Multimedia Tools and Applications (2022) 81:7103–7124
https://doi.org/10.1007/s11042-022-12049-1

An automatic algorithm for software vulnerability classification based on CNN and GRU

Qian Wang[1] · Yazhou Li[2] · Yan Wang[3] · Jiadong Ren[1]

Received: 22 January 2021 / Revised: 24 May 2021 / Accepted: 3 January 2022 /
Published online: 24 January 2022

Abstract

In order to improve the management efficiency of software vulnerability classification, reduce the risk of system being attacked and destroyed, and save the cost for vulnerability repair, this paper proposes an automatic algorithm for Software Vulnerability Classification based on convolutional neural network (CNN) and gate recurrent unit neural network (GRU), called SVC-CG. It has conducted a fusion between the models of CNN and GRU according to their advantages (CNN is good at extracting local vector features of vulnerability text and GRU is good at extracting global features related to the context of vulnerability text). The merger of the features extracted by the complementary models can represent the semantic and grammatical information more accurately. Firstly, the Skip-gram language model based on Word2Vec is used to train and generate the word vector, and the words in each vulnerability text are mapped into the space with limited dimensions to represent the semantic information. Then the CNN is used to extract the local features of the text vector, and the GRU is used to extract the global features related to the text context. We combine two complementary models to construct a SVC-CG neural network algorithm, which can represent semantic and grammatical information more accurately to realize automatic classification of vulnerabilities. The experiment uses the vulnerability data from the national vulnerability database (NVD) to train and evaluate the SVC-CG algorithm. Through experimental comparison and analysis, the SVC-CG algorithm proposed in this paper has a good performance on Macro recall rate, Macro precision rate and Macro F1-score.

Keywords Neural network · Software security · Vulnerability classification

✉ Qian Wang
wangqianysu@163.com

[1] Computer Virtual Technology and System Integration Laboratory of Hebei Province, College of Information Science and Engineering, Yanshan University, Qinhuangdao 066000 Hebei, China

[2] China Mobile Xiong'an Information and Communication Technology Co. Ltd, Xiong'an 071700 Hebei, China

[3] Computing Center, Northeastern University at Qinhuangdao, Qinhuangdao 066000 Hebei, China

1 Introduction

With the rapid development of information technology, the Internet and computers have been deeply applied in various areas of people's daily life, including energy, transportation, production, military, medical care and so on, which bring us great convenience as well as risks and hidden dangers. Computer security incidents have occurred frequently in recent years and leaded serious consequences, like confidential information leakage and loss of data privacy, which harm the interests of enterprises, organizations and individuals. However, computer security incidents are usually caused by hostile attacks via the vulnerabilities in computer system. The number of vulnerabilities has grown rapidly over the past decade. According to the statistics of the National Vulnerability Database (NVD) [21] from 2002 to 2019, the total number of vulnerabilities has reached 121,279, including 38,868 vulnerabilities with unknown categories. Therefore, the security issue focuses on the vulnerability analysis, which has been widely concerned, becomes a hot topic of scientific research. The annual distribution of vulnerabilities from 2002 to 2019 is shown in Fig. 1.

Facing such a large number of computer vulnerabilities, the limitations of traditional artificial vulnerability classification methods become more and more obvious, so it is particularly important to classify and manage vulnerabilities effectively. Attentions have already been paid to the automatic classification model of vulnerability. Hovsepyan et al. [11] proposed a novel vulnerability prediction method for software components, this method combined machine learning and text analysis technology to analyze the source code of software vulnerability, and obtained good results. Wijayasekara et al. [27] tested Naïve Bayes classifier by using text information from false descriptions, the analysis showed the feasibility of the Naïve Bayes classifier for classifying text information based on the vulnerability description. Sarang et al. [20] proposed a classification method that used Naïve Bayes classifier to classify CVE entries as vulnerability categories. The CVE entries that could not provide enough information were studied, and the unclassified CVE entries and unclassified

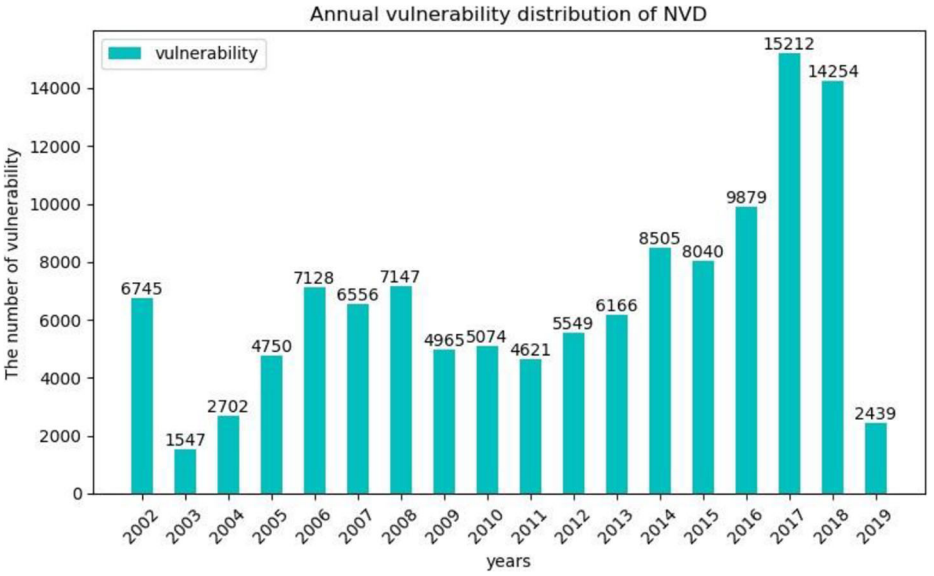


Fig. 1 Annual vulnerability distribution of NVD (2002–2019)

vulnerability documents were better analyzed. Davari et al. [5] proposed an automatic vulnerability classification framework based on activation vulnerability conditions. It used different machine learning techniques (Random Forest, C4.5 Decision Tree, Logistic Regression, and Naïve Bayes) to build the classifier with the highest F1-score to mark unknown vulnerabilities, and 580 software security defects in the Firefox project were analyzed experimentally to evaluate the effectiveness of classification. Marian et al. [8] applied Naïve Bayes algorithm and simple artificial neural network algorithm to vulnerability classification, the experimental results showed that the artificial neural network algorithm was better than Naïve Bayes algorithm in vulnerability classification. A data-driven approach [9] for vulnerability detection using machine learning was proposed by Harer et al., this paper compared the application of deep neural network model with more traditional models such as random forest and found that the best performance came from the combination of deep learning model and tree-based model. Siewruk et al. [23] proposed a software vulnerability classification system based on context awareness, which relied on machine learning to automatize the whole process. The experimental results obtained on a real-life dataset collected from one of the major mobile network operators in Poland proved that the proposed solution is useful and effective. Bhuiyan et al. [2] proposed a mapping study of publications that used security bug reports could inform researchers on software vulnerabilities. It concluded that findings from the mapping study could be leveraged to identify research opportunities in the domains of software vulnerability classification and automated vulnerability repair techniques. Aota et al. [1] proposed a scheme that automatically classifies each vulnerability description by type using machine learning. They experimentally demonstrated the performance of the proposed scheme in comparison with other algorithms, analyzed cases of misclassification, and revealed the potential for numerous human errors. Furthermore, they tried to correct these errors. It can be seen that machine learning algorithms are increasingly applied in the field of vulnerability classification to achieve automatic classification of vulnerabilities.

