A Malware Identification and Detection Method Using Mixture Correntropy-Based Deep Neural Network

Xiong Luo[1,2,3(✉)], Jianyuan Li[4], Weiping Wang[1,2,3], Yang Gao[5], and Wenbing Zhao[6]

[1] School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing 100083, China
xluo@ustb.edu.cn
[2] Beijing Key Laboratory of Knowledge Engineering for Materials Science, Beijing 100083, China
[3] Beijing Intelligent Logistics System Collaborative Innovation Center, Beijing 101149, China
[4] Department of Electrical and Computer Engineering, University of Pittsburgh, Pittsburgh, PA 15261, USA
[5] China Information Technology Security Evaluation Center, Beijing 100085, China
[6] Department of Electrical Engineering and Computer Science, Cleveland State University, Cleveland, OH 44115, USA

Abstract. With the rapid development of CPS technology, the identification and detection of malware has become a matter of concern in the industrial application of CPS. Currently, advanced machine learning methods such as deep learning are popular in the research of malware identification and detection, and some progress has been made so far. However, there are also some problems. For example, considering the existing noise or outliers in the datasets of malware, some methods are not robust enough. Therefore, the accuracy of classification of malware still needs to be improved. Aiming at it, we propose a novel method thought the combination of correntropy and deep neural network (DNN). In our proposed method for malware identification and detection, given the success of mixture correntropy as an effective similarity measure in addressing complex dataset with noise, it is therefore incorporated into a popular DNN, i.e., convolutional neural network (CNN), to reconstruct its loss function, with the purpose of further detecting the features of outliers. We present the detailed design process of our proposed method. Furthermore, the proposed method is tested both on a popular benchmark dataset and a real-world malware classification dataset, to verify its learning performance.

Keywords: Mixture correntropy · Convolutional Neural Network (CNN) · Malware detection

H. Ning (Ed.): CyberDI 2019/CyberLife 2019, CCIS 1137, pp. 321–334, 2019.
https://doi.org/10.1007/978-981-15-1922-2_23

1 Introduction

Cyber-physical systems (CPS) refer to a system that integrates computation, networking, and physical processes, where embedded computers and networks achieve real-time control of the physical process through the feedback mechanism [1]. With the development and popularization of CPS technologies, there are numerous physical equipments depending on computers and networks to achieve functional expansion, which will upgrade industrial products and technologies. For those large-scale complex systems, safety and reliability always are important issues. CPS intimately integrates the virtual world with the physical world, once the virtual world is attacked, it will inevitably affect the physical world. Hence, the information protection is essential for the CPS implementation. For the issue of information security in CPS, it has become critical to identify and detect malware on Android [2].

Generally speaking, malware refers to any cyber-attack performed in Internet. The increase of malware has become a serious issue. Due to the serious hazard the malware has brought to the internet, various methods have been proposed to defense the malicious software. Currently, there are two main methods to detect and analyze malware. One is the classical static and dynamic analysis method. Static malware feature analysis includes compile time, shell information, import functions, suspicious strings, and some others. For example, an algorithm was designed to achieve statistical binary content analysis of Fileprint [3], and a statistical value could be calculated to extract malicious communication pattern [4]. If the sample is highly encrypted, the static feature analysis may not provide much valuable information, therefore, dynamic behavior analysis technology is needed, which is also called behavior monitoring. For example, an approach was proposed for the network analysis of anomalous traffic events (NATE) [5]. The other one is based on machine learning methods, such as malware analysis based on long short-term memory (LSTM) [6], one-class support vector machine based malware detection [7], detecting malware using a deep belief network (DBN) [8], and many others. Then, with the rapid development of machine learning algorithms, more and more intelligent methods are accordingly developed to deal with malware issues.

However, these methods mentioned above also have their limitations. In particular, there may be noise or outliers in some malware data, and then the robustness of some methods is not satisfactory enough. Therefore, the accuracy of feature extraction and classification of malware is necessary to be further improved. In response to these limitations, motivated by the popular malware classification method on the basis of deep neural network (DNN) [9], we propose a novel algorithm through the use of a new similarity measure, i.e., mixture correntropy.

Correntropy is a kernel-based local similarity function [10]. Since one of the significant features of correntropy is robust to noise and large outliers [11], it is widely applied in various fields. Specifically, it can be also used within deep learning framework to improve the computational performance. For example, the stacked extreme learning machine was presented with the correntropy-optimized temporal principle component analysis (CTPCA) [12], furthermore the generalized correntropy-based stacked autoencoder (GC-SAE) was developed [13]. More recently, on the basis of correntropy, mixure correntropy is proposed and widely employed in various applications [14]. Considering that there is no application of mixture correntropy in

Android malware identification and detection, through the combination of a popular DNN, i.e., convolutional neural network (CNN) [15], we develop a novel application by proposing a mixture correntropy-based CNN, and thus using it to improve the classification accuracy for malware.

