adversarial training.

Methods

In this section, we first introduce an intuitive CNN-based model to handle the predicted sequence in "Convolutional intrusion detection model based on augmented sequence" section. However, there is a problem with the intuitive model, where the predicted sequence is as important as the original sequence. To solve it, we further propose a GCNN model in "GCNN intrusion detection model based on predicted sequence augmentation method" section. Afterwards, an FGSM-based adversarial sequence generation algorithm is proposed to generate adversarial sequences for adversarial training in "Adversarial sequence generation algorithm" section. In "Metrics" section, we demonstrate the metrics in the experiments.

Convolutional intrusion detection model based on augmented sequence

At the beginning, we give a brief introduction of augmented sequences. Then, we present the convolution operation and maxpool opration to deal with sequential data, where these two operations differ in image data. The system-call sequence augmentation method first inputs each training data sequence into the seq2seq model for prediction, and then expands it in the way of one-to-one correspondence between the prediction sequence and the original sequence. For an input sequence, the follow-up sequence that can be calculated by the prediction model is:

$$x_e = Predict\,(x)$$

where the function $Predict\,(x)$ is a function that the seq2seq model learns. Then, we concatenate x and x_e to get $\tilde{x}$, where $\tilde{x} = \{x, x_e\}$. Here, the amount of training data does not increase, but each sequence of training data contains more information.

The system call sequence is quite different from the image. We vectorize each system call. For a sequence of length l, a vector of dimension $l \times m$ is generated after the word embedding layer, where m is the dimension of the word vector. For system-call sequences, each word vector has its semantic information. There are latent meaningful local features in the embedded matrix, so a convolutional kernel of size $n \times m$ should be used to extract the local associations in the embedded matrix. Depending on the convolutional kernel size n, we can obtain different n-gram information. Each convolutional kernel k_i^n learns a feature vector v_i^n, where i means the i-th kernel. The output channel of the kernel k_i^n can be set to c_i , and the dimension d of the feature vector v_i^n can be calculated by the formula:

$$d = \frac{(l-n)}{s} + 1$$

where s is the stride when the kernel moves. After convolutional calculation, the dimension of the output feature vector about kernel k_i^n is $c_i \times d_i \times 1$. Then, the maximum pooling (MaxPool) algorithm is used to extract higher-level abstract feature vectors from the output feature vectors. Here, the MaxPool algorithm differs from that in computer vision since image processing selects the maximum value in the pooling window as the output, which belongs to local pooling, as shown in Fig. 2. But in our model, MaxPool is global pooling, where the maximum value of the entire vector is selected as the output. The formula is as follows:

$$MaxPool\,(v) = \text{argmax}\,(v)$$

We concatenate these higher-level abstract feature vectors of different kernels into a matrix. Finally, the classification output is performed through the fully connected layer and the softmax layer. Figure 3 describes the feature extraction and pooling process of the system-call word vector convolutional neural network, where different colors in embedding represent different convolutional kernels.

The input of our convolutional intrusion detection model is to directly append the prediction sequence after the original sequence as an augmented sequence. This is a simple and intuitive way since the augmented sequence can maintain the time dependency between the prediction sequence and the original sequence to the greatest extent. To sum up, the complete model architecture is shown in Fig. 4. Firstly, we take the augmented sequence directly as the network input and obtain the word vector of each system call through the embedding module. Then the convolution structure described in Fig. 3 is used to extract the local relation features of the system calls, and the MaxPool operation is performed on the feature vectors extracted by convolution kernels. Finally, the softmax layer outputs the classification results.

GCNN intrusion detection model based on predicted sequence augmentation method

There is a limitation in the convolutional intrusion detection model described above. That is, the predicted sequence is as important as the original sequence in the training phase. We believe the predicted sequence as a supplementary part of the information should provide useful information to the model. Still, there may be some invalid information in the predicted sequence, which will affect the extraction of attack features by the model. To eliminate this effect, we propose a more advanced model, gated convolutional neural network (GCNN). The model has the ability to use thresholds to select the effective part of the prediction vector, so as to better fit the objective function.

The concrete structure of our proposed GCNN is shown in Fig. 5. During the training of GCNN, the original sequence x and the predicted sequence x_e are extracted using different convolution kernels to obtain the feature vectors f and f_e, respectively. Both convolution operations use *ReLU* (Nair and Hinton 2010) as the activation function. It is worth noting that the convolution kernel of x_e uses extra activation function, *sigmoid*, to extract g on x_e. *Sigmoid* function limits the range of vector element to [0,1], and we think this reasonable transformation plays a gated role to distll the abstract features. Also, *ReLU* functions plays a gated role to maintain positive features. In this way, we obtain two types of features, distilled feature and positive feature. Then, the Hadamard product of f_e and g is used to control the contribution of the predicted feature vector, selectively. The Hadamard product of *ReLU* results and *Sigmoid* results is what we call the gated mechanism in this paper. Finally, the original sequence vector is added to get the fusion vector v_f. After the *MaxPool* layer and *Softmax* layer we can get the outputs. The computing graph of GCNN model is:

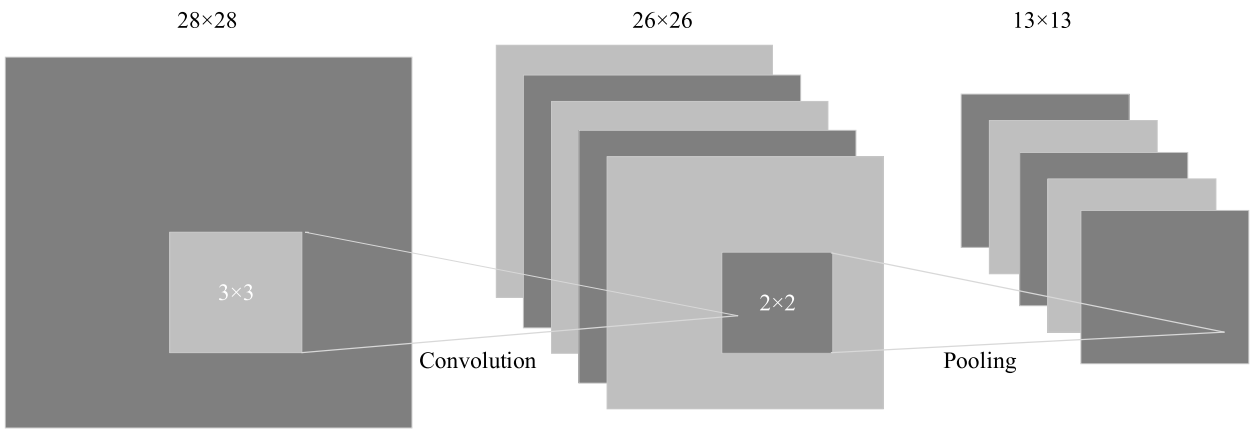

Fig. 2 The process of convolution and pooling in the image CNN model

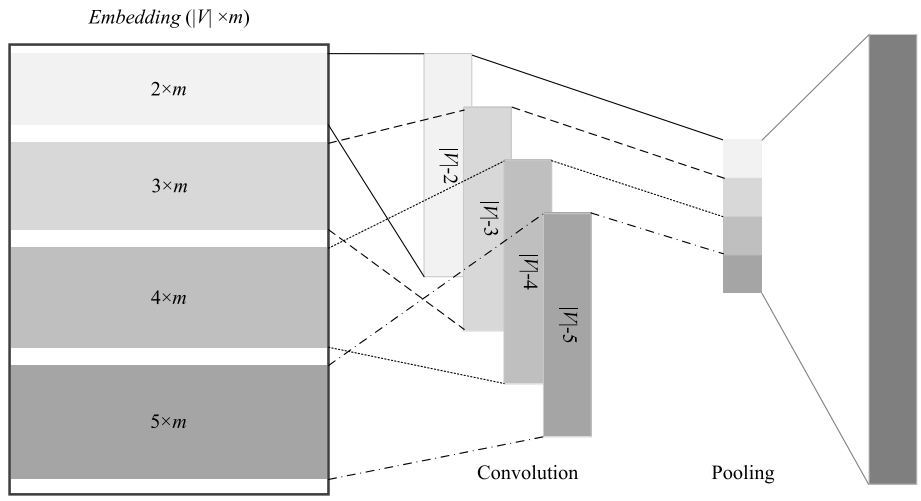

Fig. 3 The process of convolution and max-pooling in the system call intrusion-detection model

$$
\begin{aligned}
f &= ReLU\left(conv\left(embedding\left(x\right)\right)\right)\\
f_e &= ReLU\left(conv\left(embedding\left(x_e\right)\right)\right)\\
g &= sigmoid\left(conv\left(embedding\left(x_e\right)\right)\right)\\
v_f &= f + f_e \odot g\\
o &= MaxPool\left(v_f\right)\\
y &= softmax\left(Wo + b\right)
\end{aligned}
$$

Adversarial sequence generation algorithm

The generation of adversarial examples in this section prepares data for the adversarial training defense. Adversarial examples require small perturbation to the input samples. At present, all the researches on adversarial examples focused on the images (Akhtar and Mian 2018). For an image, the value of each pixel is within [0,1], and small changes to each pixel will not be captured by human eyes. But it is especially tough to disturb the sequence data. For the system-call sequence, each word vector represents the semantic information of a system call, which will lose its original meaning after being disturbed directly. Therefore, the traditional adversarial example generation algorithm is not suitable (or even impossible) to construct adversarial examples for system-call sequences. In order to solve this problem, we propose a system-call adversarial sequence generation algorithm based on the FGSM attack.