In this paper, vulnerabilities are classified according to text information in vulnerability data. However, the traditional text representation method [25] using word Vector Space Model (VSM) makes the generated word vector space high-dimensionally and sparsely, which lead to low accuracy of vulnerability classification. In recent years, deep learning [22] has developed rapidly and applied successfully in various fields such as image recognition, speech recognition [3, 29] and natural language processing [7]. Meanwhile, it has also been applied in the field of software vulnerability detection and achieved good performance. Wu F et al. [28] proposed a deep learning method for vulnerability detection, deep learning methods such as convolutional neural network (CNN) and long-term short-term memory (LSTM) were used to predict vulnerabilities, which provided a prediction accuracy of 83.6% better than traditional methods. Li Z et al. [17] designed and implemented a vulnerability detection system based on deep learning, which lightened the burden on experts, who no longer needed to define features manually. G Huang et al. [12] proposed a software vulnerability automatic classification method based on deep neural network. This method was based on TF-IDF, information gain and deep neural network to build an automatic vulnerability classifier. With the National Vulnerability Database (NVD) as its experimental data, the accuracy of vulnerability classification reached 87%, which indicated that the model had a high classification performance. In the experiment, although the multi-layer neural network was used to extract features, the traditional word vector model was still used to represent text word vectors, so, there remains the high-dimensional and sparse problems.

Aiming at the problem that traditional machine learning algorithms do not perform well on high-dimensional and sparse features of vulnerability text, and some specific vulnerability information is usually missed, this paper proposes an automatic algorithm for Software Vulnerability Classification based on convolutional neural network (CNN) and gate recurrent unit neural network (GRU), called SVC-CG. It has conducted a fusion between the models of CNN and GRU according to their advantages (CNN is good at extracting local vector features of vulnerability text and GRU is good at extracting global features related to the context of vulnerability text). The merger of the features extracted by the complementary models can represent the semantic and grammatical information more accurately. In the algorithm, the Skip-gram language model is firstly used to train and generate vulnerability word vectors, and the words in each vulnerability text are mapped to finite dimensions to represent semantic information. Then, by making full use of the advantages of CNN and GRU neural network in extracting features and representing semantic information, the SVC-CG neural network algorithm is constructed. The experiment uses the vulnerability data from the national vulnerability database (NVD) to evaluate the SVC-CG algorithm. Experimental results show that the SVC-CG algorithm proposed in this paper has a great improvement in Macro recall rate, Macro precision rate and Macro F1-score.

The rest of this paper is organized as follows. Section 2 mainly introduces the vulnerability data in the national vulnerability database (NVD), the application and development of deep learning in the field of natural language processing. In section 3, the Skip-gram language model and the SVC-CG neural network algorithm proposed in this paper are described in detail. The performance of SVC-CG algorithm is evaluated through vulnerability classification experiment in Section 4. Section 5 outlines the conclusions and future work.

2 Related work

Convolutional neural network [16] is widely used in image recognition. The Recurrent Neural Network (RNN) [6] proposed by Elman et al. was used to deal with sequence problems, while the problem of gradient disappearance or gradient dispersion occurred as the hidden layer deepens. In order to solve the problem in traditional RNN, Hochreiter S et al. [10] proposed a long-short-term memory neural network (LSTM) by improving the structure of the hidden layer. However, LSTM had a complex internal structure with many training parameters, which reduced the training efficiency of the model. So, Chung et al. [4] proposed a gated recurrent unit structure (GRU) to simplify the hidden layer structure of LSTM and improved the training efficiency of the model. At the same time, the problem of gradient disappearance or gradient dispersion was avoided.

In recent years, neural network models based on deep learning have achieved great success in the field of natural language processing (NLP), including distributed representation of words, sentence modeling based on convolutional neural networks and text classification based on recurrent neural networks. Yih et al. [30] proposed a novel training method for semantic discrimination, which was to learn the representation of word vectors and perform classification tasks on the combination features of the trained word vectors. In 2013, Tomas Mikolov [19] proposed a distributed vector representations method to train high-quality distribution vectors by training word vectors in continuous Skip-gram model to represent a large number of precise syntactic and semantic relations. Yoon Kim [14] proposed a sentence classification method based on convolutional neural network in 2014. The paper focused on

sentence-level classification tasks, using convolutional neural networks (CNN) combined with trained word vectors to do a series of experiments, and achieved very good performance on the emotional classification benchmark dataset. Kalchbrenner et al. [13] designed a Dynamic Convolution Neural Network (DCNN) model to process text of different lengths. Liu P et al. [18] proposed a text classification method based on recurrent neural network. In this paper, a multi-task learning framework was used to learn multiple related tasks together, three different shared information mechanisms were proposed to simulate specific tasks. The text of the layer was shared and experiments were carried out on the four benchmark text classification data sets. The experimental results showed a good classification performance on the model.

The description of vulnerability text is relatively short, and although some vulnerabilities have different categories, the text description is very similar. If the characteristics of these vulnerabilities cannot be well extracted, the possibility of classification error will be increased. The problem of high dimension and sparsity exist in the representation of vulnerability features will also increases the uncertainty of vulnerability classification. The main disadvantage of the common neural network structure is that it ignores the structural characteristics of the input data. No matter what type of data structure, it needs to be converted into a single vector before it is fed to the neural network, and then the single vector is used as the input of the neural network. However, for the structure of matrix, on the one hand, the structural characteristics of matrix will be ignored, on the other hand, the number of weights will be too large. As a result, a large number of parameters will generate in the training process of the model. These parameters are used to adjust the weight values, which results in the complexity of the model structure and the efficiency reduction of the training process. The CNN structure not only considers the structural characteristics of the data when processing the input data, but also realizes the parameter sharing through the sliding setting of the convolution kernel, which greatly reduces the complexity of the model during training. Therefore, we introduce CNN to process the data of matrix structure, which not only retains the structural characteristics of the input data, but also reduces the problem of excessive parameters during training. In traditional neural networks, we assume that the input data is independent of each other, but for many tasks, it is not. For example, if we want to predict the probability of the next word in a text sequence, we need to know what the words appear before it. The GRU structure can deal with sequence data well, and avoids the problem of gradient disappearance as the sequence increases in the traditional recurrent neural network, and also improves the training efficiency of the model, so we introduced the GRU structure to deal with the vulnerable text sequence.

In this paper, a SVC-CG algorithm based on CNN and GRU is proposed. The CNN is used to extract the local features of text vectors. The GRU is used to extract the global features related to the text context. The features extracted by the two complementary models are fused, so that they can represent more accurate semantic and grammatical information.

3 The proposed algorithm

3.1 Distributed representation of words

In this paper, the Skip-gram language model provided by Word2Vec is used to train and generate word vectors. Words in each text are mapped to a space with finite dimensions. In this space, the relationship between words and their positions is realized through mapping relations

to reflect their semantic relations. The Skip-gram language model trains the words by unsupervised way to obtain the distributed feature presentation of the words, namely word embedding. All the word embedding longitudinal stacks of each text are utilized to obtain two-dimensional text feature matrix. The Skip-gram language model structure is shown in Fig. 2.

The Skip-gram model mainly includes input layer, projection layer, and output layer, and its basic principle is to input the vector of w_n to predict word vectors of w_{n-2}, w_{n-1}, w_{n+1}, w_{n+2}.

Input layer words are word vectors encoded by one-hot, given a series of training words (w_1, w_2, w_3, …, w_N), the optimal objective function of the model takes the average log-likelihood function, and the formula is as follows.

$$L = \frac{1}{N}\sum_{n=1}^{N}\sum_{-s\le j\le s, j\neq 0} \log p\left(w_{n+j}|w_n\right) \tag{1}$$

where, s is the number of training words, N is the number of words in the vocabulary, and w_n is the central word.

The output layer uses the softmax function to figure out the probability of each word, which represents the probability that each word in the dictionary appears simultaneously with the input word. The formula is as follows.

$$p(w_O|w_I) = \frac{\exp\left(v'^{T}_{w_O} v_{w_I}\right)}{\sum_{n=1}^{N}\exp\left(v'^{T}_{w_n} v_{w_I}\right)} \tag{2}$$

where v_{w_I} and v'_{w_O} are vector representations of the input and output of the words.

