Sun *et al. Journal of Cloud Computing: Advances, Systems and Applications* (2020) 9:53
https://doi.org/10.1186/s13677-020-00200-y

Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH Open Access

Categorizing Malware via A Word2Vec-based Temporal Convolutional Network Scheme

Jiankun Sun[1,2,3], Xiong Luo[1,2,3*†], Honghao Gao[4*†], Weiping Wang[1,2,3], Yang Gao[5] and Xi Yang[6]

Abstract

As edge computing paradigm achieves great popularity in recent years, there remain some technical challenges that must be addressed to guarantee smart device security in Internet of Things (IoT) environment. Generally, smart devices transmit individual data across the IoT for various purposes nowadays, and it will cause losses and impose a huge threat to users since malware may steal and damage these data. To improve malware detection performance on IoT smart devices, we conduct a malware categorization analysis based on the Kaggle competition of Microsoft Malware Classification Challenge (BIG 2015) dataset in this article. Practically speaking, motivated by temporal convolutional network (TCN) structure, we propose a malware categorization scheme mainly using Word2Vec pre-trained model. Considering that the popular one-hot encoding converts input names from malicious files to high-dimensional vectors since each name is represented as one dimension in one-hot vector space, more compact vectors with fewer dimensions are obtained through the use of Word2Vec pre-training strategy, and then it can lead to fewer parameters and stronger malware feature representation. Moreover, compared with long short-term memory (LSTM), TCN demonstrates better performance with longer effective memory and faster training speed in sequence modeling tasks. The experimental comparisons on this malware dataset reveal better categorization performance with less memory usage and training time. Especially, through the performance comparison between our scheme and the state-of-the-art Word2Vec-based LSTM approach, our scheme shows approximately 1.3% higher predicted accuracy than the latter on this malware categorization task. Additionally, it also demonstrates that our scheme reduces about 90 thousand parameters and more than 1 hour on the model training time in this comparison.

Keywords: Temporal Convolutional Network (TCN), Word2Vec, Internet of Things (IoT), Malware Categorization, Edge Computing

Introduction

Recent developments in the field of edge computing have led to extensive attention on smart device security in the Internet of Things (IoT) environment [1]. Nowadays, smart devices interact with networks for various purposes. A mass of personal information, including health condition, motion record, position data, is collected by smartphones and wearable devices and uploaded to service providers [2]. Meanwhile, with the continued increase in adoption, smart devices attract considerable attention from malware developers, and it poses great threats to IoT security [3]. For instance, it will cause information disclosure provided that inbuilt cameras and microphones in edge devices are taken over by malicious programs. More seriously, as smart chips in autonomous cars can navigate routes automatically, personal safety is hardly even guaranteed if the chips are attacked [4]. Hence, research

*Correspondence: xluo@ustb.edu.cn; gaohonghao@shu.edu.cn
†Xiong Luo and Honghao Gao contributed equally to this work.
[1]School of Computer and Communication Engineering, University of Science and Technology Beijing, 100083, Beijing, China
[2]Beijing Key Laboratory of Knowledge Engineering for Materials Science, 100083, Beijing, China
Full list of author information is available at the end of the article

on malware detection and categorization on IoT remains imperative and promising.

Malware detection and analysis have received extensive discussion, yet traditional approaches are not fully available on edge devices in the IoT environment. Certain traditional defend techniques applied to general desktop computing environments rely on pre-defined rule libraries. However, the portability of smart devices causes that they are not always connected to fixed and trusted networks, and thus perimeter-based defenses, including firewalls and intrusion detection, are not available for edge devices [5]. Moreover, as smart devices put more emphasis on real-time interaction, the corresponding malware identification requires faster response speed than on traditional platforms. Current malware identification for edge devices mainly relies on the malware signature databases from software distributors, yet this approach can not meet the demand of detecting the ongoing number of malware in edge computing paradigm. Research on automatic malware analysis techniques in the IoT environment is exceptionally urgent. In our previous works, to measure the stability of cyber-physical systems (CPSs) under malicious attacks, we developed a finite-time observer to estimate the state of the CPSs [6]. Then, we proposed a kernel learning algorithm to improve the malware detection performance on complex datasets with noise [7]. In addition to detection performance, memory footprint and response speed are also of enormous importance for current smart devices on IoT, and this poses higher requirements for edge malware analysis. In this article, we are committed to improving edge malware identification performance with low memory footprint and fast response speed.

As one of the most energetic technology companies, Microsoft has paid great enthusiasm into the IoT field, and Windows-based applications have been well-developed via their Azure IoT platform services [8]. Focused on the Windows-based malware invasion problem on the IoT platform, this article proposes a malware categorization scheme for attributing malware into different families through a Word2Vec-based temporal convolutional network (TCN). The model performance is evaluated by comparing with several representative works, i.e., Naive Bayes Classifier, OneHot-based TCN, Word2Vec-based long short-term memory (LSTM), on the Microsoft Malware Classification Challenge (BIG 2015) dataset.

In this research, opcode and application programming interface (API) call name sequences are extracted from the malware assembly files firstly. Then, in consideration of the benefits of pre-training strategy for achieving better performance, a Word2Vec model, which encodes textual data with distributed representation by considering the context, is implemented for input name vectorization. Compared with one-hot encoding approach, Word2Vec encodes the input names into more compact numeric vectors by training a language model, and it leads to lower memory footprint and better representational ability. Finally, a TCN, as an advanced convolutional network structure for sequence modeling tasks, is developed to attribute the malware. Compared with other recurrent neural networks (RNNs), e.g., gated recurrent unit (GRU) and LSTM, TCN is easy to implement in parallel because of its convolutional structure. In addition, TCN demonstrates significant advantage of lower memory requirement than canonical recurrent networks due to the shared filters across the convolutional layers. Our contributions in this article are summarized as follows.

- A Word2Vec model is pre-trained to vectorize the input names. Word2Vec model vectorizes the names in extracted sequences by training a language model. In this article, we demonstrate the better performance and lower memory footprint of WordVec on malware categorization task by comparing with the popular one-hot encoding approach.
- A TCN is developed to attribute the malware. TCN is a specific convolutional network structure for sequence modeling tasks. In this article, by comparing with the LSTM sequence modeling structure, the performance of TCN computing paradigm on IoT software safety is explored, and it reveals that TCN has better performance with less training time.
- Specifically, the effectiveness of our Word2Vec-based TCN scheme is validated on the Microsoft Malware Classification Challenge (BIG 2015) dataset. Our experimental results have been compared with some other similar methods and recent works citing the same dataset, and then the comparisons show better performance of our scheme.

The remainder of this article is organized as follows. The next section gives a summary of the background consisting of Word2Vec model, TCN structure, and recent works on IoT malware classification and categorization. Following that, the proposed scheme and the time complexity are elaborated and analyzed. Then, the next part describes the experimental settings and results for model evaluation. The final section includes a conclusion of the proposed scheme and a promising direction for further research.

Background

Word2Vec model

Input name sequences from malware samples are textual data that should be encoded into numeric vectors for feature representation. Word embeddings are general approaches to map primitive representation of words into high-dimensional numeric vectors in an embedding space with maintaining word distances. Nowadays, word embeddings have gained an incresed research interest,

and among which Word2Vec is one of the most significant text representation models [9, 10]. Word2Vec assumes that the contexts in the natural language are of high correlation, and hence words can be vectorized according to the contexts [11]. Then, word vectors can be obtained from training corpus to measure the semantic similarities between words in natural language. Note that word vectors are generally generated from the weights of trained language models rather than the direct training targets in Word2Vec. Generally, Word2Vec includes two kinds of architectures, i.e., contextual bag-of-words (CBOW) and skip-gram (SG), to learn distributed representation [12–14]. A simple skip-gram model architecture is shown in Fig. 1 [10].