The proposed algorithm makes use of the padding characteristics of the input. Iterative perturbation is performed on the zero-padding part of the input sequence, and a perturbed word vector is added in each iteration until the model outputs incorrectly or the padding part is completely replaced. The perturbed adversarial sequence does

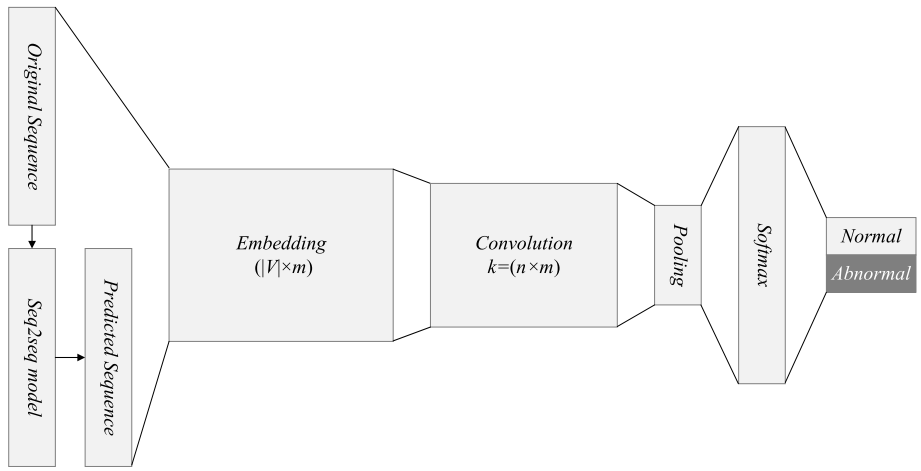

Fig. 4 CNN classification for anomaly intrusion detection trained based on the extended system call sequence

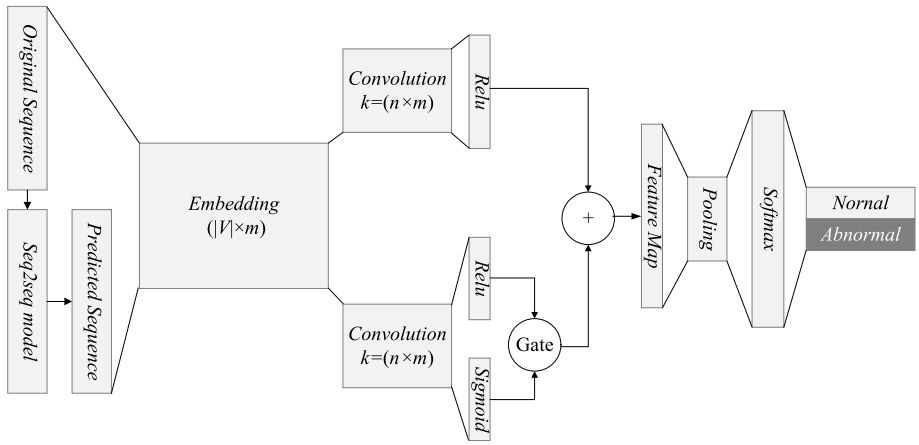

Fig. 5 The structure of gated CNN classification model for anomaly intrusion detection

not destroy the original features, but only adds perturbation in the padding part. This is because the *mask* operation is adopted to cover up the real sequence in the perturbation algorithm and prevent the algorithm from modifying it. The pseudo-code is described in Algorithm 1.

Concretely, we first convert the initial system call sequence, for instance, the sequence in Fig. 6, to embedded vectors through the *embedding* layer. Also, we can get all the system call word vectors, *EmbeddedMatrix*, according to the *embedding* layer. Then, we add the perturbation to the padding part of a system call sequence using FGSM algorithm and *mask* operation. To make the adversarial sequence close to the legitimate sequence, each vector in the padding part calculates the euclidean distance with every vector in *EmbeddedMatrix*. After that, we can obtain the legitimate closet vector index, s_e. Then we replace some perturbation with the legitimate closet vector in the padding part. In the end, a system-call adversarial sequence is generated by repeating the above process. We can use the adversarial sequences generated by Algorithm 1 to do adversarial training.

Algorithm 1: System-call Adversarial Sequence Generation Algorithm

x is the legitimate sequence. l is the legitimate sequence length. m is the input sequence length, *embedding* is a layer that transforms the a system call to a vector. *EmbeddedMatrix* is the whole system-call vectors. ε is the perturbation coefficient. $J(\cdot)$ is the objective function.

Input: x
Output: x^*
$x^* \leftarrow x$;
$x_e \leftarrow embedding(x)$;
for $idx = (l - m)$ ***to*** l **do**
  $noise = \varepsilon sign\left(\nabla_{x_e} J(\theta, x_e, y)\right)$;
  $x_e = x_e + mask(noise)$;
  $s_e = \min(euclidean\_distance!(x_e[idx], EmbeddedMatrix))$;
  $x^*[idx] = EmbeddedMatrix[s_e]$;
  if $Model(x_e) \neq Lable(x)$ **then**
    return x^*;
  end
end
return x^*;

Metrics

We include two parts of the experiment. The first part is the evaluation of the classification ability of the model, and the other is the evaluation of the robustness of the model.