Mikolov et al. [19] proposed two optimized methods of word vectors for the distributed representation methodology of words and phrases, namely Hierarchical Softmax (HS) and

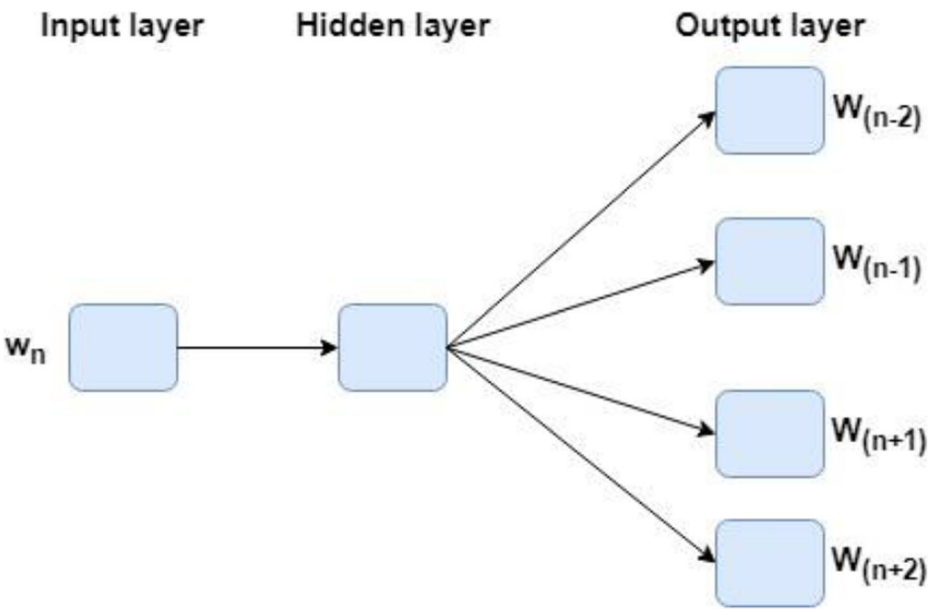


Fig. 2 General structure diagram of Skip-gram

Negative Sampling (NS), and conducted comparative experiments. The HS method needs to calculate the output probability of each word and make normalization, which leads to a large amount of calculation and lowers the training efficiency of word vectors. The NS method only updates a small part of the model weight during training process, which reduces the calculation load. The experiment proved that the NS method is better than the HS method in the training efficiency of word vectors, so NS method is used here. The main idea of NS is to select partial negative samples (non-target feature words) to update the relevant parameters of the positive samples (target feature words). Its objective function is as follows.

$$E = -\log\sigma\left({v'_{w_O}}^T h\right) - \sum_{w_n \in S_{neg}} \log\sigma\left(-{v'_{w_n}}^T h\right) \tag{3}$$

where w_O refers to the input positive sample, v'_{w_O} is the output vector of the word w_O, $h \in R^N$ is the value of the hidden layer, and S_{neg} is a negative sample set extracted arbitrarily. The optimization goal of the objective function is to maximize the probability of positive samples and minimize the probability of negative samples.

3.2 SVC-CG

In this paper, CNN is used to extract the local features of the vulnerability text and GRU is used to extract the global features related to the context of the vulnerability text. The SVC-CG model was proposed by combining the features extracted from the two complementary structures. The convolutional layer of the SVC-CG model is different from the traditional CNN. The convolution layer of traditional CNN uses the same width and height of the convolution kernel, but the width of the convolution kernel of the SVC-CG model is consistent with the dimension of the word vector. This is because each line of vector in our input represents a word in the vulnerability text. In the process of extracting features from the vulnerability text, the word is used as the minimum granularity of the vulnerability text. The input of the SVC-CG model is a sentence vector matrix, and the correlation degree between adjacent words in the sentence will be very high. Therefore, when we perform convolution calculations, we consider not only the information of the words in the vulnerability text but also the order of words and the structure of their context, which can extract the local features of the input data well. The SVC-CG algorithm mainly includes input layer, convolution layer, maximum pooling layer and GRU layer. The feature extractor composed of convolution layer and pooling layer plays an important role in the whole network structure, especially in simplifying the complexity of network structure and reducing the number of model parameters. The structure of SVC-CG model is shown in Fig. 3.

The input data of the input layer is a word vector matrix based on Word2Vec training. Firstly, the word w_n is converted into the corresponding word vector $v(w_n)$ by using Word2Vec, and the sentence composed of the word w_n is mapped to the sentence matrix s_j, which is used as the vector matrix of the embedding layer of CNN. The sentence matrix is expressed as follows.

$$S_j = \{v(w_1), v(w_2), \ldots, v(w_N)\} \tag{4}$$

where, $v(w_n) \in R^k$, $s_j \in R^{m * k}$, k represents the dimension of the word vector, m represents the number of sentences.

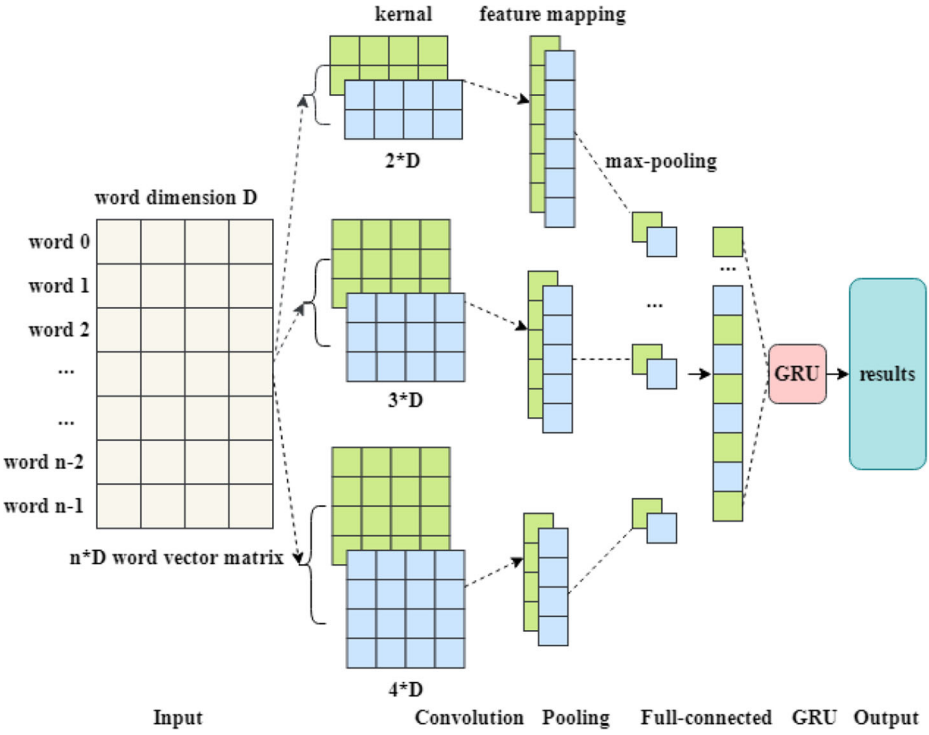


Fig. 3 The structure of SVG-CG model

The Convolution Layer can well describe the local features of the input data. In this paper, a filter with the size of $r * k$ is used to carry out convolution operation on sentence matrix s_j to extract local features of s_j. The formula is as follows.

$$c_i = f((r*k) \bullet x_{i:i+r-1} + b) \tag{5}$$

where, f represents a nonlinear activation function that uses the Relu activation function [15], $x_{i:i+r-1}$ represents r vectors from the i^{th} vector to the $(i + r - 1)^{th}$ vector in s_j. b is the bias vector, c_i is the local feature obtained by convolution operation.