A large and growing body of literature has studied the effectiveness of Word2Vec model in various areas. In [15], Word2Vec technique was applied to social relationship mining in a multimedia recommendation method. This method recommended users multimedia based on a trust relationship, and Word2Vec here was used to encode the sentiment words in related comments into word vectors. In [16], a Word2Vec-based music modeling method adopted skip-gram to model slices of music from a large music corpus. Word2Vec was proved a useful embedding technique to capture meaningful tonal and harmonic relationships in music according to their experimental results. Word2Vec has also shown powerful representation ability for inverse virtual screening in the early stage of drug discovery process. In [17], Word2Vec was combined with a dense fully connected neural network algorithm to perform a binary classification on input protein candidates. In addition, several recent studies investigating Word2Vec in the areas of malware classification and detection have been carried out. In [18], a malware detection method named DroidVecDeep was designed to detect unknown malicious applications on the Android platform. Here, features were extracted by static analysis and ranked by mean decrease impurity firstly, and then were transformed into compact vectors to train a deep classifier according to Word2Vec model. In [19], a LeNet5 structure was developed for malware classification based on the multi-channel feature matrixes, which were converted from malware binary files and assembly files via Word2Vec technique.

Temporal convolutional network

RNNs are considered the general methods for sequence modeling tasks. However, certain convolutional structures show state-of-the-art performance in some sequence modeling tasks, such as audio synthesis, machine translation, and language modeling [20–22]. Then, to verify whether convolutional structures are subject to some specific sequence modeling applications, TCN structure was developed and compared with common RNNs, such as GRU and LSTM, on a comprehensive set of sequence modeling tasks. The comparison results on these tasks indicate better performance and longer effective memory of TCN structure [23].

TCN uses a specific 1D convolutional structure for sequence information representation. Assuming $\boldsymbol{x} = (x_1, \ldots, x_t, \ldots, x_l)$ is the input sequence, l denotes the input sequence length, x_t denotes the input at time step t, $\boldsymbol{g} \in \mathbb{R}^{h \times n}$ represents n convolutional filters with kernel size h, "$\star$" denotes convolution operator, and then a canonical 1D convolutional operation can be formed as [24]:

$$G(x_t) = (\boldsymbol{x} \star \boldsymbol{g})(t) = \sum_{k=0}^{h-1} x_{t-k} \boldsymbol{g}_k^{\mathrm{T}}, \tag{1}$$

However, 1D convolutional networks are facing information leakage and output shrink problems. To overcome these limitations, TCN combines 1D fully-convolutional network (FCN) and casual convolutions [25]. In 1D FCN, hidden layers have the identical length as input sequence to prevent output length shrink. In casual convolutions, output at time step t is convolved only with the neural nodes at time t and the earlier ones in the previous layer. Moreover, considering the receptive field of 1D FCN is linear to the number of convolutional layers, dilated convolution technique is integrated into TCN structure for

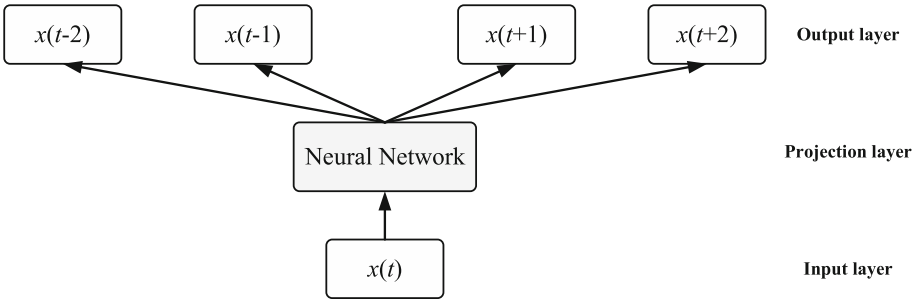

Fig. 1 The architecture of skip-gram model

longer effective memory. Then, the dilated convolutional layer can be defined as:

$$G(x_t) = \left(\boldsymbol{x} \star_d \boldsymbol{g}\right)(t) = \sum_{k=0}^{h-1} x_{t-d \cdot k} \boldsymbol{g}_k^{\mathrm{T}}, \tag{2}$$

where "$\star_d$" denotes the convolution operation with dilation factor d.

Residual connection is another important ingredient of TCN [26]. According to residual connection, the output of a branch which contains a series of transformations $\mathcal{G}$ is added to the input of the block. Assuming the input of the residual block is z, and the output of the block is o, then the residual block can be defined as:

$$o = \boldsymbol{z} + \mathcal{G}(\boldsymbol{z}). \tag{3}$$

Compared with canonical RNNs, such as LSTM and GRU, TCN always has longer effective memory and better performance. Additionally, two other advantages are determined by the particular TCN structure. The fact that neural nodes in each hidden layer are not sequentially connected enables parallel computation for higher computational efficiency, and the shared filters across each layer lead to fewer parameters in TCN. A common TCN structure is illustrated in Fig. 2 [27].

Machine learning methods on edge malware detection and categorization

With the rapid development of IoT, smart devices have suffered various attacks in edge computing paradigm. For instance, in the distributed denial-of-service (DDoS) attack on October 21, 2016, large amounts of IoT devices, such as digital video recorders (DVRs) and internet protocol (IP) cameras, were infected by Mirai to participate in this attack [28]. Therefore, research on malware categorization and analysis in the IoT environment is of great significance. As machine learning methods, such as support vector machine (SVM), extreme learning machine (ELM), neural network (NN), have shown good achievements on classification tasks, there has been a surge of interest in machine learning methods on edge malware detection in recent years. In [29], Sagar developed a three-stage malware detection model to improve detection performance. Term frequency-inverse document frequency (TF-IDF) and information gain (IG) features were extracted in the first stage, and then principal component analysis (PCA) technique was brought in for feature extraction. Finally, a deep belief network (DBN) with optimized activation function was constructed to attribute the malware. In [4], Niu et al. combined static analysis and extreme gradient boosting (XGBoost) method to overcome the low accuracy of static analysis and high resource overhead of dynamic analysis on X86-based IoT devices in an autonomous driving application. In [30], the opcodes of IoT applications were transmuted into a vector space, and then fuzzy and fast fuzzy tree methods were developed to detect and classify the malware. In addition, control flow graph (CFG) was another common choice for malware classification. In [31], a CFG-based deep learning model was constructed to identify malware and benignware IoT disassembled samples.

The proposed malware categorization scheme

In this section, a brief introduction to the malware dataset for this work are described firstly. Then, pre-processing to filter the input sequences is analysed. Furthermore, a Word2Vec-based TCN for malware categorization is elaborated. Through the employment of a pre-trained Word2Vec model, the input name sequences are embedded into a vector space, and then a TCN structure is developed for malware categorization. The whole process is illustrated in Fig. 3. The comparison between the state-of-the-art Word2Vec-based LSTM approach (left) and our proposed scheme (right) is illustrated in Fig. 4. The comparison in Fig. 4 shows that the main differences between our proposed scheme and this Word2Vec-based LSTM are pre-processing and categorization network. In pre-processing, we apply extra useful tricks for feature extraction. Continuously repeated names representing repeat processes in program execution provide no additional information for malware categorization. Therefore, the strategy to remove the repeat is designed here. In addition, too short sequences, which provide

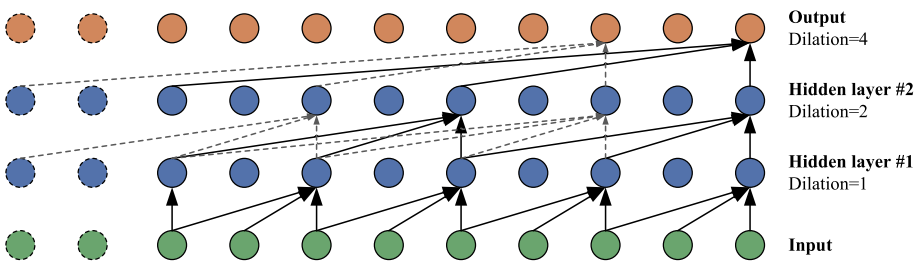

Fig. 2 A general TCN structure with causal operation. Dilation factors are equal to {1, 2, 4}, and filter size is 3. The dashed lines represent the network delayed one time step

