Design of a Fused Triple Convolutional Neural Network for Malware Detection: A Visual Classification Approach

Santosh K. Smmarwar(✉), Govind P. Gupta, and Sanjay Kumar

Department of Information Technology, National Institute of Technology, Raipur, India
{sksmmarwar.phd2019.it,gpgupta.it,skumar.it}@nitrr.ac.in

Abstract. Detection of malware signatures from executable files requires effective signal processing and sandboxing operations, wherein the executable file is scanned for any malignant behavior. The recent malware detection techniques are based on static approaches that use machine and deep learning for analyzing malware signatures from byte and assembly-level program data. The byte-patterns are based on outliers, and the program-data is classified as a malware. These methods are not capable of detecting new variants of malware with long patterns of codes and huge dataset to classify the benign or malicious files. The issue with pattern analysis of large byte code dataset needs effective classification performance. To overcome these drawbacks, this paper has proposed a novel fused-triple convolutional neural network (fCNN) based framework for malware detection. This framework improves the accuracy of malware classification by converting the byte and assembly information into image data. This framework obtained more than 98% accuracy on the Microsoft Malware Dataset.

Keywords: Malware detection · CNN · VGGNet · Byte · Assembly · Image · Classification

1 Introduction

The exponential growth in the rise of new malware poses a critical threat to the internet, the user's integrity, privacy, confidentiality and valuable resources of computer systems. The malwares can cause a major loss to the organizations, societies and nations. Malware is a software code having malicious intentions to the computer systems, mobile environment and cloud service platform *etc.* various types of malware exists based on their functionalities like viruses, worms, Trojans, adware, spyware, Ransomware and many more *etc.* [1]. A large number of internet viruses are malwares which inject themselves onto the victim's device or machine and try to run malicious programs on the host. These programs would either try to interrupt the normal working of the host machine or steal valuable information from the machine which can be later used for blackmailing or invading the user's privacy [2]. In addition, these malwares can analyse activities over the keyboard to steal passwords and other sensitive information. In current scenarios of

M. Singh et al. (Eds.): ICACDS 2021, CCIS 1440, pp. 279–289, 2021.
https://doi.org/10.1007/978-3-030-81462-5_26

technological development traditional machine learning and deep learning based methods have achieved better performance in detection and classification of malware based on signature patterns to detect known malware. However, these techniques are expensive in terms of time, resources and feature engineering and require large dataset for better detection and classification of malware and unable to detect new variants of malware [3].

To detect and classify these malwares, various researchers have proposed different approaches in the literature. A survey of such approaches is presented in the next section. From this survey, it is observed that most of these approaches depend on short-term pattern analysis, which limits their performance and applicability to real-time datasets. Moreover, these techniques require large training data sets, which again limit the system applicability for applications where limited historical data is available. To overcome these drawbacks, this research work proposes a novel fused triple CNN approach, which initially converts the byte-level and assembly-level information of the programs into images. Next, these images are processed individually and in a combined form via a triple CNN VGGNet classifier. The statistical analysis indicates that the classifier has superior performance when compared with other single-program approaches and has the flexibility to be applied for cross-platform malwares. This text also indicates some of the future research that can be taken up to achieve cross-platform analysis for the malwares using a modified version of the CNN architecture. In this paper the main contribution of our work as given below.

- In this work, a novel fused triple convolutional neural network (fCNN) design is proposed for detection of malware.
- Proposed design is able to improve the accuracy of malware classification by converting the byte and assembly information into image data and then by using a VGGNet inspired CNN design to obtain more than 98% accuracy on Microsoft malware dataset.
- These images are processed individually and in a combined form via a triple CNN VGGNet classifier.
- The statistical analysis indicates that the classifier has superior performance when compared with other single-program approaches and has the flexibility to be applied for cross-platform malwares.