In the first part of the experiment, accuracy is the most intuitive evaluation metric in the classification problem, which measures the proportion of correctly classified samples to the total samples. For data with uneven distribution of positive and negative examples, the accuracy will bias the evaluation of the model. So, we need to introduce other metrics to assess models' capabilities. In the binary classification, the classification results of the model will show the following four outcomes, which make up the confusion matrix (Stehman 1997) as shown in Table 1.

There are many evaluation metrics (Powers 2011) derived from the confusion matrix, such as precision and recall. Precision represents the proportion of positive samples in the data predicted by the model to be positive. Recall represents the proportion of the data predicted by the model as positive to the actual positive samples. The calculation formulas of precision and recall and other metrics are shown in Table 2.

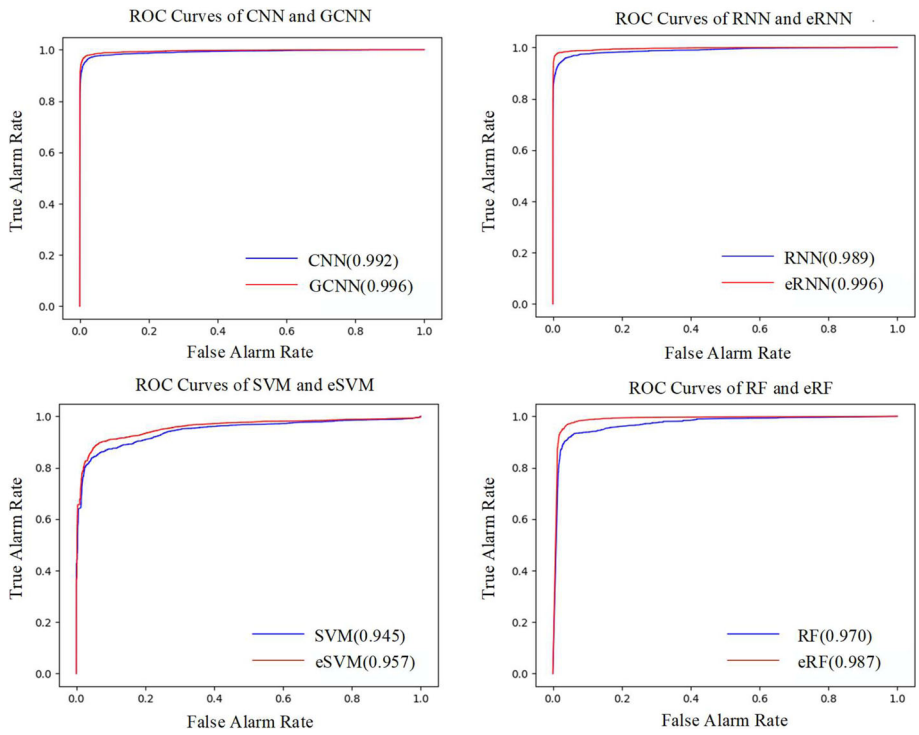

Fig. 6 The process of crafting training set and test set

The ROC curve (Fawcett 2006) calculates the TPR and FPR of the model according to the classification threshold, and then draws the curve with TPR as the vertical coordinate and FPR as the horizontal coordinate. The larger the FPR is, the more the actual negative samples are in the predicted positive class. In addition, the larger the TPR is, the more the real positive examples are in the predicted positive class. In general, the area under the curve (AUC) of the ROC curve is used to evaluate a classifier directly. The characteristic of the ROC curve is very stable. When the distribution of positive and negative samples in the test set changes, the ROC curve can remain unchanged. Thus, it is suitable to evaluate the uneven data.

In the adversarial training part of the experiment, we introduce three evaluation metrics: Acc_{adv}, perturbation rate and crafting rate. ACC_{adv} score is the accuracy score on the adversarial examples, which presents the ability of misclassification of adversarial sequences generated by the proposed algorithm 1. The perturbation rate is the percentage of the perturbed elements in all the elements when generating the adversarial sequence, indicating the degree of added noise. Crafting rate is the proportion of generated sequences that can cause misclassification in all generated sequences. Crafting rate of generating adversarial examples can intuitively reflect whether the adversarial examples are successfully generated or not. These three indicators can fully reflect the quality of the adversarial sequence and the robustness of the model after adversarial training.

Table 1 The Confusion Matrix

	Positive sample	Negative sample
True prediction	True Positive (TP)	False Positive (FP)
False prediction	False Negative (FN)	True Positive (TP)

Experiment and analysis

In this section, we describe the dataset and how we preprocess it. Model details and hyper-parameters setups in the experiments are introduced. Then, we analyze the classification performance among verification

Table 2 The calculation formula for the evaluation metric of the binary classification model

Metric	Formula
Precision	$Precision = \frac{TP}{TP+FP}$
Recall	$Recall = \frac{TP}{TP+FN}$
FPR	$FPR = \frac{FP}{FP+TN}$
TNR	$TNR = \frac{TN}{FP+TN}$
$F1-score$	$F1{-}score = \frac{2TP}{2TP+FP+FN}$
Accuracy	$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$

Table 5 Overview of the ADFA-LD dataset

category		file number
Normal		833
Abnormal	Adduser	91
	Hydra_FTP	162
	Hydra_SSH	176
	Java_Meterpreter	124
	Meterpreter	75
	Web_Shell	118

models. Finally, we make a comprehensive evaluation of the model's robustness after adversarial training.