The filter slides from the top to the bottom of s_j with a step size of 1, and finally gets the vector set c of local features.

$$c = [c_1, c_2, , c_{N-r+1}] \tag{6}$$

The max-pooling is the most important part of extracting local features based on the convolution operation. The size of the feature vector can be greatly reduced by the pooling operation. This paper uses the method of max-pooling to extract the feature with maximum value to replace the whole local feature.

$$\hat{c} = \max(c) \tag{7}$$

GRU is a variant of LSTM that makes the structure simpler and iteration faster while preserving the LSTM effect. The GRU consists of two gated structures, the reset gate r_t and the update gate z_t. Its structure is shown in Fig. 4.

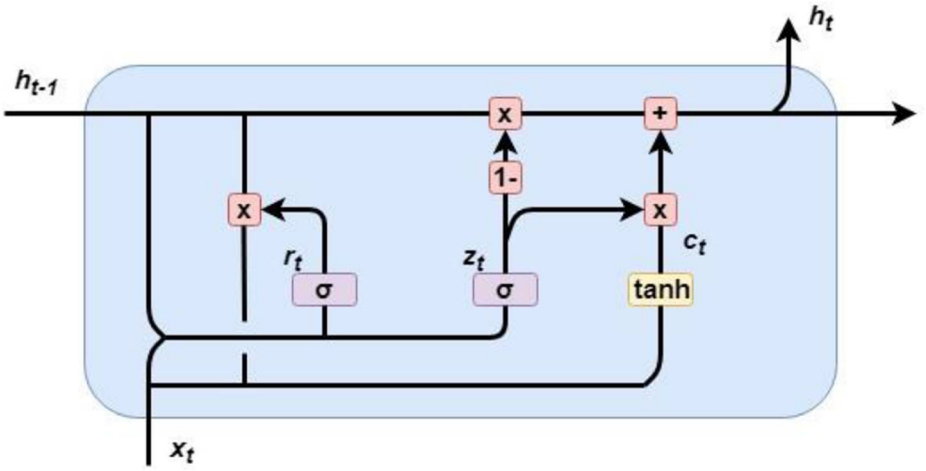


Fig. 4 Structure diagram of GRU

The specific calculation formula of the internal structure of GRU is as follows.

$$r_t = \sigma(x_t U_r + h_{t-1} W_r + b_r) \tag{8}$$

$$z_t = \sigma(x_t U_z + h_{t-1} W_z + b_z) \tag{9}$$

$$c_t = \tanh(x_t U_c + r_t \odot h_{t-1} W_c + b_c) \tag{10}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot c_t \tag{11}$$

$$y_t = \sigma(W_o h_t + b_o) \tag{12}$$

where, h_{t-1} is the output of the hidden layer at time $t-1$, x_t is the input at time t, h_t is the output of the hidden layer at time t, r_t represents reset gate, z_t represents update gate, c_t represents the candidate value of the hidden layer node at time t, y_t represents output, U and W represent the parameter matrix, b represents the offset vector, $\odot$ represents matrix element multiplication, σ represents the sigmoid activation function.

In order to convert the output layer into a probability distribution of the classification category, we add the softmax layer, and the calculation formula is as follows.

$$p_j = \frac{\exp\left(y_j\right)}{\sum_{a=1}^{J} \exp(y_a)}, j = 1, 2, , J \tag{13}$$

where, y_j represents the output of category j, p_j represents the probability of category j, and J represents the total number of categories.

We use the cross entropy loss function to calculate the loss of the model. The calculation formula of the cross entropy loss is as follows.

$$L\left(\hat{y}^{(i)}, y^{(i)}\right) = \sum_{i=1}^{n} y^{(i)} \log \hat{y}^{(i)} + \left(1 - y^{(i)}\right) \log\left(1 - \hat{y}^{(i)}\right) \tag{14}$$

where, $y^{(i)}$ is the real label of the sample i, and $\hat{y}^{(i)}$ is the predicted output.

4 Experiment

The vulnerability classification experiments in this paper mainly include three parts: vulnerability data preprocessing, word vector generation and SVC-CG algorithm. The experimental flow chart is shown in Fig. 5. Firstly, we carry out preprocessing operations on NVD vulnerability database, such as special character removal, word splitting, lemmatization and stop word filtering. Then, we used the skip-gram language model and the distributed representation method of the word optimized by negative sampling to train the word vector of the preprocessed NVD vulnerability database, and divided the training set and the test set. Finally, we conducted parameter training on the SVG-CG model proposed in this paper to achieve a better vulnerability classification effect.

4.1 Experimental environment

The experiment was conducted on PC with Intel(R) Core (TM) i5–4460 processor, 3.20 GHz and 8.00 GB memory, running Windows 7 operating system. Experiments were performed using the Python 3.6 programming language and the Keras/TensorFlow deep learning framework.

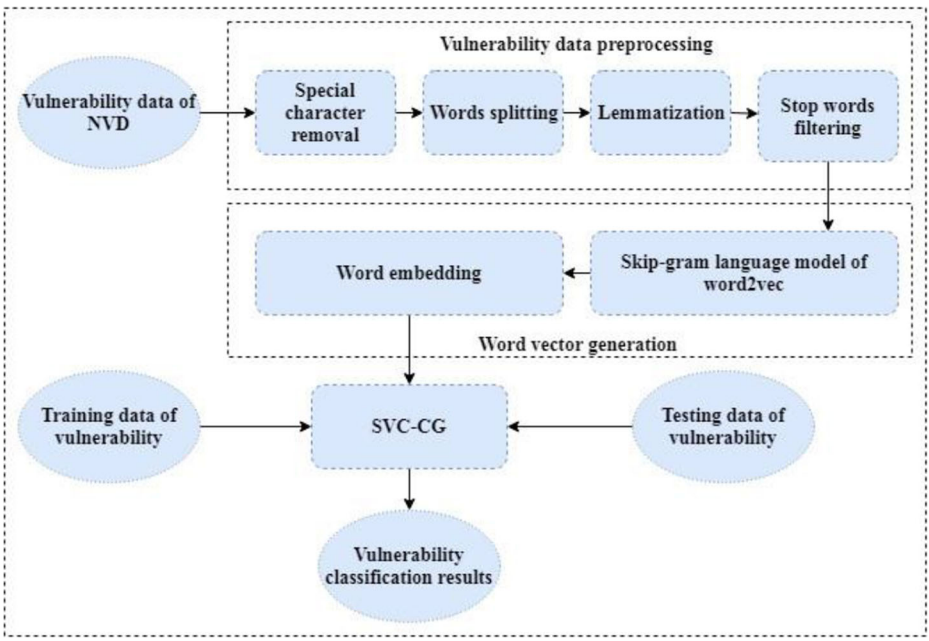


Fig. 5 Experimental flow chart

4.2 Data set of experiment

This paper uses vulnerability data from the NVD. The vulnerability data collected by NVD is normative with the international coding grammar standard. Therefore, it can be used as the benchmark data set for software vulnerability classification research. NVD provides both XML and JSON vulnerability files. This paper uses a XML format of vulnerability file, which contains vulnerability information such as CVE-ID, CVSS_score, CVSS_access, CVSS_vector, vuln-source, CWE-ID and vuln-summary. According to the vulnerability information required by the vulnerability classification experiment, this paper uses the Python programming language to extract CVE-ID, CWE-ID and vuln-summary data from the XML vulnerability file. Unrelated fields and incomplete data will be deleted. Some of the extracted vulnerability information is shown in Table 1, where CVE-ID represents the number of vulnerability, and CWE-ID represents the vulnerability type. The specific vulnerability categories can be obtained according to CWE standard [26], and vuln-summary is the detailed information of vulnerability.