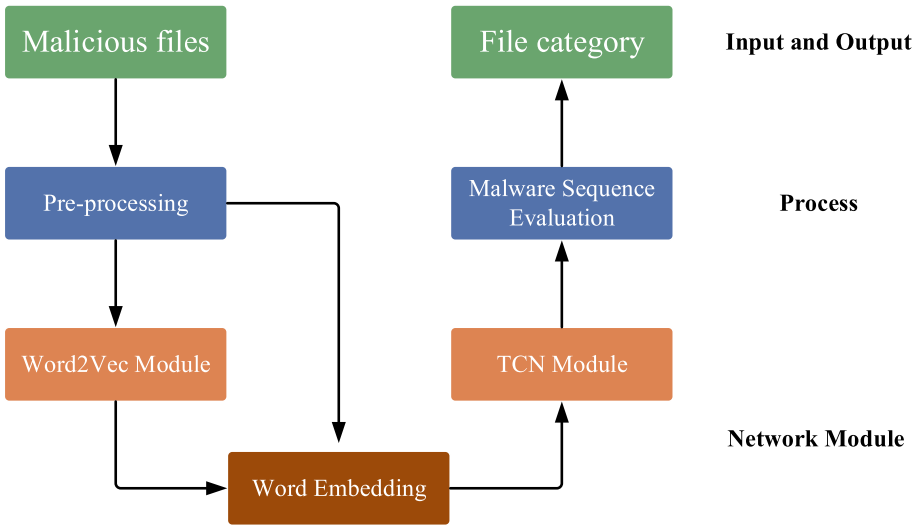

Fig. 3 The whole process of our proposed scheme. The whole process consists of input, output, preprocessing and test set validation process, and three network modules Word2Vec pre-trained model, input embedding module and TCN categorization module

inadequate information for family classification and lead in much noise for feature representation, are eliminated in our scheme. Considering the categorization network, TCN in our scheme has longer effective memory due to the dilated convolution structure. Moreover, residual structure is another reason that our scheme performs better than Word2Vec-based LSTM. More details about the proposed scheme are described in the following parts.

Dataset

In this article, experiments on the Kaggle competition of Microsoft Malware Classification Challenge (BIG 2015)

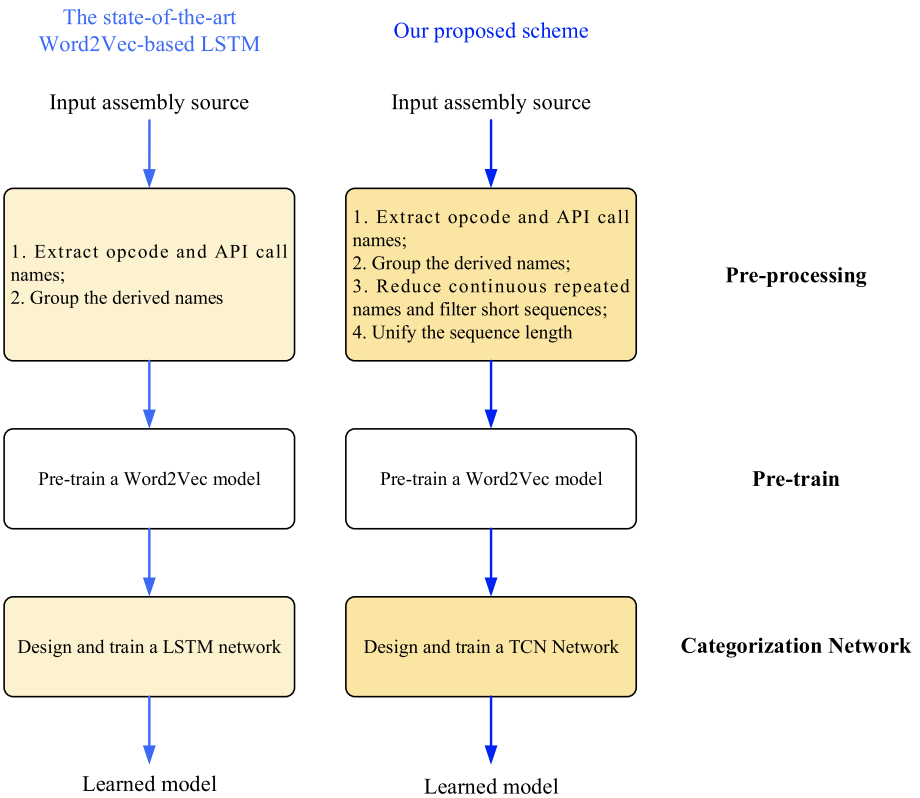

Fig. 4 Comparison between the state-of-the-art Word2Vec-based LSTM and our proposed scheme. The main different parts are shaded light grey

dataset [32] are performed to evaluate the proposed scheme. The original dataset with approximately 500GB consists of more than 20K malware samples belonging to nine malware families. In this work, considering the test data with no labels are unavailable for supervised tasks, only labeled training data in the whole competition dataset are utilized. The corresponding assembly source file of every malicious program is produced from binary file through interactive disassembler pro (http://www.hex-rays.com/products/ida/). Then the opcode and API call name sequences are extracted from the corresponding assembly source files.

Pre-processing

Input name sequences are roughly extracted from assembly source files, and therefore further data processing is an essential and primary step before feature representation [33]. Some extracted sequences contain many consecutive duplicate opcode and API call names which supply no more information for modeling. Then reducing consecutive repeated names is an imperative procedure. Meanwhile, extracted sequences from assembly source files have unequal lengths so that unifying the length of the sequences is another consideration. As a whole, main data pre-processing techniques in this work are as follows:

- Filter consecutive duplicate opcode and API call names: Remove the consecutive and identical names in input sequences to avoid redundant information.
- Filter short sequences: Some sequences from assembly source files which just consist of several opcode and API call names may contain insufficient information to identify the corresponding programs, and these sequences will be removed from the dataset.
- Unify the sequence length: Samples with various length are tricky for neural networks, and therefore unifying the sequence length is imperative for malware categorization. In this work, a sequence length L is pre-set to equalize the lengths [34]. The sequences with length longer than L retain the first L names, and those shorter than L are unified via zero-padding.

After the data pre-processing, the sample size in dataset reaches 10868 and the vocabulary contains 1121 unique opcode and API call names. In the experiments, the extracted sequences are split into training set, validation set, and test set with the proportion of 0.64, 0.16, and 0.2, respectively. The statistical information of each category is shown in Fig. 5 and data samples are shown in Fig. 6.

Word2Vec-based TCN structure

Word2Vec-based TCN mainly consists of a Word2Vec and a TCN sequence analysis model. In this structure, input sequences are transmitted to Word2Vec model in the first step, and then the embedding layer weights are initialized with the numeric vectors from the trained Word2Vec model. Subsequently, a specific TCN for malware categorization is trained. Finally, the Word2Vec-based TCN model is automatically evaluated on the test set. The algorithm description is presented in Algorithm 1.