Verification model description

For the purpose of verifying the proposed method, we choose four standard intrusion detection classifiers based on machine learning for comparison, including support vector machine (SVM) model (Cortes and Vapnik 1995), random forest (RF) model (Kam 1995), convolutional neural network (CNN) model (LeCun et al. 1989; LeCun et al. 1998), and recurrent neural network (RNN) model (Elman 1990). The reason for choosing these four models is that they are represented in their respective technical fields.

SVM maps the input space to the high-dimensional feature space through the kernel function. Then optimal separation hyperplane is constructed to classify the input data. RF is a classic representative of ensemble learning. It is an algorithm that builds a variety of decision tree models into a kind of decision model in an ensemble learning manner. Because RF has the characteristics of easy implementation, low computational overhead, and powerful performance, it is widely used in a variety of learning tasks. RNN's outstanding sequence processing ability makes it shine in time series modeling and has now become the basic model in natural language processing tasks. CNN also has an irreplaceable position in the field of computer vision. We have a structural improvement in convolutional neural networks in our work.

Table 3 Examples of correspondences between numbers and system-call names

num	system-call name
0	sys_restart_syscall
1	sys_exit
2	sys_fork
3	sys_read
4	sys_write
...	
307	sys_recvmmsg

Table 4 The ADFA-LD dataset

Train		Test	
seq_len	Pairs	seq_len	Pairs
10	8000	10	1600
20	6000	20	1200
30	4000	30	800

For the recurrent neural network, we construct a two-layer LSTM neural network with 256 neurons. Hidden layers are connected using Dropouts. The cross entropy function is used to calculate the loss value, and Adam optimizer is used to update the models' parameters. For the input sequence, the recurrent network uses the hidden state at the last timestep as the sequence's semantic vector. Then, the semantic vector connects to a fully connected neural network for classification. For the convolutional neural network, three different sizes of convolution kernels are used, whose sizes are [3, 4, 5], and each size has 128 convolution kernels.

Both models have two fully connected layers, whose nodes are 128 and 32, and use $L2$ regularization to prevent overfitting. The settings of the hyper-parameters, such as the optimization algorithm and the learning rate, are the same for these two models. We share parameters for the two *ReLU* convolution operations when training GCNN model because the purpose of both is to extract the semantic relationship between the original sequence and the predicted sequence. Sharing parameters can help the model to fit the objective function faster, and reduce the memory and processor consumption in the training phase. SVM and RF which we use are the default models in the scikit-learn python package.

In our experiments, verification models are trained on two kinds of inputs: original sequences and augmented sequences. Models trained on original sequences are called RNN, CNN, SVM, RF. Models trained on augmented sequences are called eRNN, eCNN, eSVM, eRF, GCNN.

Dataset description and preprocessing

In our experiments, we choose ADFA Linux Dataset (ADFA-LD) (Creech and Hu 2013) because the ADFA-LD dataset includes six abnormal attacks occurring in the

Table 6 The performance of models trained with the original sequence

Classfier	Accuracy	Precision	Recall	F1-score
RNN	95.8%	96.5%	96.6%	96.6%
CNN	96.9%	97.7%	97.1%	97.4%
SVM	86.9%	85.9%	94.2%	89.8%
RF	94.7%	94.6%	96.9%	95.7%

real-world Ubuntu 11.04 system, which means the attacks contained in the dataset are closer to real situations. Also, the ADFA-LD dataset is the latest open source system-call dataset in these years, newer than the UNM dataset and DARPA dataset. Moreover, the number of sequences in the data set is large enough to be used to train neural networks. Table 5 shows the number of files for each category in the dataset. Each file is a system call sequence that a process invokes during execution. It is worth noting that each sequence is composed of a string of numbers, and each number represents a system call name. Table 3 shows some examples of the corresponding relationships between numbers and the actual system calls.

We give an example to demonstrate how we process the dataset. Assuming that a normal sequence file contains a system-call sequence of length 100, we can slice this sequence. If we want the seq2seq model's input-output sequence pair with length 5, then we can get 10 pairs. The schematic diagram is shown in Fig. 6. We do the same thing on the whole sequence files in the dataset. Afterward, we divide all the sequence pairs into a training set and a test set at a ratio of 5 : 1. Through the construction method mentioned above, the training set and test set used in the experiment are shown in Table 4. One thing to declare is that we do not pay much attention to the system-call names, because the model cannot learn from the system call names directly. On the contrary, the sequence contains more semantic information.