This paper selected 43,496 NVD vulnerability data from 2002 to 2019 for experimental research, including 16 vulnerability categories. The statistical quantity distribution of different vulnerability categories is shown in Fig. 6.

According to the CWE standard, the categories of vulnerabilities corresponding to each CWE-ID are shown in Table 2.

In order to verify the vulnerability classification performance of the SVC-CG algorithm proposed in this paper, we use the NVD vulnerability data from 2002 to 2019 as an experimental data evaluation model. The total number of NVD vulnerability data used in this experiment is 43,496, including 16 vulnerability categories. The experimental data is divided into training set and test set, including 32,262 training sets and 10,874 test sets.

4.3 Data preprocessing

Before performing the vulnerability classification task, a series of pre-processing operations need to be done on the vulnerability text data as below.

1) Remove punctuation and special characters

The original vulnerability text contains many punctuation marks and special characters, but these elements are not semantically related to the context. Therefore, all the punctuation marks

Table 1 Examples of NVD

CVE-ID	CWE-ID	vuln-summary
CVE-2019-9961	CWE-79	A cross-site scripting (XSS) vulnerability in ressource view in core/modules/-resource/RESOURCEVIEW.php in Wikindx prior to version 5.7.0 allows remote attackers to inject arbitrary web script or HTML via the id parameter.
CVE-2019-9962	CWE-119	XnView MP 0.93.1 on Windows allows remote attackers to cause a denial of service (application crash) or possibly have unspecified other impact via a crafted file, related to VCRUNTIME140!memcpy.
…	…	…

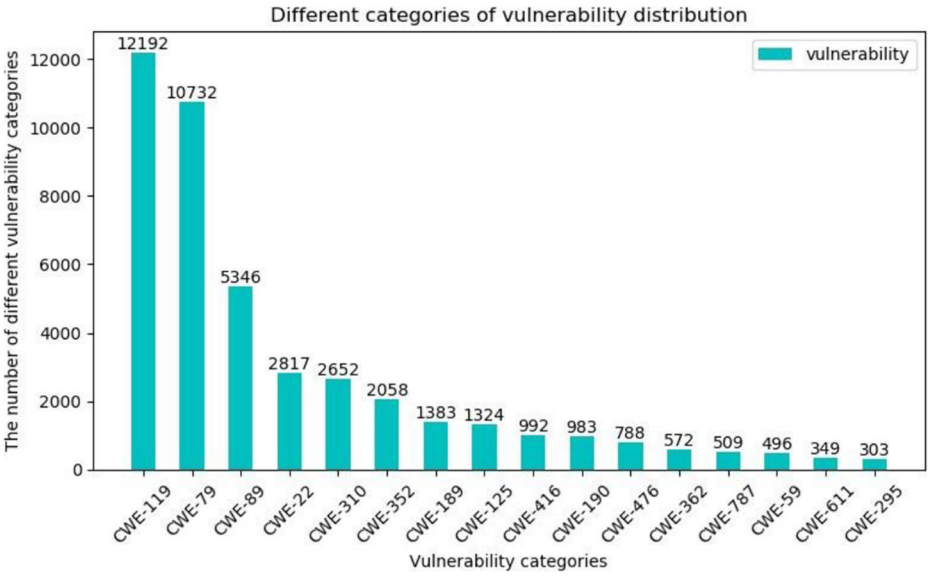


Fig. 6 Vulnerability distribution of different categories

and special characters in the text need to be filtered first, and only the words with more semantic information are retained.

2) Word segmentation and conversion of characters from uppercase to lowercase

Word segmentation of vulnerability text refers to the segmentation of coherent vulnerability text information into words, which converts the whole vulnerability text information into the smallest semantic unit that can be counted by statistics. For the vulnerability text described in English, word segmentation is very simple. The whole vulnerability text can be divided into

Table 2 Vulnerability type according to CWE standard

CWE-ID	Vulnerability type
CWE-119	Buffer Errors
CWE-79	Cross-Site Scripting (XSS)
CWE-89	SQL Injection
CWE-22	Path Traversal
CWE-310	Cryptographic Issues
CWE-352	Cross-Site Request Forgery (CSRF)
CWE-189	Numeric Errors
CWE-125	Out-of-bounds Read
CWE-416	Use After Free
CWE-190	Integer Overflow or Wraparound
CWE-476	Null Pointer Dereference
CWE-362	Race Conditions
CWE-787	Out-of-bounds Write
CWE-59	Link Following
CWE-611	XXE
CWE-295	Improper Certificate Validation

words by identifying spaces or punctuation marks. Then convert all the uppercase letters in the word to lowercase letters.

3) Lemmatization

Lemmatization refers to the transformation of non-root form into root form in the word set, and that is the verb in English description changing according to the person into the verb prototype. Convert the plural form of a noun to the singular form, convert gerund form to verb prototype, etc. From the perspective of data mining, these words should belong to the same category of semantically similar words.

4) Stop word filtering

Stop word filtering refers to words that appear frequently in the vulnerability text and contribute little to the classification of text information. For example, common prepositions, articles, auxiliary words, modal verbs, pronouns, and conjunctions are meaningless to the vulnerability classification, so these words should be filtered out. In this paper, the English general stop word list [24] is used to filter stop words for vulnerability information.

4.4 Word vector generation

In this paper, word distributed representation method based on the Skip-gram language model in word2vec is used to train and generate word vectors in the experimental data set of vulnerability. Meanwhile, negative sampling method is used to optimize the Skip-gram model, so as to improve the training efficiency of word vectors. The experiment uses 43,496 NVD vulnerability data as the word vector training corpus, the word vector dimension is set to 300, the words with the word frequency less than 10 in the corpus are removed, and the number of samples when using negative sampling for optimization is set to 5.

4.5 SVC-CG algorithm settings

In the experiment, SVC-CG algorithm mainly includes an input layer, a convolution layer, a maximum pooling layer, two GRU layers and an output layer. In the input layer, the preprocessed vulnerability text is taken as the input with a 50*300 matrix of word vector. The convolution layer is a 3*300 matrix, and the downward moving step size is 1. The input of the convolution layer is also a 50*300 matrix, and the output is a 50*256 matrix. Relu activation follows the convolution layer. The input of the max-pooling layer is a 50*256 matrix, the output is a 25*256 matrix, and the size of the pooling window is set to 2. The number of neurons in the two GRU layers is set to 256, the output layer is set to 16, which represents 16 types of vulnerabilities. Figure 7 shows the detailed structure of the SVG-CG model.

The model of SVG-CG uses Dropout for over-fitting, the Dropout value is set to 0.5. The softmax is used as the activation function of the output layer, the cross entropy loss function is used to calculate the loss of the model, and the Adam optimizer is used to minimize the loss function. The batch size of model training is set to 800, and the number of iterations is set to 25. The parameter settings in this paper are based on Reference [31] and grid search method.