The main contributions of this paper can be summarized as follows.

(1) In consideration of those advantages of mixture correntropy in addressing data more flexibly and stably, it is hereby incorporated into the implementation framework of CNN, through the reconstruction of loss function in CNN. Then, it is expected that the learning performance can be further improved by using our proposed method.
(2) Our proposed method is used to handle an important but challenging issue in the guarantee of information security in CPS, which is the Android malware identification and detection. Compared to other traditional malware detection methods, the learning performance in relation to accuracy and robustness would be further improved owing to the use of mixture correntropy.

The remainder of this paper is organized as follows. In Sect. 2, we provide a simple analysis on the related technologies, including correntropy and mixture correntropy, CNN, and deep learning-based malware detection. In Sect. 3, the detailed design process of our proposed method is presented. The experiment results and discussion are given in Sect. 4. Finally, this paper is concluded in Sect. 5.

2 Background

2.1 Correntropy and Mixture Correntropy

Correntropy. Inspired by information theoretic learning (ITL) [16], correntropy is an extension of the basic definition of correlation function, which is a similarity measure function of two random variables (X, Y). It is defined as:

$$V(X,Y) = \mathrm{E}[k_\sigma(X-Y)] = \int k_\sigma(x-y)dF_{XY}(x,y) \tag{1}$$

where $k_\sigma(\cdot)$ denotes any type of kernel function with bandwidth of σ, $\mathbf{E}$ is the expectation operator, $F_{XY}(x,y)$ refers to the joint distribution of (X,Y). Without mentioned otherwise, the kernel function in this paper takes Gaussian kernel:

$$k_\delta(x,y) = G_\sigma(e) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{e^2}{2\sigma^2}\right) \tag{2}$$

where e refers to $(x-y)$. Correntropy is symmetric, positive, and bounded, and contains all even moments of arbitrary variables [17]. Since in real-world data processing tasks, the joint probability density (PDF) of samples is usually unknown, and sample sets $\{x_i, y_i\}_{i=1}^N$ is limited, the sample estimator can be defined as:

$$\widehat{V}(X,Y) = \frac{1}{N}\sum_{i=1}^{N} k_\sigma(x_i - y_i) \tag{3}$$

which is also named as the empirical correntropy.

Mixture Correntropy. On the basis of correntropy, the concept of mixture correntropy is generated [14]. When correntropy is applied to process data, the kernel bandwidth σ directly affect the performance of the function. Concisely, a small value of bandwidth allows algorithms to perform better in processing data with noise and outliers, but it may also lead to a slow convergence. Conversely, a large value of bandwidth will cause the reduction of robustness. Therefore, the mixture correntropy is proposed to improve the original algorithm, and it is defined as:

$$M(X, Y) = \mathrm{E}[\boldsymbol{\alpha} \boldsymbol{k}_{\boldsymbol{\sigma}_1}(\boldsymbol{e}) + (1 - \boldsymbol{\alpha}) \boldsymbol{k}_{\boldsymbol{\sigma}_2}(\boldsymbol{e})] \tag{4}$$

where $\boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_2$ are bandwidths of two kernel functions, and $0 \leq \boldsymbol{\alpha} \leq 1$ refers to mixture coefficient that controls the ratio between two kernel functions. In addition, the sample estimator can be defined as:

$$\widehat{M}(e) = \frac{1}{N} \sum_{i=1}^{N} [\boldsymbol{\alpha} k_{\boldsymbol{\sigma}_1}(e_i) + (1 - \boldsymbol{\alpha}) \boldsymbol{k}_{\boldsymbol{\sigma}_2}(e_i) \tag{5}$$

where e_i refers to $(x_i - y_i)$. In this paper, we take the mixture of two Gaussian kernel. Here, $\widehat{M}(e)$ can be represented as:

$$\widehat{M}(e) = \frac{1}{N} \sum_{i=1}^{N} [\boldsymbol{\alpha} G_{\boldsymbol{\sigma}_1}(e_i) + (1 - \boldsymbol{\alpha}) G_{\boldsymbol{\sigma}_2}(e_i)] \tag{6}$$

2.2 Convolutional Neural Network (CNN)

CNN is a type of feedforward neural networks with convolutional computation, and it can be regarded as a DNN. As one of the representative algorithms of deep learning, it has been utilized in various fields, such as the image classification [18], object recognition [19], and natural language processing [20]. CNN mimics the visual perception mechanism of living organisms, and thus can be employed for the supervised learning and unsupervised learning.