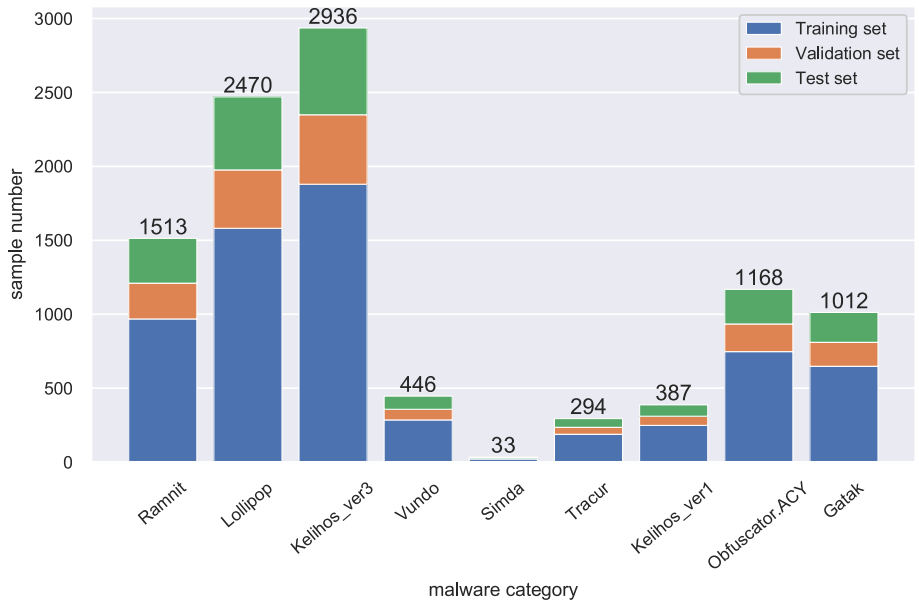

Fig. 5 Data statistics of the malware dataset for each category. The number above the bar is the total number of the corresponding category

dec	push	dec	sub	cmp	dec	mov	dec	mov	dec	mov
push	movzx	push	mov	push	mov	push	mov	push	add	mov
sub	push	call	GetWindowsDirectoryA	test	pop	add	retn	mov	push	call
dd	retf	db	mov	push	pop	push	pop	retn	mov	push
include	push	call	GetFileAttributesW	xor	cmp	setnz	mov	retn	push	mov
include	mov	add	mov	lea	mov	imul	mov	sub	mov	sub
inc	retn	xor	retn	push	xor	test	xor	pop	retn	push
dd	mov	push	mov	sub	lea	push	add	pop	push	pop
include	align	push	mov	push	call	add	mov	pop	retn	align
mov	push	mov	sub	lea	adc	push	add	pop	push	retn
dd	dw	dd	align	dd	align	dd	align	dw	align	unicode
align	push	call	RegisterWindowMessageW	retn	align	call	RegisterWindowMessageW	retn	align	retn
align	mov	push	mov	sub	push	xor	cmp	mov	cmp	mov
popa	xlat	bound	popa	mov	daa	arpl	inc	stosd	out	or
include	align	push	mov	push	call	add	mov	pop	retn	align
pop	xor	call	push	db	retn	push	pop	xor	call	push
include	align	push	mov	push	call	add	mov	pop	retn	align
include	push	call	LoadLibraryA	push	call	GetProcAddress	call	retn	push	mov

Fig. 6 A part of malware opcode and API call name sequence samples

Algorithm 1: Word2Vec-based TCN

Input:
training set, test set, number of filters, batch size, embedding size, number of hidden layers, dropout rate, number of stacks, kernel size.
Output:
The output matrix.
(**a**) Train a Word2Vec model using the input sequences in training set.
(**b**) Construct the TCN model.
(**c**) Train the TCN model:
(**c**-**1**) Use the trained Word2Vec weights to initialize the weights in embedding layer;
(**c**-**2**) Process forward propagation based on training set, and calculate the categorical cross-entropy loss according to the loss function in "Loss function and optimization" section;
(**c**-**3**) Process back propagation to obtain gradients, and update model parameters by the gradients;
(**c**-**4**) Step out if model reaches convergence, else jump to (**c**-**2**).
(**d**) Evaluate model performance. Feed the input sequences in test set into trained TCN model, and output the predicted malware class matrix.

TCN, which consists of several specific convolutional strutures, is an advanced sequence modeling structure. Compared with common RNNs, such as LSTM and GRU, TCN is characterized by fewer network parameters and faster training speed with better performance on sequence modeling tasks. In this article, a TCN structure as illustrated in Fig. 7 is developed for malware categorization. In Fig. 7, the TCN is constructed by stacked residual blocks where the dilation factor is exponentially grown as the blocks are stacked. In addition, each residual block contains two dilated causal convolutional layers and all the convolutional layers contain 32 filters in this TCN. Finally, in the last layer, "fc" which is a fully connected layer with 9 hidden neurons and softmax activation function outputs the predicted family probabilities.

Loss function and optimization

Considering malware categorization on the Microsoft Malware Classification Challenge (Big 2015) dataset is affiliated with multi-class problems, categorical cross-entropy loss function is adopted in this article.

Assuming y_{ij} denotes the true probability of the ith sample belonging to malware family j, $\hat{y}_{ij}$ denotes the predicted probability of the ith sample belonging to family j, N denotes the sample size, M denotes the number of malware families, and then categorical cross-entropy loss function is defined as:

$$\mathbb{L}_{loss} = -\sum_{i=1}^{N}\sum_{j=1}^{M}\widehat{y}_{ij}\log y_{ij}. \tag{4}$$

Adam optimizer, which combines the first moment estimation and the second moment estimation of the gradient, is a common optimizer in neural networks [35]. Hence Adam optimizer is employed in this work.

Time complexity

When 1D convolutional structure is used for sequence modeling in natural language processing, the input sequences are always encoded into numeric vectors firstly. Then, assuming $\boldsymbol{x} \in \mathbb{R}^{l\times m}$ is an input sequence where l denotes the length of the input sequence and m denotes the dimensionality of embedding space, n denotes the number of convolutional filters, h denotes the length of the 1D convolutional filter kernel ($l \gg h$), and then the time complexity of the 1D convolutional layer is:

$$\mathcal{O}(lmhn). \tag{5}$$

Assuming d is the dilation factor of the dilated convolutional layer in TCN, the time complexity of this dilated convolutional layer is:

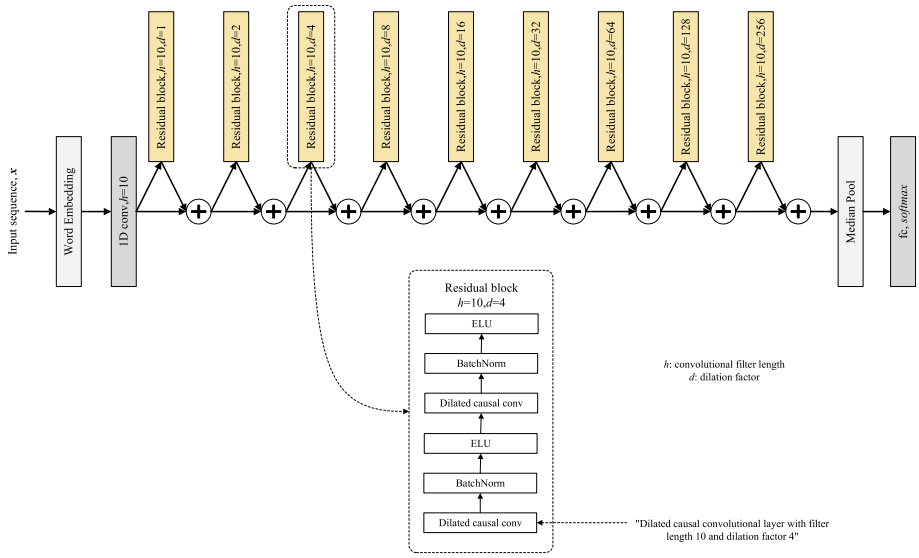

Fig. 7 The TCN structure for malware categorization. The first convolutional layer (in grey) is used for matching the embedding size and filter length, then some continuous residual blocks with different dilation factors are stacked (in yellow)

$$\mathcal{O}\left(\frac{1}{d}lmhn\right). \tag{6}$$

Moreover, the mathematical form of a residual connection is:

$$o = z + \mathcal{G}(z), \tag{7}$$

where o denotes the residual block output, z denotes the input of the block and $\mathcal{G}$ denotes a series of transformations. It can be seen that the residual connection is linear, assuming $\mathcal{G}$ in a residual block contains two dilated convolutional layers which is the general case, then the time complexity of this TCN structure can be approximately estimated as:

$$\mathcal{O}\left(\frac{2}{d}lmhn\right). \tag{8}$$

From (5) and (8), the time complexity between TCN residual block and 1D convolutional structure is roughly comparable. Considering the input data are determinate after pre-processing and embedding space construction, the number of convolutional filters and the length of filter kernels is the main variable parameters in convolutional structure for time consumption. Moreover, since dilated convolutions are potent tricks in TCN structure for a large receptive field, TCN residual blocks enable less computing time with the growth of the dilation factor. Finally, the TCN will achieve good performance with less time consumption by stacking with several residual blocks.