4.6 Evaluation index

In this paper, according to the confusion matrix and the multi-classification, the precision rate (P_i), recall rate (R_i) and F1-score ($F1_i$) are adopted to evaluate the classification performance of

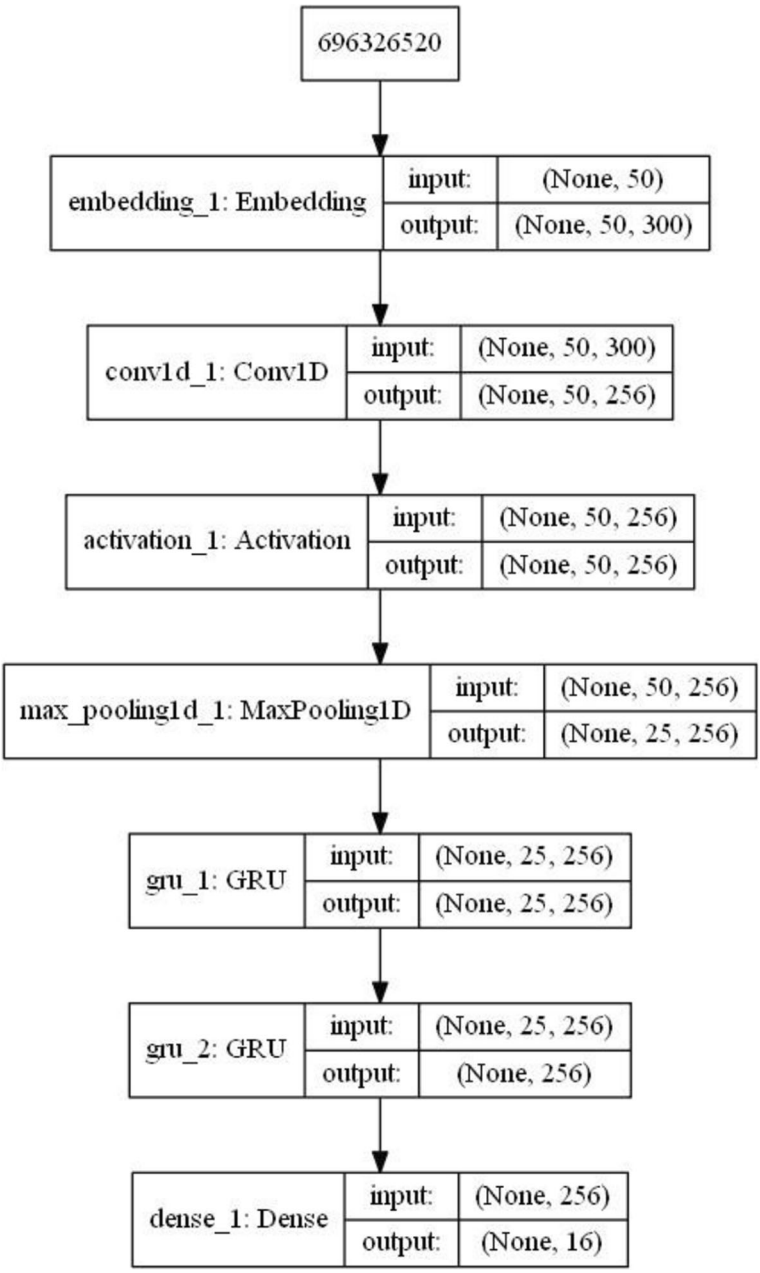


Fig. 7 Algorithm setting of SVC-CG

the SVC-CG algorithm, where i represents the category ID of the vulnerability. The confusion matrix is shown in Table 3.

For the vulnerabilities of Category i, TP_i indicates that the actual category of the sample is positive and predicted as a positive one. TN_i indicates that the actual category of the sample is negative and predicted as a negative one. FP_i indicates that the actual category of the sample is negative and predicted as a positive one, FN_i indicates that the real category of the sample is positive and predicted as a negative one. P_i indicates the ratio of the number of positive samples correctly classified to the number of samples classified as positive ones, R_i indicates the ratio of the number of the positive samples correctly classified to the number of actual positive samples, $F1_i$ represents the harmonic mean of P_i and R_i. The calculation formula of each indicator is as follows.

$$P_{\mathrm{i}} = \frac{TP_i}{TP_i + FP_i} \tag{15}$$

$$R_i = \frac{TP_i}{TP_i + FN_i} \tag{16}$$

$$F1_i = \frac{2*P_i*R_i}{P_i + R_i} \tag{17}$$

In order to comprehensively evaluate the classification performance of SVC-CG, *Macro P*, *Macro R* and *Macro F*1 are finally taken as the comprehensive evaluation indexes. Macro index is generally used as a comprehensive evaluation index for multi-classification problems. It refers to the arithmetic average value of the statistical index value of each category in all categories, which reflects the comprehensive performance of SVC-CG in the problem of multi-classification of vulnerabilities. The calculation formulas are as follows.

$$Macro\ P = \frac{1}{n}\sum_{i=1}^{n} P_i \tag{18}$$

$$Macro\ R = \frac{1}{n}\sum_{i=1}^{n} R_i \tag{19}$$

$$Macro\ F1 = \frac{2*Macro\ P*Macro\ R}{Macro\ P + Macro\ R} \tag{20}$$

where, n represents the total number of categories.

Table 3 Confusion matrix

CWE-ID	Predicted category	
	Positive	Negative
Positive	TP	FN
Negative	FP	TN

4.7 Results and analysis

In order to verify the vulnerability classification performance of the SVC-CG algorithm proposed in this paper, we use the vulnerability training set to train the SVC-CG, and then use the test set to evaluate its classification performance. The change in training loss of the SVC-CG is shown in Fig. 8.

As seen from Fig. 8, when the SVC-CG iterates for 20 times, the loss value drops to a relatively low stable value, and the model achieves a good convergence effect. In the iteration process, as the number of iterations of model training increases, the classification performance of the model on the test set changes as shown in Fig. 9.

It can be seen from Fig. 9 that with the increase of the number of iterations, the model classification of *Macro P*, *Macro R* and *Macro F*1 is gradually improved. When the iteration reaches 15 times, the classification performance of the model tends to be stable and at a relatively high value. We use the vulnerability test set to classify and test the trained SVC-CG, and the multi-class confusion matrix of the detailed classification results of all vulnerability categories is shown in Fig. 10. The row represents the actual category and the column represents the predicted category. From the figure, we can see 16 types of vulnerabilities and their prediction results, including the numbers of correctly predicted type and wrongly predicted type. On the diagonal, they are the correctly predicted number, and others are wrongly predicted numbers for the corresponding vulnerability type. The sum of all the numbers in each line represents the total number of corresponding vulnerability types.

In order to evaluate the classification performance of the SVC-CG proposed in this paper for different vulnerability categories, we respectively obtain the P_i, R_i and $F1_i$ of SVC-CG for different vulnerability categories. The results are shown in Table 4.

From the results in Table 4, we can see that the performance of SVC-CG for the overall classification of the vulnerability has reached a relatively high level, but the classification