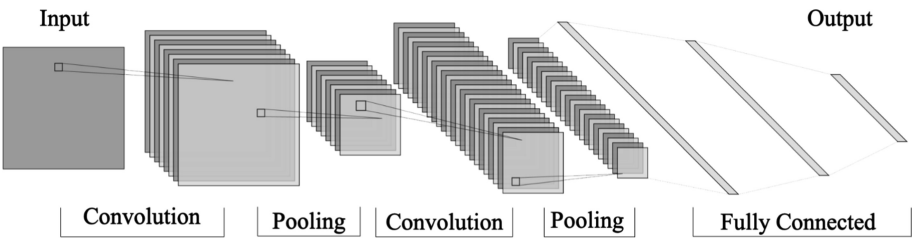

Fig. 1. LeNet-5 model.

Here, we implement a LeNet-5 model [15], which is a common model of CNN. As shown in Fig. 1, there are 7 layers in LeNet-5, including 2 convolutional layers, 2

pooling layers, and 3 fully-connected layers. Each layer contains different number of training parameters. The function of convolutional layer is to extract features from the input data. Thereafter, the feature graph will be transferred to the pooling layer for feature selection and information filtering. Fully-connected layers only pass signals to other fully-connected layers. The feature graph loses its spatial topology in the fully-connected layers and is expanded as a vector. The output layer uses the softmax function to output classification labels.

2.3 The Deep Learning-Based Malware Detection

Deep learning, as a kind of advanced data mining strategy in the machine learning area, has gained tremendous attention and inspired diverse practical applications. To address the security issues of CPS, a variety of deep learning-based malware detection methods have been proposed over the years. An originally designed deep learning model was performed to analyze more than 200 features extracted from static analysis and dynamic analysis of Android App [21]. Using convolutional and recurrent network layers, a neural network was constructed to achieve the best features of malware system [22]. Echo state networks (ESN) and recurrent neural networks (RNN) are utilized for the projection stage to realize the feature extraction [23]. Because the number of potential features would be very large, the random projections were explored to reduce the dimensionality of input data, and several large-scale neural network systems were trained to implement the classification [24].

However, none of the aforementioned methods specifically involved the issue of robustness in the algorithm. Hence, considering that there may be some noise and outliers in the malware data, our algorithm is proposed.

3 The Proposed Method

The application programming interface (API) call sequences are firstly inputted into our proposed method. After preprocessing these data, our mixture correntropy-based convolutional neural network model is used as a classifier to achieve classification for malware.

3.1 Training Convolutional Neural Network with Mixture Correntropy-Based Loss Function

Here, the loss means the cost for predicting the label to be $f(x)$, the predicted label, instead of the true label. In classification tasks, the algorithm aims to maximize the similarity between the output and the labels, in other word, the corrrentropy can be maximized, which is to minimize the expected loss. Therefore, a simple Gaussian kernel correntropy induced loss function can be defined as:

$$\widehat{L}(e) = 1 - G_{\sigma}(e) \tag{7}$$

which is called the C-loss function [25].

The loss function based on two Gaussian kernel mixed correntropy can be defined as:

$$\begin{aligned}\widehat{L}(e) &= 1 - \widehat{M}(e) \\ &= 1 - \frac{1}{\sqrt{2\pi}N}\sum_{i=1}^{N}\left[\frac{\boldsymbol{\alpha}}{\boldsymbol{\sigma}_1}\exp\left(-\frac{e_i^2}{2\sigma_1^2}\right) + \frac{(1-\boldsymbol{\alpha})}{\boldsymbol{\sigma}_2}\exp\left(-\frac{e_i^2}{2\sigma_2^2}\right)\right]\end{aligned} \tag{8}$$

With our proposed loss function, the pseudo-code of whole process for classification tasks is presented in Algorithm 1.

Algorithm 1 Mixture correntropy induced loss CNN

Input: Training set, test set, parameters $\boldsymbol{\sigma_1}$ and $\boldsymbol{\sigma_2}$, number of iterations, T batch size, kernel size of each layers, $t = 0$.