Experiments

To evaluate the performance of our proposed malware categorization scheme, the classical Naive Bayes Classifier for N-gram model (Ngram NBC, for short) is the baseline in our experiments [36]. In addition, to verify that the numeric vectors from pre-trained Word2Vec model are capable to represent the malware feature sequences more precisely, the current popular one-hot encoding technique combined with TCN (OneHotTCN, for short) is compared in our experiments. Then, our proposed scheme (Word2VecTCN, for short) is compared with the state-of-the-art malware categorization model in [34] (Word2VecLSTM, for short). Finally, our scheme is compared with some other recent works on the same malware dataset.

Experimental environment

Our experiments are conducted on the Kaggle competition of Microsoft Malware Classification Challenge (BIG 2015) dataset to evaluate the malware categorization performance on the IoT malware recognition task. Considering the samples in each category are different in quantity, we divide the dataset into training, validation, and test set in a stratified fashion to ensure the same relative proportion in each set. More dataset statistical information is in the previous section.

Here, our experiments are implemented by Python with some additional libraries, such as TensorFlow, Keras, and some others, while the training and evaluation processes are conducted on Tesla K80 GPU in Google Colaboratory system, which is a Google cloud service supporting

Table 1 Parameters selection

Parameter	TCN	LSTM
max sequence length	600	600
batch size	64	64
learning rate	1e-3	1e-3
embedding size	300	300
number of layers	1	2
dropout rate	0.5	0.2
hidden layer neuron	-	128
number of filters	32	-
number of stacks	1	-
dilations	$2^0, \ldots, 2^8$	-
kernel size	10	-

artificial intelligence research [37]. In addition, early stopping and learning rate schedule are extra strategies in the training phase. The learning rate is initially 0.001, and then reduced to 10% of the original value if the validation loss stops declining for 5 epochs.

Metrics

The following basic criteria are universally defined for performance evaluation of machine learning techniques: true positive (TP), true negative (TN), false positive (FP), and false negative (FN). Here, to evaluate the performance of the malware categorization models, some metrics based on the above criteria such as true positive rate (TPR), false positive rate (FPR), positive predictive value (PPV), F-measure (F-M), accuracy (ACC) are calculated and compared [38]. In the experiment, the metrics of each malware family are computed firstly. Then, considering class imbalance problem in this dataset, further weighted results of the nine malware families are also calculated in this article. The metrics are defined as:

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} \times 100\%,$$

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP}+\mathrm{TN}} \times 100\%,$$

$$\mathrm{PPV} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}} \times 100\%,$$

$$\mathrm{F\text{-}M} = \frac{2 \times \mathrm{TP}}{2 \times \mathrm{TP}+\mathrm{FP}+\mathrm{FN}} \times 100\%,$$

$$\mathrm{ACC} = \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{TN}+\mathrm{FP}+\mathrm{FN}} \times 100\%.$$

Additionally, total training time, test time, and training time per epoch are other important indexes to be used for time consumption evaluation in this article.

Parameter selection

The parameters in our proposed scheme and comparison methods are elaborated in this section. As shown in Table 1, TCN and LSTM have some similar parameters. Here, "max sequence length" is the maximum length of the opcode and API call name sequences. Then, the sequences whose lengths are longer than the threshold are clipped to "max sequence length", and the shorter ones are padded with 0 to reach the fixed "max sequence length". The parameter "batch size" is the number of samples fed

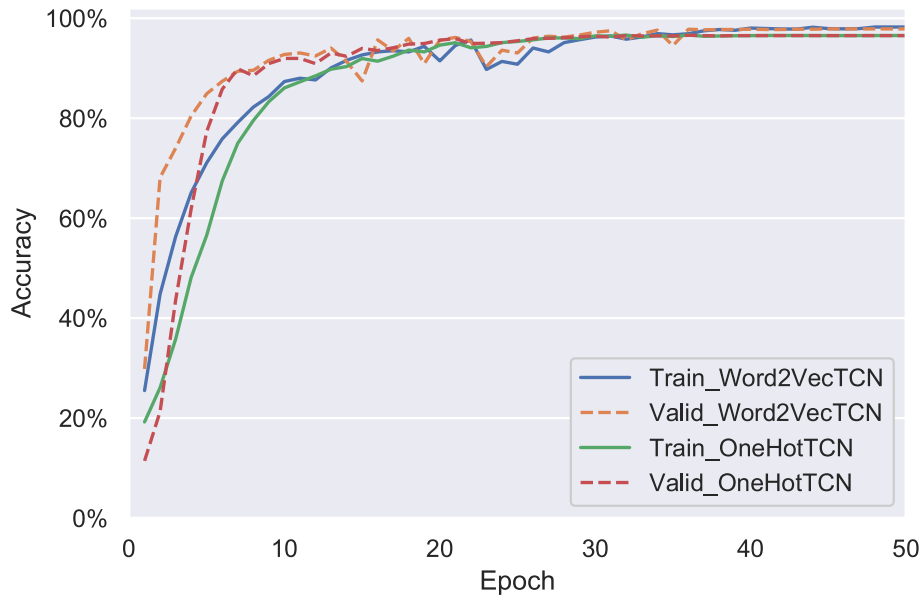

Fig. 8 The accuracy comparison between TCN-based methods

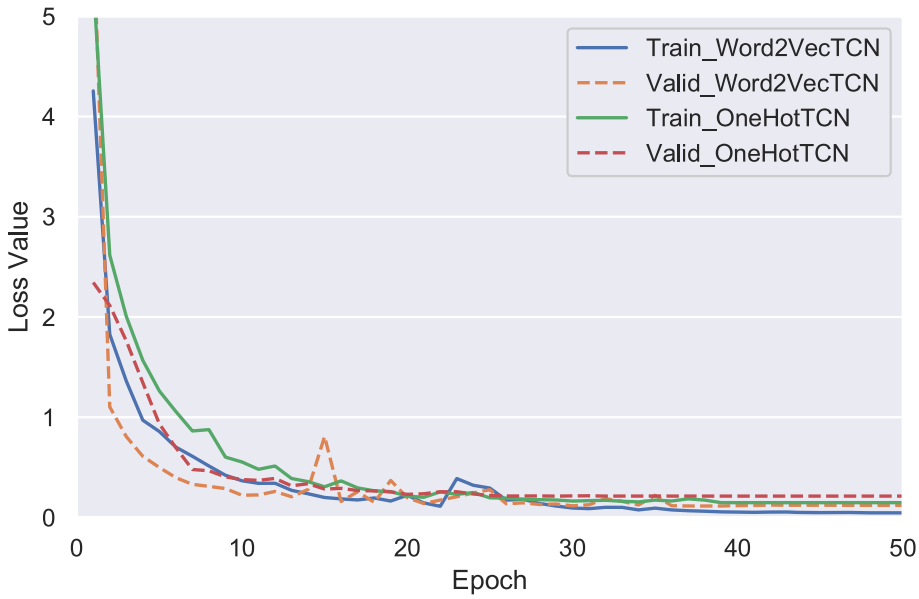

Fig. 9 The loss comparison between TCN-based methods

into the models in each iteration. The parameter "learning rate" is the learning rate in the optimization procedure. Malware opcode and API call names should be mapped into numeric vectors before feature representation, and "embedding size" is the dimension of embedding space. The parameter "number of layers" points out the number of the network layers. For example, two LSTMs are stacked in this article. The parameter "dropout rate" is the dropout proportion of the network nodes in the training phase. The parameter "hidden layer neuron" represents the number of neural neurons in LSTM hidden layers. The parameter "number of filters" is the filter size in the convolutional layers. The parameter "number of stacks" is the number of stacked convolutional structures in residual blocks. Considering dilated convolution used in TCN, "dilations" is a list of dilation factors in dilated convolution. The parameter "kernel size" is the filter kernel size in the convolutional layers. There is no need to tune all