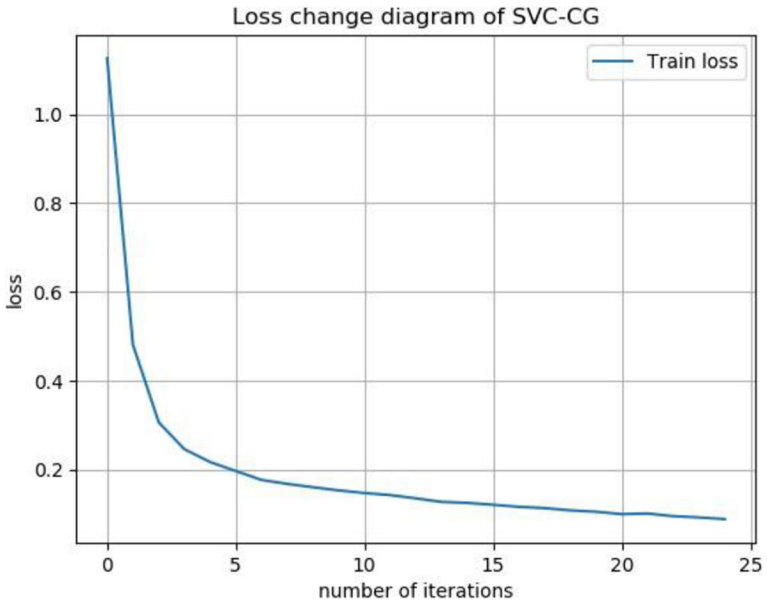


Fig. 8 Loss change diagram of SVC-CG

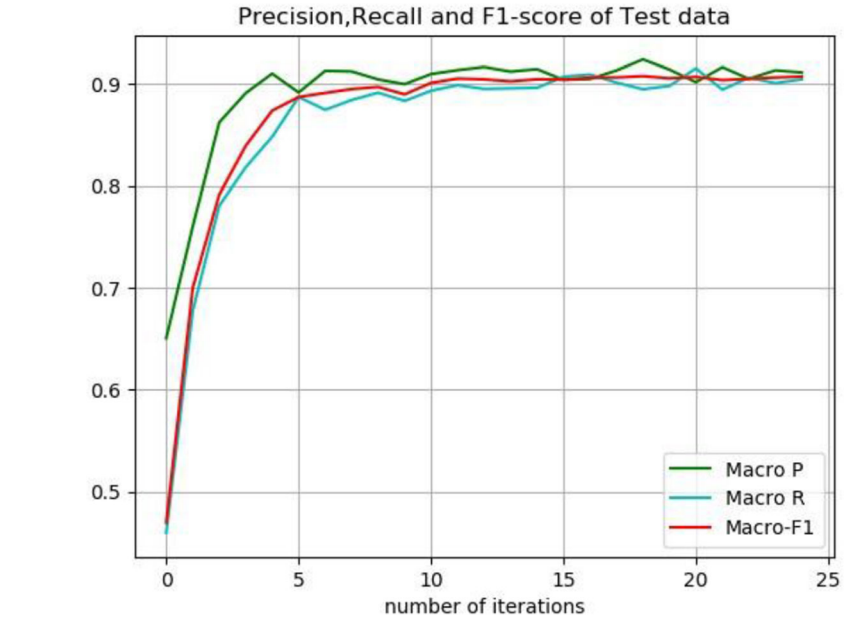


Fig. 9 Changes of classification performance of SVC-CG

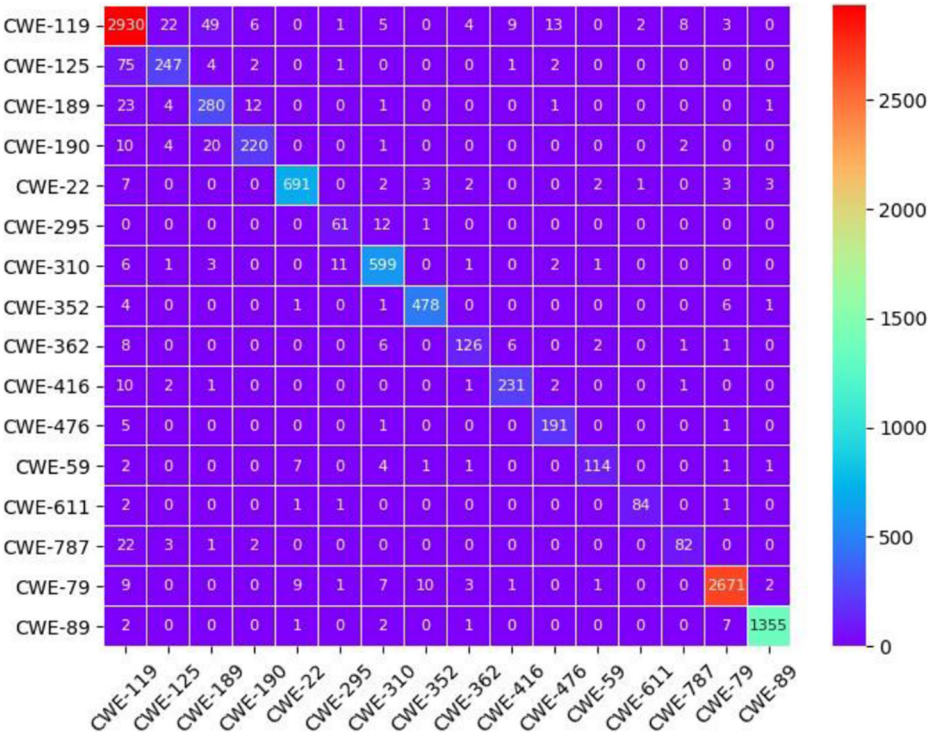


Fig. 10 Multi-class confusion matrix of the test data

Table 4 Vulnerability classification result of test data

Vulnerability type	Vulnerability number
CWE-119	3052
CWE-125	332
CWE-189	322
CWE-190	257
CWE-22	714
CWE-295	74
CWE-310	624
CWE-352	491
CWE-362	150
CWE-416	248
CWE-476	198
CWE-59	131
CWE-611	89
CWE-787	110
CWE-79	2714
CWE-89	1368

performance of different vulnerability categories is still different. The classification performances of the SVC-CG for CWE-79 and CWE-89 are the best in all vulnerability categories, and the *F*1 values reach 0.99. But for CWE-125 and CWE-787, the classification performances are relatively low, and the *F*1 value are 0.80.

There are many traditional machine learning algorithms, but not all of them can be used as baseline algorithms, because some of them are not applicable or low efficiency for text classification. SVM, Naive Bayes and KNN have been widely applied in the field of text classification and achieved good results, which are representative enough for baseline algorithms. Therefore, these algorithms are chosen for comparison. The experimental results show that due to the large number of vulnerability data and less description information of vulnerability text, the word vector space generated by the traditional machine learning algorithms are of high dimension and sparsity, resulting in low efficiency of vulnerability training and poor classification effect. Therefore, this paper proposes SVG-CG model based on previous studies. In order to comprehensively evaluate the performance of the SVC-CG proposed in the classification of vulnerabilities. The experimental results of the paper proposed by G Huang et al. are compared on the basis of ensuring that the vulnerability classification evaluation indicators and the number of training iterations of the model are consistent. The results on the indicators of *Macro P*, *Macro R* and *Macro F*1 are shown in Table 5.

According to the experimental comparison results, it can be seen intuitively that SVC-CG in this paper is far superior to other vulnerability classification algorithms in the three comprehensive evaluation indicators of *Macro P*, *Macro R* and *Macro F*1. The SVC-CG is

Table 5 Results of different classification algorithms

Algorithm	Macro P	Macro R	Macro F1
TFI-SVM	0.68	0.61	0.63
TFI-NB	0.78	0.72	0.77
TFI-KNN	0.78	0.76	0.77
TFI-DNN	0.82	0.85	0.81
SVC-CG	**0.92**	**0.90**	**0.91**

10%, 5%, and 10% higher than the TFI-DNN algorithm on the *Macro P*, *Macro R* and *Macro F*1 indicators. The comparison results show that the SVC-CG proposed in this paper has been greatly improved in automatic vulnerability classification and the result of SVG-CG model is better than that of the traditional baseline models.

5 Conclusion

In order to effectively improve the efficiency of vulnerability classification management and reduce the cost of vulnerability classification, this paper applies deep learning to software vulnerability classification, and proposes an automatic classification method of software vulnerability based on CNN and GRU neural network, called SVC-CG, which is based on Skip-gram language. The word distributed representation method well expresses the semantic and grammatical information between words, and also solves high dimension vector space and data sparseness problems. The SVC-CG vulnerability classification algorithm is compared with the TFI-SVM, TFI-NB, TFI-KNN and TFI-DNN vulnerability classification algorithms on the NVD vulnerability dataset. The results show that the SVC-CG has great improvement on *Macro P*, *Macro R* and *Macro F*1, and performs better than other vulnerability classification algorithms. The research work in this paper shows that SVC-CG has achieved good results in vulnerability classification and it also provides a basis for conducting relevant research on the benchmark vulnerability dataset.