In the verification stage of the experiment, due to the large differences in the models' structures used in the experiments, we need to do specific data preprocessing for a specific model. For convolutional neural networks and recurrent neural networks, it is necessary to use word embedding to convert system calls into word vector for calculation. For SVM and RF, a vector space model is used to map the system call sequence to a feature vector. The vector dimension is the number of system calls, which is 307 shown in Table 3, and each component value represents the frequency of the system call in the sequence. For instance, for the sequence {3,4,7,7,9,11,3,3,4,7,12}, the mapped vector is expressed as {0,0,3,2,0,0,3,0,1,0,1,1,0,0},

Table 7 The performance of models trained with augmented sequence

Classfier	Accuracy	Precision	Recall	F1-score
eRNN	**97.6%**	97.8%	98.3%	98.0%
eCNN	97.3%	97.8%	97.7%	97.8%
eSVM	87.4%	86.0%	94.8%	90.2%
eRF	96.2%	96.0%	97.8%	96.9%
GCNN	*97.5%*	**98.8%**	**98.8%**	**98.8%**

When training, we use the normalization method to normalize the elements of the vector to be distributed among [0,1].

Model classification analysis

Tables 6 and 7 show the comparative performance of verification models under all metrics. From Tables 6 and 7, we can see that compared with the models trained on the original sequences, the performance of the four intrusion detection models trained on the augmented sequences has been significantly improved. Concretely, the metrics of the eRNN model are improved by about 1.5%, while the evaluation values of the eRF models are improved by 2%. However, compared with the CNN model and the SVM model, eCNN and eSVM have no noticeable improvement.

By comparing the evaluation indicators of each model, we can see that GCNN has achieved the best results of four models under the evaluation of all metrics. It strongly proves that augmented sequences can provide more practical information for the model and enable the model to detect potential intrusion behaviors and to provide supports for decision making. The convolutional gate helps GCNN model to fit the feature function of the attack sequence by selectively extracting useful information. Meanwhile, the characteristics of the potential attack behavior are enlarged in the sequence through the model, so as to improve the model's ability to classify and decide abnormal sequences.

Figure 7 shows the ROC curves and AUC values of classification models trained on the original sequences and

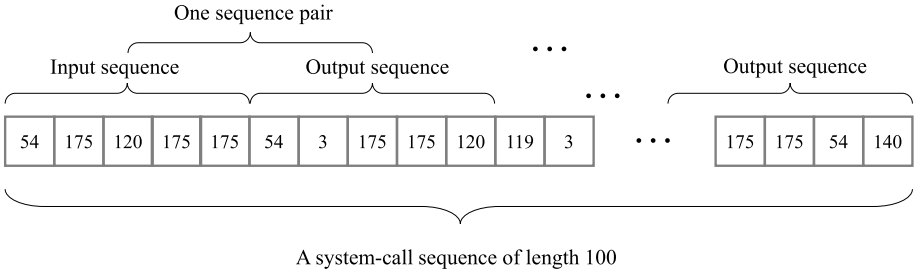

Fig. 7 ROC curves and AUC values of intrusion detection models trained on the original sequences and the augmented sequences respectively

Table 8 Comparison between models with adversarial training and models without adversarial training

	Model	ACC_{orig}	ACC_{adv}	Perturbation rate	Crafting rate
Adversarial sequences attack	CNN	96.40%	34.40%	19.50%	64.20%
	GCNN	97.50%	35.40%	22.70%	63.70%
Adversarial training	CNN	96.90%	45.20%	28.00%	53.20%
	GCNN	97.90%	60.70%	28.60%	37.90%

augmented sequences. As can be seen from the figure, the classification ability of the intrusion detection models has been improved compared with the models trained on original sequences and our proposed model, GCNN, achieves the best AUC value. In summary, GCNN can fully extract the potential semantic information from the augmented sequence to improve its classification ability. In the next section, we will improve its robustness by adversarial training.

Model robustness analysis

In our experiments, we choose the half of the test set to generate adversarial sequences and evaluate the effectiveness of the adversarial sequence generation algorithm mentioned in "Adversarial sequence generation algorithm" section. Then, we use the generated adversarial sequences to do adversarial training. Afterward, adversarial sequences are generated on the other half of the test set and are used to attack adversarial-training models. Eventually, we evaluate the robustness of the model after adversarial training.

The first three rows of Table 8 represent the performance of the two models, mentioned in "Convolutional intrusion detection model based on augmented sequence" and "GCNN intrusion detection model based on predicted sequence augmentation method" sections, on the augmented sequences and adversarial sequences generated by algorithm 1. It can be seen that the adversarial sequence can cause a great decrease in the accuracy of CNN and GCNN models, where the accuracy of the CNN model drops by 62%, and the accuracy of GCNN drops by 60.6%. From the perspective of perturbation rate and crafting rate, the average perturbation rate is about 21%, and the crafting rate is about 64%. We believe that the proposed adversarial sequence algorithm can generate effective adversarial sequences with a high crafting rate and low perturbation rate, and can be used to do adversarial training.