The remaining part of the paper consists of the following sections as follows Sect. 2 presents the related work, Sect. 3 indicates the design of proposed triple CNN architecture, Sect. 4 shows the result evaluation, Sect. 5 describes the brief about the dataset and finally Sect. 6 presents the conclusion and future scope of our work.

2 Related Work

The process of converting byte and assembly file data into imagery for malware classification has proven to be an effective approach. This can be observed from [4], wherein space filling curve mapping (SCFM) and Markov dot plot (MDP) have been used to convert the byte-file data into image data to identify decompression bomb attacks. The research work in [5] converts the opcodes and application programming interface (API) calls into word-to-vector features and obtains the derived names from these features. The approach is found to be approximately 1.3% better in terms of accuracy performance when compared to a long-short-term-memory (LSTM) based approach. The work in [6] evaluates structural and behavioural features from the malware files to classify polymorphic malwares. The work in [7] showcases that LSTM-based convolutional neural networks with transfer learning are most suited for this purpose. In [8], authors have suggested that Random Forests (RF) and k-Nearest Neighbours (kNN) classifiers are best suited for the purpose of malware classification. A specific file-access control vulnerability detector using machine learning is proposed in [9]. In [10] the byte files are converted into images, and an eXtreme Gradient Boosting (XGBoost) classifier is used to classify these images into malware categories. The work in [11] uses soft-relevance value (s-value) to find out the relevance of one-type of malware signatures with non-malware signatures. Another approach for detection of unknown malwares is described in [12]. This approach uses graph-embedding wherein the byte files are converted into eigenspace graphs using power iteration method. Malwares can be injected in systems using portable document files (PDFs). The work in [13] scans these PDF files for malware patterns by converting the PDF file byte data into directed graphs. A generic approach for detection of malware on mobile devices is mentioned in [14]. It uses a skip-gram model for detecting different kinds of malwares on Android devices. Another byte-to-image conversion approach that utilizes Markov image creation is mentioned in [15]. Here, the input byte-level data is converted into image format using bytes transfer probability matrices. An accuracy of 97% is achieved on Microsoft dataset, which can be improved by using better CNN architecture design that involves grouping of byte files with similar execution structure. Such a work can be observed in [16], which claims to improve the accuracy of classification to 99% by simply using linear classifiers like SVM, RF and kNN. The work in [15] and [16] can be combined to develop a highly accurate malware classification system. The work in [15] can also be extended using the work in [17], wherein random forests are used for classification of malware image data for improving the accuracy of classification to nearly 99%. Moreover, a multi-view learning method can be adopted for making the malware detection system applicable to any kind of operating system. This can be referred from the work in [18]. All these views are combined to form a master-feature vector and are given to a CNN classification engine. Due to the master feature vector, the final accuracy of the proposed system is nearly 99.6%, which is almost 1% higher than using only byte and operation code feature sets. This inspired the underlying research to develop a triple CNN architecture for obtaining better malware classification performance. The selection of a CNN engine for such a system design is also important. So, the work in [19] was referred. The use of image-based malware detection can also be confirmed from the work done in [20–22] and [23] which represent the byte-level and assembly level data into different visual formats and then applies machine learning methods to identify malware signatures. The work in [24] suggests the use of registry analysis and byte-level analysis as previously mentioned and combines it with machine learning approaches to obtain better classification performance even for cloud malwares.

To evaluate the underlying research a large-scale dataset is needed; therefore, this text uses the Microsoft Malware Classification dataset as described in [25]. This dataset consists of different kinds of files for each kind of OS operation. These operations are linked to malware activities, which assists in linking the files to malware activities. This dataset is also used in [26] and [27] for identification of malwares and prove that CNN-based approaches along with visual data representation are best suited for malware detection. The concept can be further extended to web-based malware detection and mobile-based malware detection.