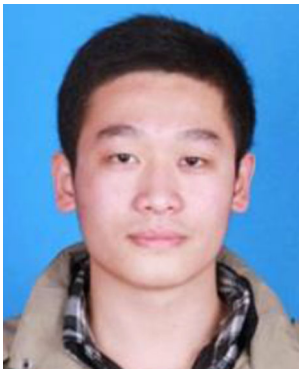
Acknowledgments This work is supported by the National Natural Science Foundation of China under Grant Nos. 61807028, 61802332, and 61772449, the Youth Foundation of Hebei Educational Committee of China under Grant No. QN2021145, the Natural Science Foundation of Hebei Province of China under Grant No. F2019203120. The Fundamental Research Funds for the Central Universities under Grant No.N182303036.The authors are grateful to valuable comments and suggestions of the reviewers.

References

1. Aota M, Kanehara H, Kubo M, Murata N, Sun B, Takahashi T (2020) Automation of vulnerability classification from its description using machine learning. 2020 IEEE Symposium on Computers and Communications (ISCC), pp 1–7
2. Bhuiyan FA, Sharif MB, Rahman A (2021) Security bug report usage for software vulnerability research: a systematic mapping study. IEEE Access 9:28471–28495
3. Chiu C-C, Sainath TN, Wu Y, Prabhavalkar R, Nguyen P, Chen Z, Kannan A, Weiss RJ, Rao K, Gonina E (2018) State-of-the-art speech recognition with sequence-to-sequence models. In Proc. of 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, pp 4774–4778
4. Chung J, Gulcehre C, Cho KH, Bengio Y (2014) Empirical evaluation of gated recurrent neural networks on sequence modeling. CoRR, abs/1412.3555, arXiv: 1412.3555
5. Davari M, Zulkernine M, Jaafar F (2017) An automatic software vulnerability classification framework. In Proc. of 2017 International Conference on Software Security and Assurance (ICSSA), IEEE, pp 44–49
6. Elman JL (1990) Finding structure in time. Cogn Sci 14(2):179–211
7. Gardner M, Grus J, Neumann M, Tafjord O, Zettlemoyer L (2018) AllenNLP: a deep semantic natural language processing platform. CoRR, abs/1803.07640, arXiv: 1803.07640
8. Gawron M, Cheng F, Meinel C (2018) Automatic vulnerability classification using machine learning. In Proc. of Risks and Security of Internet and Systems, Springer International Publishing, Cham pp 3–17

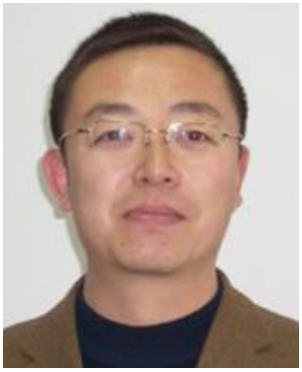
9. Harer JA, Kim LY, Russell RL, Ozdemir O, Kosta LR, Rangamani A, Hamilton LH, Centeno GI, Key JR, Ellingwood PM (2010) Automated software vulnerability detection with machine learning. CoRR, abs/1803.04497, arXiv: 1803.04497
10. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
11. Hovsepyan A, Scandariato R, Joosen W, Walden J (2012) Software vulnerability prediction using text analysis techniques. In Proc. of the 4th International Workshop on Security Measurements and Metrics, ACM, pp 7–10
12. Huang G, Li Y, Wang Q, Ren J, Cheng Y, Zhao X (2019) Automatic classification method for software vulnerability based on deep neural network. IEEE Access 7(1):28291–28298
13. Kalchbrenner N, Grefenstette E, Blunsom P (2014) A convolutional neural network for modelling sentences. CoRR, abs/1404.2188, arXiv: 1404.2188
14. Kim Y (2014) Convolutional neural networks for sentence classification. CoRR, abs/1408.5882, arXiv: 1408.5882
15. Krizhevsky A, Sutskever I, Hinton GE (2012) ImageNet classification with deep convolutional neural networks. In Proc. of the 25th International Conference on Neural Information Processing Systems - Volume 1, Curran Associates Inc, USA, pp 1097–1105
16. Lecun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324
17. Li Z, Zou D, Xu S, Ou X, Jin H, Wang S, Deng Z, Zhong Y (2018) VulDeePecker: a deep learning-based system for vulnerability detection. CoRR, abs/1801.01681, arXiv: 1801.01681
18. Liu P, Qiu X, Huang X (2016) Recurrent neural network for text classification with multi-task learning. CoRR, abs/1605.05101, arXiv: 1605.05101
19. Mikolov T, Sutskever I, Chen K, Corrado G, Dean J (2013) Distributed representations of words and phrases and their compositionality. Adv Neural Inf Proces Syst 26:3111–3119
20. Na S, Kim T, Kim H (2017) A study on the classification of common vulnerabilities and exposures using Naïve Bayes. In Proc. of Advances on Broad-Band Wireless Computing, Communication and Applications, Springer International Publishing, Cham, pp. 657–662
21. National Vulnerability Database [Online]. Available: http://nvd.nist.gov/vuln/data-feeds.
22. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M (2015) ImageNet large scale visual recognition challenge. Int J Comput Vis 115(3):211–252
23. Siewruk G, Mazurczyk W (2021) Context-aware software vulnerability classification using machine learning. IEEE Access
24. Stop Word List [Online]. Available: https://pypi.org/project/stop-words/
25. Turney PD, Pantel P (2010) From frequency to meaning: vector space models of semantics. J Artif Intell Res 37(1):141–188
26. Vulnerability Categories [Online]. Available: https://nvd.nist.gov/vuln/categories
27. Wijayasekara D, Manic M, McQueen M (2014) Vulnerability identification and classification via text mining bug databases. In Proc. of IECON 2014 - 40th Annual Conference of the IEEE Industrial Electronics Society, pp 3612–3618
28. Wu F, Wang J, Liu J, Wang W (2018) Vulnerability detection with deep learning. in Proc. of IEEE International Conference on Computer & Communications, IEEE, pp 1298–1302
29. Xiong W, Droppo J, Huang X, Seide F, Seltzer ML, Stolcke A, Yu D, Zweig G (2017) Toward human parity in conversational speech recognition. IEEE/ACM Trans Audio Speech Language Process 25(12): 2410–2423
30. Yih W-T, Toutanova K, Platt JC, Meek C (2011) Learning discriminative projections for text similarity measures. In Proc. of Proceedings of the Fifteenth Conference on Computational Natural Language Learning, Association for Computational Linguistics, pp 247–256
31. Zhang Y, Wallace B (2015) A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. Comput Sci, arXiv:1510.03820

Qian Wang received the B.S. degree, M.S. degree and Ph.D. degree in the school of Information Science and Engineering, Yanshan University, China, in 2009, 2012 and 2016, respectively. She has been in University of Hull from 2015 to 2016 as a visiting scholar. From 2016, she has become a lecture in the school of Information Science and Engineering, Yanshan University, China. She is now an associate professor. Her research interests include data mining, complex network and software security.



Yazhou Li received the B.S. degree from the School of Hebei normal university of science and technology, China, in 2016. He received the M.S. degree in the school of Information Science and Engineering, Yanshan University, China, in 2019. He is currently an engineering in China Mobile Xiong'an Information and Communication Technology Co. Ltd. His research interests include data mining, machine learning and software security.

Yan Wang received the M.S. degree in Computer Architecture in 2008, and the Ph.D. degree in Communication and Information System from the Northeastern University, in 2018. She is currently a Senior Experimenter of Computing Center. Her research interests include Mobile edge computing and 5G system design, software security. She is a member of China Computer Federation.



Jiadong Ren received his B.S. and M.S. degrees in Northeast Heavy Machinery Institute in 1989 and 1994. And he received his Ph.D. degree in 1999 from Harbin Institute of Technology. He is a professor in the school of Information Science and Engineering, Yanshan University, China. His research interests include data mining, complex network and software security. He is a senior member of the Chinese Computer Society, member of IEEE SMC Society and ACM.