The last two rows of Table 8 are the evaluation of adversarial sequences on adversarial-training models. It can be seen that under the premise of the classification accuracy, the adversarial-training model can better resist the attack of the adversarial sequences. Under the attack of adversarial sequences, the accuracy of GCNN decreases only by 37.2%. Also, the increased perturbation rates and decreased crafting rates indicate that the difficulty of generating adversarial sequences on the adversarial-training models, and the concealment of the adversarial sequences becomes weak. This indirectly shows that the robustness of the model after adversarial training is indeed enhanced.

From Table 8, we can conclude that adversarial training of the model through the system-call adversarial sequences constructed in Algorithm 1 can effectively improve the classification accuracy and robustness of the model, and the robustness improvement of GCNN is more prominent.

Conclusion and future work

To address the problem of under-reporting and false positives and improve the detection capability of the intrusion detection system effectively, we propose a gated convolutional neural network architecture based on the augmented sequence, where the classification performance is significantly improved. Furthermore, to enhance the robustness and security of the model, an adversarial sequence generation algorithm based on FGSM is proposed to generate adversarial examples for adversarial training, and the ability of the model to resist adversarial attack is improved through adversarial training. The effectiveness of the proposed method and the model are verified through experimental analysis.

In future work, research on the experimental investigation of scenarios of live traffic data and live processing overhead for IDS will be carried out. Also, this paper focuses on the convolution extraction of n-gram information. Using sophisticated RNN structures such as GRU or LSTM to extract semantic information will be considered in future work. Moreover, we take into consideration that how to compare black-box attacks with white-box attacks to design better algorithms.

Authors' contributions

Drafting the manuscript: Yixiang Wang, Shaohua lv, Jinqiang Wang. Revising the manuscript critically for important intellectual content: Jiqiang Liu, Xiaolin Chang. All authors read and approved the final manuscript.

Funding

This work was supported in part by the Fundamental Research Funds for the Central Universities of China under Grants 2019YJS049.

Competing interests

No potential conflict of interest was reported by the authors.

Author details
[1]Beijing Key Laboratory of Security and Privacy in Intelligent Transportation, Beijing Jiaotong University, 3 Shangyuancun, 100044 Beijing, China. [2]Beijing Key Lab of Traffic Data Analysis and Mining, Beijing Jiaotong University, 3 Shangyuancun, 100044 Beijing, China.

Received: 20 April 2020 Accepted: 8 October 2020
Published online: 15 December 2020