Output: Loss, accuracy (Acc)

1. Construct the CNN model;
2. Train the model:
   - *a.* Sample a batch of data from training set;
   - *b.* Process forward propagation through the data, and calculate the mixture correntropy induced loss according to (8);
   - *c.* Process backward propagation to calculate the gradients;
   - *d.* Update the layer parameters using the gradient;
   - *e.* $t = t + 1$. If $t < T$, loop from step *a*;
3. Process test set in trained CNN model, and calculate Acc.

3.2 Testing Classifier Through Metrics

To evaluate the performance of classification algorithm, the Accuracy (Acc) is defined as:

$$\text{Acc} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{9}$$

where TP, TN, FP, FP refers to true positive, true negative, false positive, and false positive, respectively.

4 Experimental Results and Discussion

In this section, the experiments on a popular benchmark dataset and a real-world malware classification dataset are conducted to evaluate the performance of our proposed algorithm. Considering the noise in the real-word malware data is hard to be removed, to show the strengths of our algorithm clearly, we firstly applied algorithms

to the original image dataset, and then add noise factor. Here, our experiments are implemented with Python compiler environment running on a computer with a 1.6 GHZ CPU and an 8 GB RAM.

4.1 Classification Results on Benchmark Dataset

Experimental Results. The comparison is conducted among four methods, including the support vector machine (SVM), the traditional CNN classifier with the mean square error (MSE)-induced loss function (CNN+MSE), the CNN classifier with correntropy-induced loss function (CNN+Correntropy), and our method, i.e., the CNN classifier with mixture correntropy-induced loss function. In this experiment, we use Fashion-mnist dataset [26]. As shown in Fig. 2, Fashion-MNIST is a dataset of article images, each sample is a 28 $\times$ 28 grayscale image. There are 10 classes in this dataset, consisting training set of 60,000 examples and a test set of 10,000 examples.

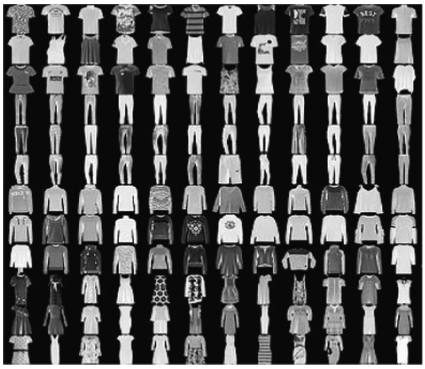

Fig. 2. Illustration of Fashion-mnist dataset.

Firstly, we apply each algorithm on original dataset, and then we add noise into original dataset to test the robustness of each algorithm.

Table 1. Performance of each algorithm on original Fashion-mnist dataset.

Algorithm	Accuracy
SVM	0.8575
CNN-MSE	0.9147
CNN-Correntropy ($\sigma = 0.8$)	0.9154
CNN-MixCorrentropy ($\boldsymbol{\sigma}_1 = 0.5$, $\boldsymbol{\sigma}_2 = 3$, $\boldsymbol{\alpha} = 0.5$)	0.9170

As shown in Table 1, in the original dataset, the accuracy of CNN-MSE, CNN-Correntropy and CNN-MixCorrentropy is very close, and is better than that of SVM. Table 2 presents the performance of four classification method in dataset with Gaussian

noise. The accuracy of all the algorithms decreases, among which, the accuracy of SVM decreases significantly, CNN-MixCorrentropy achieves the best performance. The result shows the robustness of mixture correntropy induced loss function. Specifically, Figs. 3 and 4 show the loss and accuracy of CNN-MixtureCorrentropy on the original dataset and on the dataset with noise, respectively.

Table 2. Performance of each algorithm on Fashion-mnist dataset with normal distribution Gaussian noise (noise factor = 0.3).

Algorithm	Accuracy
SVM	0.6765
CNN-MSE	0.8489
CNN-Correntropy ($\sigma = 0.8$)	0.8575
CNN-MixCorrentropy ($\boldsymbol{\sigma}_1 = 0.5$ $\boldsymbol{\sigma}_2 = 4$ $\boldsymbol{\alpha} = 0.3$)	0.8605

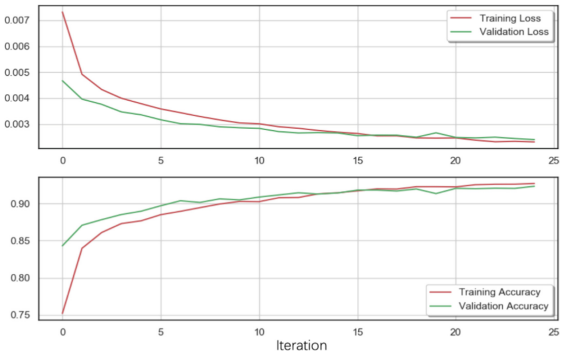

Fig. 3. Loss and accuracy of CNN-MixtureCorrentropy on the original dataset.