Table 1. Microsoft malware dataset sample composition

Class ID	Malware family name	Number of samples	Type of attack
1	Ramnit	1487	Worms
2	Lollipop	2321	Adware
3	Kelihos_ver3	2880	Backdoor
4	Vundo	407	Trojan
5	Simda	42	Backdoor
6	Tracur	705	Trojan download malware
7	Kelihos_ver1	361	Backdoor
8	Obfuscation.ACY	1163	Any type of obfuscated malware
9	Gatak	983	Backdoor

3 Proposed Malware Detection Framework Using Fused-Triple CNN

In the proposed fused-Triple CNN based malware detection framework, both the byte file data and the assembly file data need to be converted into image data. Both set of image datasets are then merged to obtain a 3rd fused image. This 3rd image dataset is generated by appending the bytes from the assembly file to the byte file. Once these images are generated, then an individual CNN architecture is adopted to classify the images into one of 9 malware categories. These categories are listed in Table 1. The description of each of these steps can be observed in the following sub-sections, while the overall architecture diagram for the entire system can be observed from Fig. 1, wherein all the blocks and their connections are shown. The final fusion process evaluates the best of three selection processes to obtain the final malware class.

3.1 Conversion of Byte Files into Image Data

The byte files contain data in the form of operation codes, and operands. Each data is of one byte, and thus is directly added to an image array. Due to the variation in program size, each byte file has a different length. To normalize this length, the images are converted into 224×224 size using bicubic interpolation method as shown in Fig. 2.

3.2 Conversion of Assembly Files into Images

Assembly files contain the following attributes,

- Headers indicating addresses, read/write permissions, and other metadata
- Text attribute consisting of the text data needed by the assembly file
- RData consisting of the read data or operation codes in the assembly file
- IData consisting of import data, which indicates the different external programs imported by this assembly file
- Data consisting of memory and addressing data

From these data fields, the RData and IData fields are selected for image creation. This is due to the fact that other data which is added to the assembly file is verified by the OS and cannot contain any malicious code. While the data which will be continuously read and imported by the program might contain malicious code bytes. All these bytes are combined together to form the final image file using the same process as discussed in Sect. 3.1. The final image is converted into 224 × 224 sizes for normalization as depicted in Fig. 3.

3.3 Generation of the Fused Image

The fused image is created by adding the bytes from both byte file and assembly file one-after-the-other. All these bytes when combined create a new image file, which consists of both assembly and byte file patterns as shown in Fig. 4. The resulting image is converted into 448 × 448 sizes for normalization and further processing.

3.4 CNN Design for Byte File, Assembly File, and Fused File Classification

The VGGNet classifier is selected for classification of these images. Architecture of the VGGNet is illustrated in Fig. 5. The input layer is 224 × 224 × N in size, wherein 224 × 224 is the image size, while N is the number of images in the training set. A sample CNN architecture with 224 × 224 × N configuration can be observed from Fig. 5, wherein the input image is given to the CNN, and the CNN uses a rectilinear unit (ReLU) to convert this data into 112 × 112 form using Max pooling. This process is repeated 6 times, wherein the final features of size 7 × 7 × 512 are obtained. The final fully connected layer is capable of classifying the data into 1 of 'M' classes; in this case the value of M is kept as 9, due to the need of 9 malware classes at the output. Two such CNNs are deployed in order to classify the assembly file and the byte files individually.

For the fused file an additional convolutional + ReLU layer along with a Max pooling layer is attached in order to obtain the final class. This is done because the size of input imagery for byte and assembly files is twice the size of the fused file. These networks are trained, and the final class is obtained via fusion of the outputs of these networks.

In order to perform class fusion, the resulting outputs are merged in such that a manner, that if a minimum 2 out of 3 networks produce the same output, then the network agrees on the final result. If none of the outputs are the same, then the output is marked to be a combination of these malwares, and thus assists in discovery of new malwares from the system. Description of the dataset and result evaluation of the proposed framework is discussed in the next section.