References

Akhtar N, Mian A (2018) Threat of adversarial attacks on deep learning in computer vision: A survey. IEEE Access 6:14410–14430
Al-Qatf M, Lasheng Y, Al-Habib M, Al-Sabahi K (2018) Deep learning approach combining sparse autoencoder with svm for network intrusion detection. IEEE Access 6:52843–52856
Bahdanau D, Cho K, Bengio Y (2015) Neural Machine Translation by Jointly Learning to Align and Translate. cite arxiv:1409.0473Comment: Accepted at ICLR 2015 as oral presentation. https://nyuscholars.nyu.edu/en/publications/neural-machine-translation-by-jointly-learning-to-align-and-trans-2
Cho K, Van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y (2014) Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078:1724–1734. https://www.aclweb.org/anthology/D14-1179.bib
Chung J, Gulcehre C, Cho K, Bengio Y (2014) Empirical evaluation of gated recurrent neural networks on sequence modeling. ACM, Montreal. https://nyuscholars.nyu.edu/en/publications/empirical-evaluation-of-gated-recurrent-neural-networks-on-sequen
Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20(3):273–297
Creech G, Hu J (2013) Generation of a new ids test dataset: Time to retire the kdd collection. In: 2013 IEEE Wireless Communications and Networking Conference (WCNC). IEEE, Shanghai. pp 4487–4492. https://ieeexplore.ieee.org/document/6555301
Devlin J, Chang M-W, Lee K, Toutanova K (2018) Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805:4171–4186. https://www.aclweb.org/anthology/N19-1423.bib
Elman J (1990) Finding structure in time. Cogn Sci 14(2):179–211
Fawcett T (2006) An introduction to roc analysis. Pattern Recogn Lett 27(8):861–874
Forrest S, Hofmeyr S, Somayaji A (2008) The evolution of system-call monitoring. In: 2008 Annual Computer Security Applications Conference (ACSAC). IEEE, Anaheim. pp 418–430. https://ieeexplore.ieee.org/document/4721577
Gao N, Gao L, Gao Q, Wang H (2014) An intrusion detection model based on deep belief networks. In: 2014 Second International Conference on Advanced Cloud and Big Data. IEEE, Huangshan. pp 247–252. https://ieeexplore.ieee.org/document/7176101
Goodfellow IJ, Shlens J, Szegedy C (2015) Explaining and Harnessing Adversarial Examples
Hao S, Long J, Yang Y (2019) Bl-ids: Detecting web attacks using bi-lstm model based on deep learning. In: International Conference on Security and Privacy in New Computing Environments. Springer. pp 551–563
Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
Kam H (1995) Random decision forest. In: Proceedings of the 3rd International Conference on Document Analysis and Recognition, vol. 1416. IEEE, Montreal, Canada, August. p 278282
Karras T, Laine S, Aila T (2019) A style-based generator architecture for generative adversarial networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. IEEE, Long Beach. pp 4401–4410. https://ieeexplore.ieee.org/abstract/document/8953766
Kim J, Kim H (2015) Applying recurrent neural network to intrusion detection with hessian free optimization. In: International Workshop on Information Security Applications. Springer, Jeju Island. pp 357–369. https://link.springer.com/book/10.1007/978-3-319-31875-2
Kim J, Kim J, Thu H, Kim H (2016) Long short term memory recurrent neural network classifier for intrusion detection. In: 2016 International Conference on Platform Technology and Service (PlatCon). IEEE, Jeju. pp 1–5. https://ieeexplore.ieee.org/document/7456805
Kobayashi S (2018) Contextual augmentation: Data augmentation by words with paradigmatic relations. arXiv preprint arXiv:1805.06201:452–457. https://www.aclweb.org/anthology/N18-2072.bib
Kusner M, Hernández-Lobato J (2016) Gans for sequences of discrete elements with the gumbel-softmax distribution. arXiv preprint arXiv:1611.04051
Lan Z, Chen M, Goodman S, Gimpel K, Sharma P, Soricut R (2019) Albert: A lite bert for self-supervised learning of language representations. arXiv preprint arXiv:1909.11942
LeCun Y, Boser B, Denker J, Henderson D, Howard R, Hubbard W, Jackel L (1989) Backpropagation applied to handwritten zip code recognition. Neural Comput 1(4):541–551
LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324
Liu Y, Liu S, Zhao X (2017) Intrusion detection algorithm based on convolutional neural network. DEStech Trans Eng Technol Res (iceta):62–67. https://dl.acm.org/doi/abs/10.1145/3313991.3314009
Lv S, Wang J, Yang Y, Liu J (2018) Intrusion prediction with system-call sequence-to-sequence model. IEEE Access 6:71413–71421
Mikolov T, Karafiat M, Burget L, Cernocký J, Khudanpur S (2010) Recurrent neural network based language model. Twelfth Annu Conf Int Speech Commun Assoc:1045–1048
Nair V, Hinton G (2010) Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th International Conference on Machine Learning (ICML-10). ACM, Madison. pp 807–814. https://dl.acm.org/doi/10.5555/3104322.3104425
Powers D (2011) Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. https://dl.acm.org/doi/10.1145/1143844.1143874
Rosenberg I, Shabtai A, Rokach L, Elovici Y (2018) Generic Black-Box End-to-End Attack Against State of the Art API Call Based Malware Classifiers. arXiv 1707.05970
Shorten C, Khoshgoftaar T (2019) A survey on image data augmentation for deep learning. J Big Data 6(1):60
Singla A, Bertino E, Verma D (2019) Overcoming the lack of labeled data: Training intrusion detection models using transfer learning. In: 2019 IEEE International Conference on Smart Computing (SMARTCOMP). IEEE, Washington. pp 69–74. https://ieeexplore.ieee.org/document/8783997
Stehman S (1997) Selecting and interpreting measures of thematic classification accuracy. Remote Sens Environ 62(1):77–89
Sutskever I, Vinyals O, Le Q (2014) Sequence to sequence learning with neural networks. In: Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2 NIPS'14. MIT Press, Cambridge, MA, USA. pp 3104–3112
Szegedy C, Zaremba W, Sutskever I, Bruna J, Erhan D, Goodfellow I, Fergus R (2014) Intriguing properties of neural networks
Ustebay S, Turgut Z, Aydin M (2018) Intrusion detection system with recursive feature elimination by using random forest and deep learning classifier. In: 2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT). IEEE, Ankara. pp 71–76. https://ieeexplore.ieee.org/document/8625318/
Wang X, Huang Q, Celikyilmaz A, Gao J, Shen D, Wang Y-F, Wang W, Zhang L (2019) Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. IEEE, Long Beach. pp 6629–6638. https://ieeexplore.ieee.org/abstract/document/8953608
Wei J, Zou K (2019) Eda: Easy data augmentation techniques for boosting performance on text classification tasks. arXiv preprint arXiv:1901.11196:6382–6388. https://www.aclweb.org/anthology/D19-1670.bib
Xu C, Shen J, Du X, Zhang F (2018) An intrusion detection system using a deep neural network with gated recurrent units. IEEE Access 6:48697–48707
Yu L, Zhang W, Wang J, Yu Y (2019) Seqgan: Sequence generative adversarial nets with policy gradient. In: Thirty-First AAAI Conference on Artificial Intelligence. ACM, San Francisco. https://dl.acm.org/doi/10.5555/3298483.3298649

Publisher's Note