Impact of Parameters $\boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_2$. In the experiments, we set up different values of $\boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_2$, and apply them into Fashion-mnist dataset with noise. We define that $\boldsymbol{\sigma}_1 < \boldsymbol{\sigma}_2$. Figure 5 shows the results, which implies that when $0 < \boldsymbol{\sigma}_1 < 1$, for different values of $\boldsymbol{\sigma}_1$ and $\boldsymbol{\sigma}_2$, the performance is basically the same. But when we set $\boldsymbol{\sigma}_1 > 1$, the accuracy decreases significantly.

Additionally, as shown in Fig. 6, when we set $0 < \boldsymbol{\sigma}_1 < 1$, the value of α basically does not affect the performance of our algorithm.

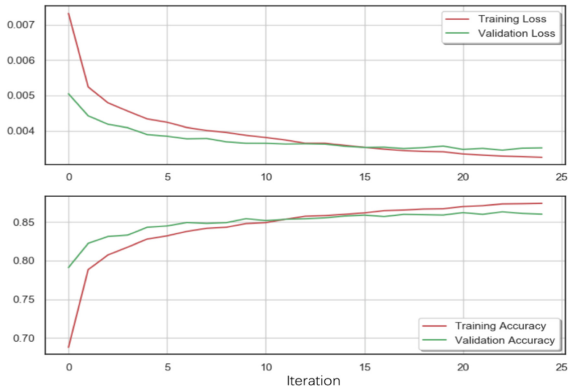

Fig. 4. Loss and accuracy of CNN-MixtureCorrentropy on the dataset with noise.

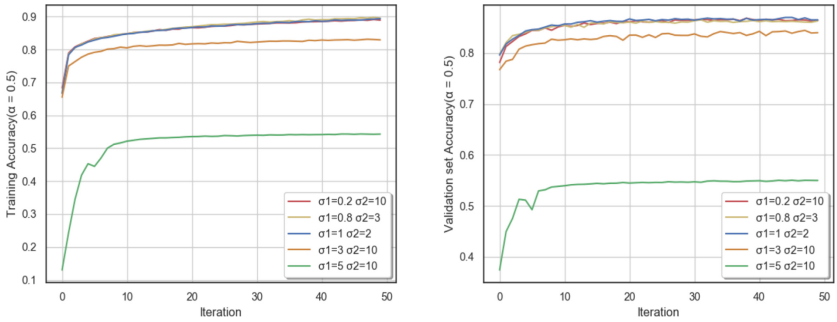

Fig. 5. Performance of CNN-MixtureCorrentropy ($\alpha = 0.5$).

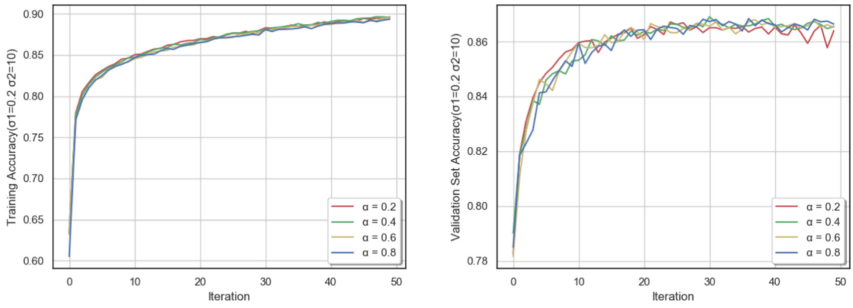

Fig. 6. Performance of CNN-MixtureCorrentropy ($\boldsymbol{\sigma}_1 = 0.2$, $\boldsymbol{\sigma}_2 = 10$).

4.2 Classification Results on a Real-World Malware Dataset

One of the most efficient ways in analyzing the malware data is to extract API call information [27]. The Windows API is a set of predefined Windows functions that control the behavior of various parts of Windows. Each action of the user triggers the

execution of one or more functions to tell Windows what is happening. API call information can be obtained statically and dynamically [28].

In this test task, we generate our dataset by randomly selecting malware samples from two malware datasets, Dasmalwerk [29] and VirusShare [30], and then use Cuckoo Sandbox to analyze malicious files. Dasmalwerk and VirusShare are the datasets of different types of executable malware from Internet. From the reports generated by Cuckoo, API call information is extracted (Fig. 7).