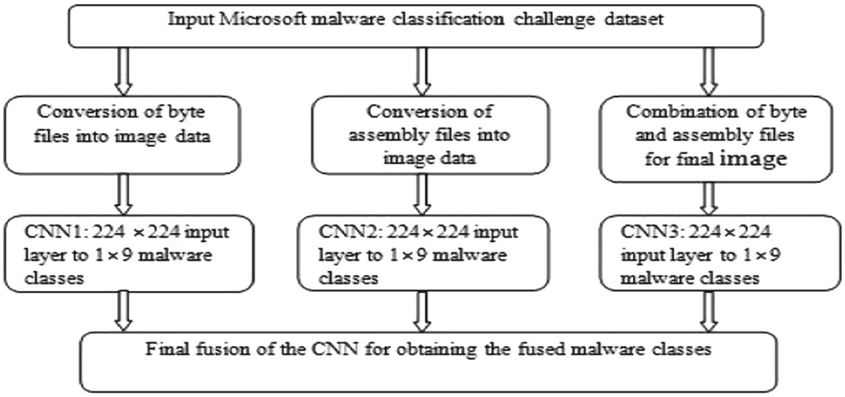
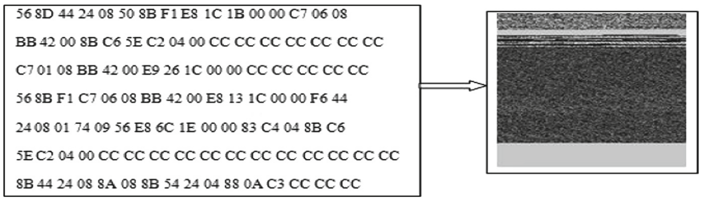
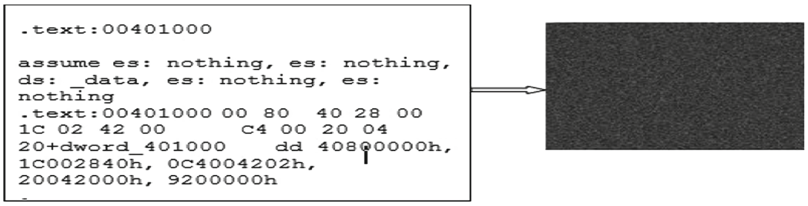

Input Microsoft malware classification challenge dataset
Conversion of byte files into image data
Conversion of assembly files into image data
Combination of byte and assembly files for final image
CNN1: 224 × 224 input layer to 1 × 9 malware classes
CNN2: 224 × 224 input layer to 1 × 9 malware classes
CNN3: 224 × 224 input layer to 1 × 9 malware classes
Final fusion of the CNN for obtaining the fused malware classes

Fig. 1. Proposed fused Triple CNN based malware detection Framework



```
56 8D 44 24 08 50 8B F1 E8 1C 1B 00 00 C7 06 08
BB 42 00 8B C6 5E C2 04 00 CC CC CC CC CC CC CC
C7 01 08 BB 42 00 E9 26 1C 00 00 CC CC CC CC CC
56 8B F1 C7 06 08 BB 42 00 E8 13 1C 00 00 F6 44
24 08 01 74 09 56 E8 6C 1E 00 00 83 C4 04 8B C6
5E C2 04 00 CC CC CC CC CC CC CC CC CC CC CC CC
8B 44 24 08 8A 08 8B 54 24 04 88 0A C3 CC CC CC
```

Fig. 2. Hexadecimal view of Byte files to image conversion



```
.text:00401000

assume es: nothing, es: nothing,
ds: _data, es: nothing, es:
nothing
.text:00401000 00 80  40 28  00
1C 02  42  00      C4  00  20  04
20+dword_401000    dd 40800000h,
1C002840h, 0C4004202h,
20042000h, 9200000h
```