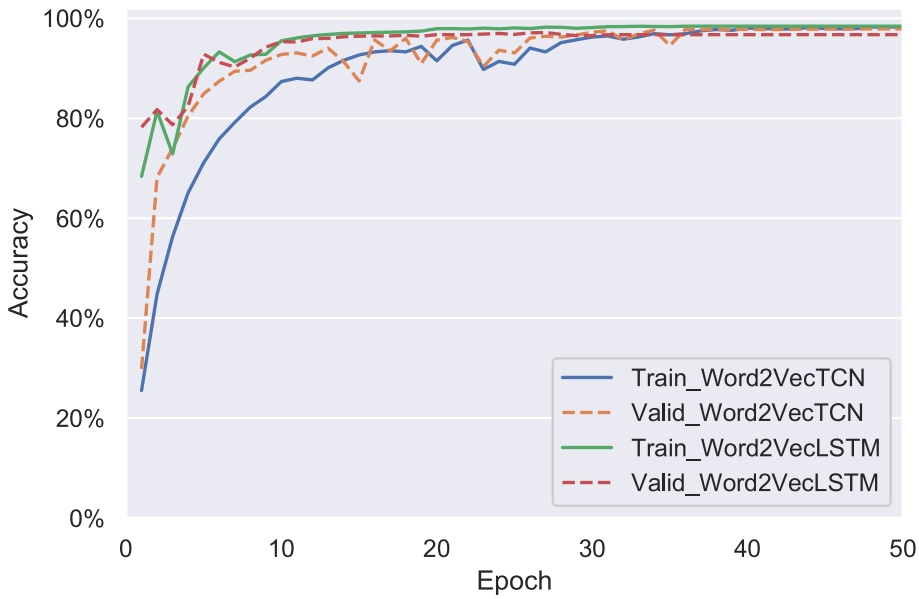

Fig. 10 The accuracy comparison between the state-of-the-art approach and our proposed scheme

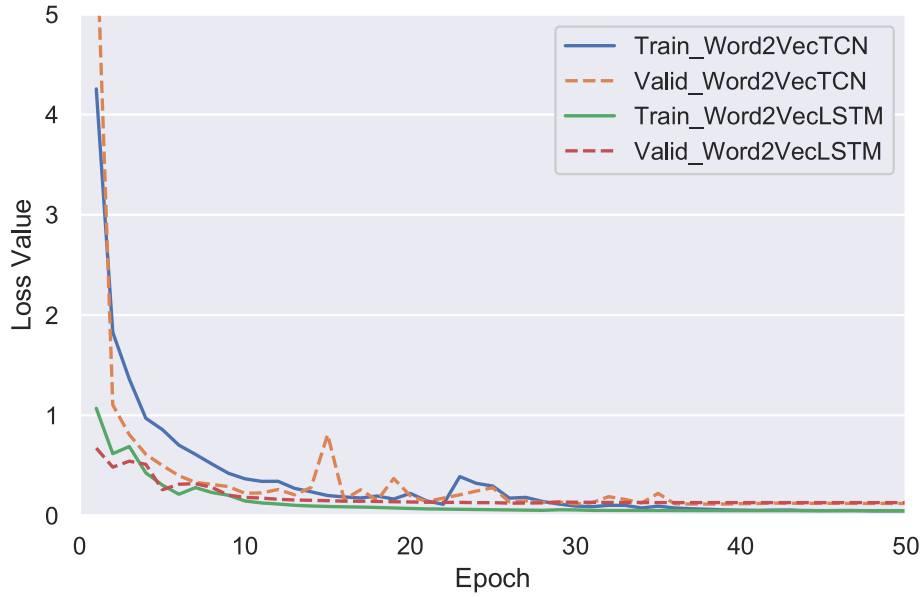

Fig. 11 The loss comparison between the state-of-the-art approach and our proposed scheme

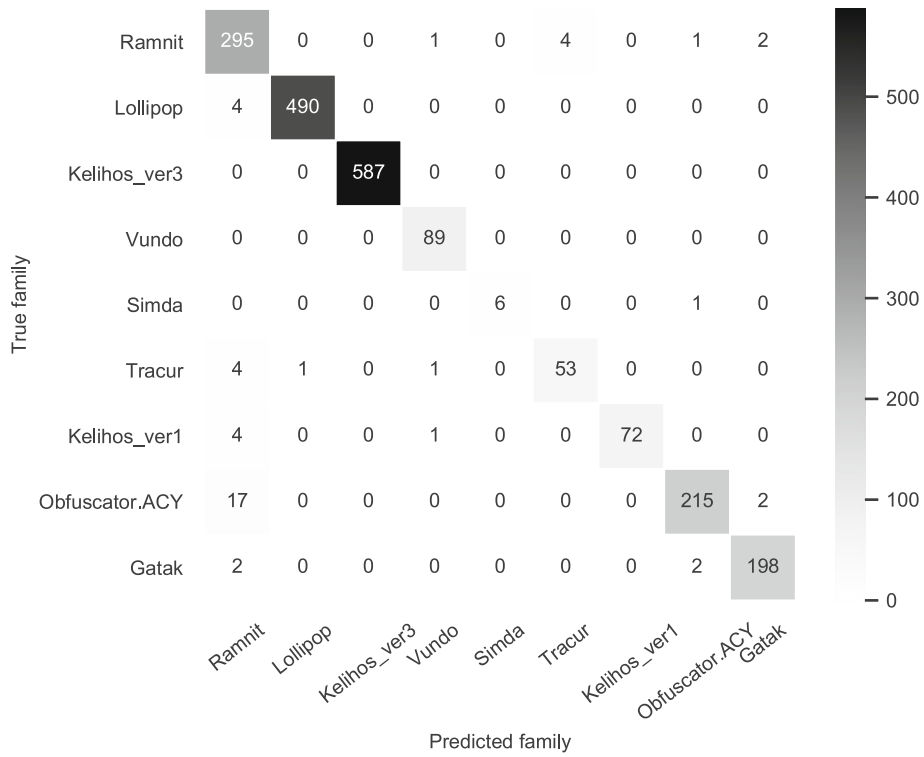

Fig. 12 Confusion matrix for malware categorization with the Word2Vec-based TCN scheme

Table 2 Performance for malware categorization with the Word2Vec-based TCN structure

Family	TPR	FPR	PPV	F-M
Ramnit	97.00%	1.80%	90.20%	93.50%
Lollipop	98.80%	0.10%	99.80%	99.30%
Kelihos_ver3	100.00%	0.10%	99.80%	99.90%
Vundo	98.90%	0.20%	96.70%	97.80%
Simda	85.70%	0.10%	85.70%	85.70%
Tracur	88.10%	0.20%	94.50%	91.20%
Kelihos_ver1	93.50%	0.00%	100.00%	96.60%
Obfuscator.ACY	93.20%	0.40%	96.90%	95.00%
Gatak	97.00%	0.20%	98.50%	97.80%

parameters in both networks, and "-" represents the corresponding parameter is inexistent in current network. Moreover, the parameters in OneHot-based method are basically identical with those in Word2Vec-based TCN, except that there is no "embedding size" in OneHot-based TCN.