NtAllocateVirtualMemory	NtFreeVirtualMemory	NtAllocateVirtualMemory	GetFileType
NtAllocateVirtualMemory	SetErrorMode	LdrLoadDll	LoadStringA
LdrGetDllHandle	NtAllocateVirtualMemory	NtFreeVirtualMemory	NtAllocateVirtualMemory
GetSystemTimeAsFileTime	NtAllocateVirtualMemory	NtFreeVirtualMemory	NtAllocateVirtualMemory
GetFileAttributesW	NtCreateMutant	GetSystemTimeAsFileTime	NtOpenKey
FindResourceExW	LoadResource	FindResourceExW	LoadResource
GetFileAttributesW	NtCreateMutant	GetSystemTimeAsFileTime	NtOpenKey
GetSystemTimeAsFileTime	LdrGetDllHandle	LdrGetProcedureAddress	GetFileType
GetSystemTimeAsFileTime	LdrGetDllHandle	LdrGetProcedureAddress	GetFileType
NtAllocateVirtualMemory	SetErrorMode	LdrLoadDll	LoadStringA
GetSystemTimeAsFileTime	LdrGetDllHandle	LdrGetProcedureAddress	GetFileType
GetFileAttributesW	NtCreateMutant	GetSystemTimeAsFileTime	NtOpenKey
GetSystemTimeAsFileTime	LdrGetDllHandle	LdrGetProcedureAddress	GetFileType
LdrGetProcedureAddress	LdrGetDllHandle	NtAllocateVirtualMemory	GetFileType
LdrGetProcedureAddress	LdrGetDllHandle	LdrGetProcedureAddress	LdrGetDllHandle
SetErrorMode	NtCreateFile	NtAllocateVirtualMemory	SetFilePointer
GetSystemTimeAsFileTime	SetUnhandledExceptionFilter	NtAllocateVirtualMemory	CoInitializeEx
NtAllocateVirtualMemory	NtFreeVirtualMemory	NtAllocateVirtualMemory	GetFileType
GetSystemTimeAsFileTime	LdrGetDllHandle	LdrGetProcedureAddress	SetUnhandledExceptionFil
RegOpenKeyExA	NtClose	NtQueryAttributesFile	LoadStringA
MessageBoxTimeoutA	LdrGetDllHandle	LdrGetProcedureAddress	LdrGetDllHandle
GetSystemTimeAsFileTime	LdrGetDllHandle	LdrGetProcedureAddress	SetUnhandledExceptionFil
GetSystemTimeAsFileTime	NtAllocateVirtualMemory	NtFreeVirtualMemory	NtAllocateVirtualMemory
GetSystemTimeAsFileTime	LdrGetDllHandle	LdrGetProcedureAddress	GetFileType

Fig. 7. A part of malware API call report.

In this experiment, our proposed algorithm is specifically applied to binary classification task for malware. Aiming that distinguish the malware samples and normal benign sample, we need sufficient quantity of both malware samples and benign samples. We downloaded portable application from Internet and tested them by antivirus software. If the application is safe, we take it as a benign sample. Firstly, we randomly select the same amount of benign sample and malware sample, precisely, 200 samples of each class. As shown in Fig. 8, the word vector is input into CNN model and trained for multiple times. Dropout is used to prevent over fitting. The output of the model is the predicted label. Input size is (400, 995, 128), which refers to (sample number, maximum number of API calls, embedding). SVM is a classical and common classification method, thus, we take SVM as one of comparison algorithm. We use five-fold cross validation, the original data is randomly partitioned into 5 subsamples. For each time, 4 subsamples are used as train data, 1 subsample is used as test data, which means we take 320 samples as train set, 80 samples as test set. The process is then repeated 5 times. The final Accuracy (Acc) is defined as:

$$\text{Acc} = \frac{1}{5}\sum_{i=1}^{5} Acc_i \tag{10}$$

where Acc_i refers to the accuracy of each time.

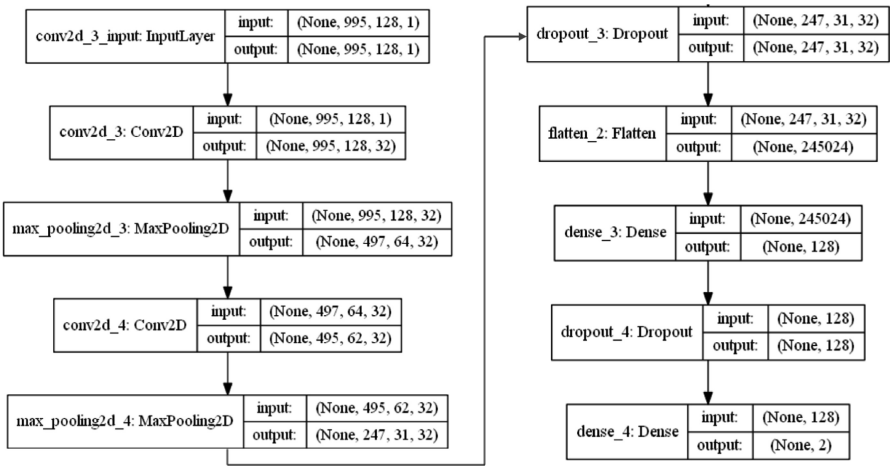

Fig. 8. Processing model.

Table 3. Performance of each algorithm on a real-world malware dataset.

Algorithm	Accuracy
SVM	0.7387
CNN-MSE	0.8025
CNN-Correntropy ($\sigma = 0.7$)	0.7937
CNN-MixCorrentropy ($\boldsymbol{\sigma}_1 = 0.4$ $\boldsymbol{\sigma}_2 = 3$ $\boldsymbol{\alpha} = 0.5$)	0.8156

The word2vec method is used to transfer text document into vectors. Table 3 shows the accuracy of each algorithm. CNN with different lose function perform better than SVM. Obviously, CNN with mixture correntropy loss function performs best. Because of the noise factors in the real-world malware dataset, our method shows the strongest robustness among four algorithms. Moreover, we also find that compared to MSE loss function, mixture correntropy loss function has faster convergence speed, demonstrated in Fig. 9.

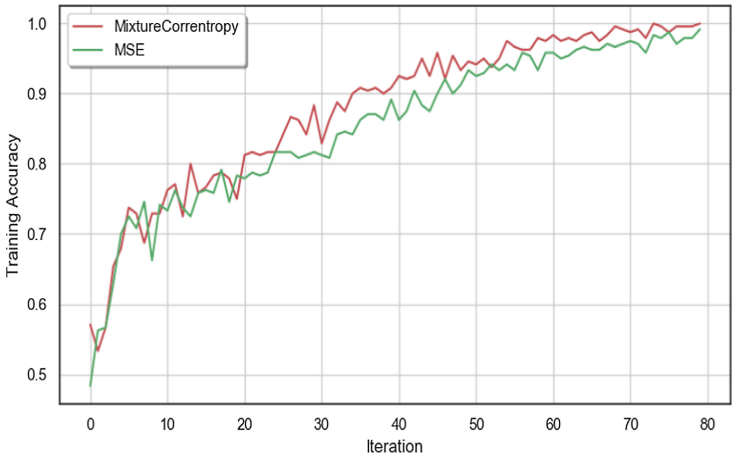

Fig. 9. Performance of convergence on training set.

5 Conclusion

This paper aims at dealing with a challenging issue in the achievement of malware identification and detection in CPS application, which is the Android malware classifier. Staring from the general analysis of related work on mixture correntropy and CNN, in this paper we present a novel malware identification and detection method on the basis of our proposed CNN classifier with mixture correntropy-induced loss function. Then, compared to other traditional classifiers, the data cloud be handled more flexibly and stably with a higher classification accuracy and a better robustness through the use of our mixture correntropy-based CNN model, due to the incorporation of mixture correntropy especially used to outlier learning problems. In the experiments, the classification performance of our proposed method and other popular algorithms are compared on a benchmark dataset and a real-world Android malware dataset. The experimental results verify the effectiveness and efficiency of our method.

Acknowledgement. This work was supported in part by the National Natural Science Foundation of China under grants U1836106 and U1736117, by the National Key Research and Development Program of China under grant 2018YFC0808306, and by the Beijing Intelligent Logistics System Collaborative Innovation Center under Grant BILSCIC-2019KF-08.

References

1. Lee, E.A.: Cyber physical systems: design challenges. In: 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC), Orlando, pp. 363–369. IEEE (2008)
2. Neuman, D.C.: Challenges in security for cyber-physical systems. In: DHS Workshop on Future Directions in Cyber-Physical Systems Security, pp. 22–24 (2009)