Results

Experimental results are shown in this section. Figures 8 and 9 reveal the accuracy and loss comparisons between our scheme and OneHot-based TCN in the training phase. Figures 10 and 11 reveal the accuracy and loss comparisons between our scheme and Word2Vec-based LSTM in the training phase. The confusion matrix on the test set of our scheme is illustrated in Fig. 12. Then the metrics on each family of our scheme are computed in Table 2. The weighted evaluation metrics and the time consumption comparisons on this malware categorization task are presented in Tables 3 and 4, separately. Finally, accuracy comparison between our scheme and other works on this dataset is conducted in Table 5.

The comparisons between our proposed Word2Vec-based scheme and OneHot-based one are shown in Figs. 8 and 9. From Fig. 8, the validation accuracy of our scheme is initially 29.8% and increases to a final value of 97.9%, while the validation accuracy of OneHot-based TCN method is initially 11.4% and grows to a final accuracy of 96.5%. From Fig. 9, the validation loss is initially 5.88 and decreases to 0.12 finally of our scheme, while the validation loss of OneHot-based TCN is initially 2.34 and decreases to 0.21 finally. The two figures reveal Word2Vec owns stronger feature representation ability than the one-hot encoding on this malware category dataset. Specifically, in terms of the embedding layer, the dimension of numeric vectors generated from one-hot encoding reaches 1121 which is the number of unique opcode and API call names, while the dimension of numeric vectors trained from Word2Vec is 300. It can reduce large memory footprint in edge devices.

The comparisons between our proposed scheme and the state-of-the-art Word2Vec-based LSTM model are shown in Figs. 10 and 11. From these two figures, considering that the "dropout rate" in our scheme is higher than that in Word2Vec-based LSTM, our scheme is a bit behind Word2Vec-based LSTM model at the beginning of the training phase. Still, with the powerful feature representation ability, our scheme achieves higher accuracy and lower loss than Word2Vec-based LSTM model both on the training set and validation set finally. Furthermore, Word2Vec-based model needs to train about 672 thousand parameters while our scheme just requires approximately 582 thousand parameters, and the results show that Word2Vec-based TCN has better representation ability and lower running memory in the training phase.

Figure 12 presents the predicted results on test set of our scheme visually, and Table 2 computes the metrics on each malware family. The result combining Fig. 12 and Table 2 reveals that FPR of "Ramnit" is the highest among the nine families, and therefore how to identify the malware samples which are conceived as "Ramnit" more accurately is the bottleneck to enhance the performance of Word2Vec-based TCN scheme. When applying this scheme to the practical IoT environment, the samples recognized as "Ramnit" need to be paid more attention.

Tables 3, 4, and 5 show the comparisons of our scheme and some representative methods. From Table 3, the weighted metrics are computed and compared. The results show that the weighted F-measure and the accuracy of our scheme are approximately 1.2% and 1.3% higher than those of the Word2Vec-based LSTM, and the weighted FPR of our scheme is approximately 0.3% lower. Among all these metrics, Word2Vec-based TCN achieves the best performance. In Table 4, "Training time" is the runtime in the whole training phase, "Test time" is the

Table 3 Performance comparisons of malware categorization

Method	weighted TPR	weighted FPR	weighted PPV	weighted F-M	ACC
Ngram NBC	84.50%	1.80%	87.50%	85.30%	84.50%
OneHotTCN	96.00%	0.50%	96.20%	95.90%	96.00%
Word2VecLSTM	96.20%	0.70%	96.40%	96.30%	96.20%
Word2VecTCN	**97.50%**	**0.40%**	**97.60%**	**97.50%**	**97.50%**

The values in boldface show the best results among the comparisons

Table 4 Computing time comparisons of malware categorization (second)

Method	Training time	Test time	Training time (per epoch)
Ngram NBC	0.244	9.016	-
OneHotTCN	571.356	22.568	14.650
Word2VecLSTM	4712.674	11.935	130.908
Word2VecTCN	732.179	16.442	17.432

execution time on the test set, and "Training time (per epoch)" counts average time per epoch in the training phase. Considering the convolutional structure is easy to be trained in parallel and the parameters of our scheme are fewer than those in LSTM, TCN takes much less training time than LSTM. In addition, our proposed scheme has been compared with the other three recent works which are also on the same Microsoft malware dataset in Table 5. The comparison also verifies the good performance of our scheme.

Conclusion

In this article, a Word2Vec-based TCN scheme is proposed for malware categorization in consideration of edge computing security. Opcode and API call name sequences are extracted from malicious samples firstly, and then the pre-processing is conducted for data cleaning. Subsequently, through the Word2Vec pre-training on the feature sequences, numeric vectors of the input names are generated. Additionally, the malware feature sequences represented by numeric vectors are fed into TCN to fit an IoT malware categorization model. Finally, the model performance is evaluated on the test set. The comparisons with other representative works verify that our proposed scheme can achieve decent performance while requiring a small quantity of memory and training time. From the occupancy of resources point-of-view, the benefits of combining Word2Vec model and TCN structure are noticeable.

Considering the low occupancy of resources and good computing performance of our scheme, it has potential applications on smart devices for security. As a universal malware categorization scheme, our scheme suggests its promising applications in multiple fields of edge computing security, such as intelligent transportation system security control, smart factory protection and some others. The applications of our scheme on these edge computing fields will be considered in future work.

Table 5 Performance comparison with recently similar works citing the Microsoft dataset

Method	Accuracy
Word2VecTCN (proposed)	**97.52%**
Rahul, R.K. et al. [39]	94.91%
Cho, Y. [40]	96.00%
Sung, Y. et al. [41]	96.76%

The entries in boldface show the highest accuracy and the corresponding method among this performance comparison

Acknowledgements
Not Applicable.

Authors' contributions
Xiong Luo and Honghao Gao provided the original ideas and were responsible for revising the whole article; Jiankun Sun designed and performed the experiments and wrote the original article; Weiping Wang, Yang Gao, and Xi Yang analyzed the data. All authors read and approved the final manuscript. Corresponding authors: Xiong Luo and Honghao Gao. They contributed equally to this work.

Funding
This work was supported in part by the National Natural Science Foundation of China under Grants U1836106 and U1736117, in part by the Beijing Natural Science Foundation under Grant 19L2029, in part by the Beijing Intelligent Logistics System Collaborative Innovation Center under Grant BILSCIC-2019KF-08, in part by the Scientific and Technological Innovation Foundation of Shunde Graduate School, USTB, under Grant BK19BF006, and in part by the Fundamental Research Funds for the University of Science and Technology Beijing under Grant FRF-BD-19-012A.

Availability of data and materials
The dataset is available from the Kaggle competition link: https://www.kaggle.com/c/malware-classification/data.

Competing interests
The authors declare that they have no competing interests.

Author details
[1]School of Computer and Communication Engineering, University of Science and Technology Beijing, 100083, Beijing, China. [2]Beijing Key Laboratory of Knowledge Engineering for Materials Science, 100083, Beijing, China. [3]Institute of Artificial Intelligence, University of Science and Technology Beijing, 100083, Beijing, China. [4]Computing Center, Shanghai University, 200444, Shanghai, China. [5]China Information Technology Security Evaluation Center, 100085, Beijing, China. [6]Beijing Intelligent Logistics System Collaborative Innovation Center, 101149, Beijing, China.