3. Stolfo, S.J., Wang, K., Li, W.-J.: Fileprint analysis for malware detection. In: ACM CCS WORM (2005)
4. Thakar, N., Praveen, K.A., Vikas, T.: System and method to detect domain generation algorithm malware and systems infected by such malware. U.S. Patent Application 16/264,667 (2019)
5. Taylor, C., Alves-Foss, J.: NATE: network analysis of a nomalous traffic events: a low-cost approach. In: Proceedings of the 2001 Workshop on New Security Paradigms, pp. 89–96. ACM, New York (2001)
6. Xiao, X., Zhang, S., Mercaldo, F., Hu, G., Sangaiah, A.K.: Android malware detection based on system call sequences and LSTM. Multimedia Tools Appl. **78**(4), 3979–3999 (2019)
7. Peiravian, N., Zhu, X.: Machine learning for android malware detection using permission and API calls. In: 25th International Conference on Tools with Artificial Intelligence, Herndon, pp. 300–305. IEEE (2013)
8. Yuxin, D., Zhu, S.: Malware detection based on deep learning algorithm. Neural Comput. Appl. **31**(2), 461–472 (2019)
9. Tobiyama, S., Yamaguchi, Y., Shimada, H., Ikuse, T., Yagi, T.: Malware detection with deep neural network using process behavior. In: 40th Annual Computer Software and Applications Conference (COMPSAC), Atlanta, vol. 2, pp. 577–582. IEEE (2016)
10. Liu, W., Pokharel, P.P., Principe, J.C.: Correntropy: a localized similarity measure. In: The 2006 IEEE International Joint Conference on Neural Network Proceedings, Vancouver, pp. 4919–4924. IEEE (2006)
11. Gunduz, A., Principe, J.C.: Correntropy as a novel measure for nonlinearity tests. Sig. Process. **89**(1), 14–23 (2009)
12. Luo, X., et al.: Towards enhancing stacked extreme learning machine with sparse autoencoder by correntropy. J. Franklin Inst. **355**(4), 1945–1966 (2018)
13. Chen, L., Qu, H., Zhao, J.: Generalized Correntropy based deep learning in presence of non-Gaussian noises. Neurocomputing **278**, 41–50 (2018)
14. Chen, B., Wang, X., Lu, N., Wang, S., Cao, J., Qin, J.: Mixture correntropy for robust learning. Pattern Recogn. **79**, 318–327 (2018)
15. LeCun, Y., Bengio, Y.: Convolutional networks for images, speech, and time series. In: The Handbook of Brain Theory and Neural Networks, vol. 3361, no. 10, p. 1995 (1995)
16. Principe, J.C.: Information Theoretic Learning: Renyi's Entropy and Kernel Perspectives. ISS. Springer, New York (2010). https://doi.org/10.1007/978-1-4419-1570-2
17. Liu, W., Pokharel, P.P., Príncipe, J.C.: Correntropy: properties and applications in non-Gaussian signal processing. IEEE Trans. Signal Process. **55**(11), 5286–5298 (2007)
18. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
19. Maturana, D., Scherer, S.: VoxNet: a 3D convolutional neural network for real-time object recognition. In: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, pp. 922–928. IEEE (2015)
20. Kim, Y.: Convolutional neural networks for sentence classification. In: EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, pp. 1746–1751 (2014)
21. Yuan, Z., Lu, Y., Wang, Z., Xue, Y.: Droid-Sec: deep learning in android malware detection. ACM SIGCOMM Comput. Commun. Rev. **44**(4), 371–372 (2014)
22. Kolosnjaji, B., Zarras, A., Webster, G., Eckert, C.: Deep learning for classification of malware system call sequences. In: Kang, B.H., Bai, Q. (eds.) AI 2016. LNCS (LNAI), vol. 9992, pp. 137–149. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-50127-7_11

23. Pascanu, R., Stokes, J.W., Sanossian, H., Marinescu, M., Thomas, A.: Malware classification with recurrent networks. In: 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brisbane, pp. 1916–1920. IEEE (2015)
24. Dahl, G.E., Stokes, J.W., Deng, L., Yu, D.: Large-scale malware classification using random projections and neural networks. In: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, Vancouver, pp. 3422–3426. IEEE (2013)
25. Singh, A., Pokharel, R., Príncipe, J.C.: The C-loss function for pattern classification. Pattern Recogn. **47**(2), 441–453 (2014)
26. Xiao, H., Rasul, K., Vollgraf, R.: Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747 (2017)
27. Shankarapani, M.K., Ramamoorthy, S., Movva, R.S., Mukkamala, S.: Malware detection using assembly and API call sequences. J. Comput. Virol. **7**(2), 107–119 (2011)
28. Idika, N., Mathur, A.P.: A survey of malware detection techniques, vol. 48. Purdue University (2007)
29. Dasmalwerk Homepage. https://dasmalwerk.eu. Accessed 10 Sept 2019
30. VirusShare Homepage. https://virusshare.com. Accessed 10 Sept 2019