Received: 1 April 2020 Accepted: 8 September 2020
Published online: 23 September 2020

References

1. Gao H, Duan Y, Shao L, Sun X (2019) Transformation-based processing of typed resources for multimedia sources in the IoT environment. Wirel Netw.1–17. https://doi.org/10.1007/s11276-019-02200-6
2. Chen M, Li Y, Luo X, Wang W, Wang L, Zhao W (2019) A novel human activity recognition scheme for smart health using multilayer extreme learning machine. IEEE Internet Things J 6(2):1410–1418
3. Darabian H, Dehghantanha A, Hashemi S, Homayoun S, Choo K-KR (2020) An opcode-based technique for polymorphic Internet of Things malware detection. Concurr Comput 32(6):5173
4. Niu W, Zhang X, Du X, Hu T, Xie X, Guizani N (2019) Detecting malware on X86-based IoT devices in autonomous driving. IEEE Wirel Commun 26(4):80–87
5. Abawajy J, Huda S, Sharmeen S, Hassan MM, Almogren A (2018) Identifying cyber threats to mobile-IoT applications in edge computing paradigm. Future Gener Comput Syst 89:525–538
6. Guo Y, Han B, Wang W, Yuan M (2019) State estimation and event-triggered control for cyber-physical systems under malicious attack. Math Probl Eng 2019:1–10
7. Xue N, Luo X, Gao Y, Wang W, Wang L, Huang C, Zhao W (2019) Kernel mixture correntropy conjugate gradient algorithm for time series prediction. Entropy 21(8):785

8. Bakhshi Z, Ali B, Jawad M (2018) Industrial iot security threats and concerns by considering cisco and microsoft iot reference models. In: 2018 IEEE Wireless Communications and Networking Conference Workshops (WCNC). IEEE, Barcelona. pp 173–178
9. Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient estimation of word representations in vector space. In: International Conference on Learning Representations 2013, Scottsdale
10. Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J (2013) Distributed representations of words and phrases and their compositionality. In: Neural Information Processing Systems 2013. MIT Press, Lake Tahoe. pp 3111–3119
11. Gao H, Xu Y, Yin Y, Zhang W, Li R, Wang X (2019) Context-aware QoS prediction with neural collaborative filtering for Internet-of-Things services. IEEE Internet Things J 7(5):4532–4542
12. Luo Q, Xu W, Guo J (2014) A study on the CBOW model's overfitting and stability. In: Proceedings of the 5th International Workshop on Web-Scale Knowledge Representation Retrieval & Reasoning - Web-KR '14. ACM Press, Shanghai. pp 9–12
13. Guthrie D, Allison B, Liu W, Guthrie L, Wilks Y (2006) A closer look at skip-gram modelling. In: Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC-2006). ELRA, Genoa. pp 1222–1225
14. Kottur S, Vedantam R, Moura JMF, Parikh D (2016) Visual word2vec (vis-w2v): Learning visually grounded word embeddings using abstract scenes. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, Las Vegas. pp 4985–4994
15. Baek J-W, Chung K-Y (2020) Multimedia recommendation using Word2Vec-based social relationship mining. Multimed Tools Appl.1–17. https://doi.org/10.1007/s11042-019-08607-9
16. Chuan C-H, Agres K, Herremans D (2020) From context to concept: Exploring semantic relationships in music with word2vec. Neural Comput & Applic 32(4):1023–1036
17. Zhang H, Liao L, Cai Y, Hu Y, Wang H (2019) IVS2vec: A tool of inverse virtual screening based on word2vec and deep learning techniques. Methods 166:57–65
18. Chen T, Mao Q, Lv M, Cheng H, Li Y (2019) DroidVecDeep: Android malware detection based on Word2Vec and Deep Belief Network. KSII T Internet Inf 13(4):2180–2197
19. Qiao Y, Jiang Q, Jiang Z, Gu L (2019) A multi-channel visualization method for malware classification based on deep learning. In: 2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE). IEEE, Rotorua, New Zealand. pp 757–762
20. Purwins H, Li B, Virtanen T, Schlüter J, Chang S-Y, Sainath T (2019) Deep learning for audio signal processing. IEEE J Sel Topics Signal Process 13(2):206–219
21. Dauphin YN, Fan A, Auli M, Grangier D (2017) Language modeling with gated convolutional networks. In: Proceedings of the 34th International Conference on Machine Learning. ACM, Sydney. pp 933–941
22. Gehring J, Auli M, Grangier D, Yarats D, Dauphin YN (2017) Convolutional sequence to sequence learning. In: Proceedings of the 34th International Conference on Machine Learning. ACM, Sydney. pp 1243–1252
23. Kim TS, Reiter A (2017) Interpretable 3D human action analysis with temporal convolutional networks. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). IEEE, Honolulu. pp 1623–1631
24. You J, Wang Y, Pal A, Eksombatchai P, Rosenburg C, Leskovec J (2019) Hierarchical temporal convolutional networks for dynamic recommender systems. In: The World Wide Web Conference on - WWW '19. ACM Press, San Francisco. pp 2236–2246
25. Lea C, Vidal R, Reiter A, Hager GD (2016) Temporal convolutional networks: A unified approach to action segmentation. In: Computer Vision - ECCV 2016 Workshops. Springer International Publishing, Amsterdam. pp 47–54
26. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: the 29th IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas. pp 770–778
27. MatthewDavies EP, Bock S (2019) Temporal convolutional networks for musical audio beat tracking. In: 2019 27th European Signal Processing Conference (EUSIPCO). IEEE, A Coruna. pp 1–5
28. Kumar A, Lim TJ (2019) EDIMA: Early detection of IoT malware network activity using machine learning techniques. In: 2019 IEEE 5th World Forum on Internet of Things (WF-IoT). IEEE, Limerick. pp 289–294
29. Sagar GVR (2019) Malware detection using optimized activation-based deep belief network: An application on Internet of Things. J Info Know Mgmt 18(04):1950042
30. Dovom EM, Azmoodeh A, Dehghantanha A, Newton DE, Parizi RM, Karimipour H (2019) Fuzzy pattern tree for edge malware detection and categorization in IoT. J Syst Architect 97:1–7
31. Alasmary H, Khormali A, Anwar A, Park J, Choi J, Abusnaina A, Awad A, Nyang D, Mohaisen A (2019) Analyzing and detecting emerging Internet of Things malware: A graph-based approach. IEEE Internet Things J 6(5):8977–8988
32. Ronen R, Radu M, Feuerstein C, Yom-Tov E, Ahmadi M (2018) Microsoft malware classification challenge. arXiv preprint arXiv:1802.10135
33. Ghorbani M, Bahaghighat M, Xin Q, Özen F (2020) ConvLSTMConv network: A deep learning approach for sentiment analysis in cloud computing. J Cloud Comp 9(1):16
34. Kang J, Jang S, Li S, Jeong Y-S, Sung Y (2019) Long short-term memory-based malware classification method for information security. Comput Electr Eng 77:366–375
35. Song R, Xiao Z, Lin J, Liu M (2020) CIES: Cloud-based intelligent evaluation service for video homework using CNN-LSTM network. J Cloud Comp 9(1):7
36. Kang B, Yerima SY, Mclaughlin K, Sezer S (2016) N-opcode analysis for android malware classification and categorization. In: 2016 International Conference On Cyber Security And Protection Of Digital Services (Cyber Security), London. pp 1–7
37. Bisong E (2019) Google colaboratory. In: Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners. Apress, Berkeley, CA. pp 59–64
38. Hansen SS, Larsen TMT, Stevanovic M, Pedersen JM (2016) An approach for detection and family classification of malware based on behavioral analysis. In: 2016 International Conference on Computing, Networking and Communications (ICNC). IEEE, Kauai. pp 1–5
39. Rahul RK, Anjali T, Menon VK, Soman KP (2017) Deep learning for network flow analysis and malware classification. In: Security in Computing and Communications, vol. 746. Springer Singapore, Singapore. pp 226–235
40. Cho Y (2019) Dynamic RNN -CNN based malware classifier for deep learning algorithm. In: 2019 29th International Telecommunication Networks and Applications Conference (ITNAC). IEEE, Auckland. pp 1–6
41. Sung Y, Jang S, Jeong Y-S, Park JHJJ (2020) Malware classification algorithm using advanced Word2vec-based Bi-LSTM for ground control stations. Comput Commun 153:342–348

Publisher's Note