Fig. 3. Views of assembly file to image conversion

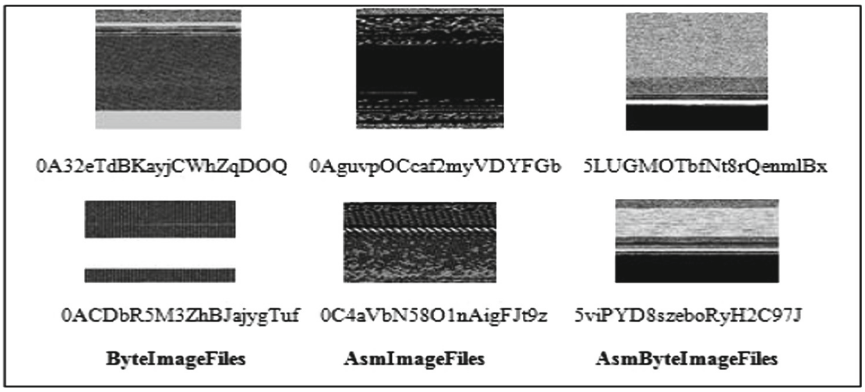


Fig. 4. Generation of the fused image files from byte files and asm files

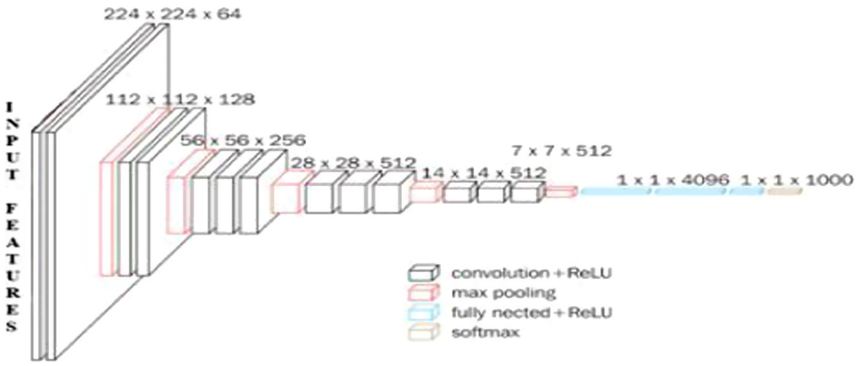


Fig. 5. Design of VGG16 architecture

4 Dataset Description

The proposed model used the Microsoft malware classification dataset BIG 2015 challenge from Kaggle provided by Microsoft. The dataset contains the 10,349 malware sample files of nine types of malware families shown in Table 1, each sample having the twenty-character class ID hash value of binary files and assembly files. The experiment is performed on windows 10 home, G7 core i9 10th Gen-(16 GB/1 TB SD/8 GB graphics/NVIDIA RTX 2070/300 Hz). The proposed model used python programming language with keras and Tensor flow packages.

5 Result Analysis

To evaluate the system, the entire Microsoft malware classification dataset is used. The dataset is divided into the following parts for training, testing and validation for incremental evaluation.

- 50:25:25
- 60:20:20
- 70:15:15
- 80:10:10

The classification results from each of these evaluations are tabulated using Table 2, wherein it can be observed that the network produces 99.9% classification accuracy even for 60:20:20 training, testing and validation split. Due to a combination of triple CNN, the entire dataset can be classified with highest possible accuracy with limited training set information. The selection of training set requires careful segregation of records such that the final data split is capable of classifying testing and validation sets with high accuracy.

These results when compared with other implementations also showcase superior performance in terms of accuracy of classification. This performance can be observed from Table 3, wherein five state-of-art techniques are used for comparison. From these

Table 2. Resulting accuracy of the proposed framework for each of the training, testing and validation splits

Train	Test	Validation	Accuracy (%)
50	25	25	95.40
60	20	20	99.90
70	15	15	99.91
80	10	10	99.98

Table 3. Accuracy comparison of proposed method with existing methods

Method	Accuracy (%)
TCN [5]	97.52
SCFM & MDP [4]	99.36
XGBoost [10]	99.77
HYDRA [3]	99.75
Soft relevance [11]	99.8
Proposed (fCNN)	99.98

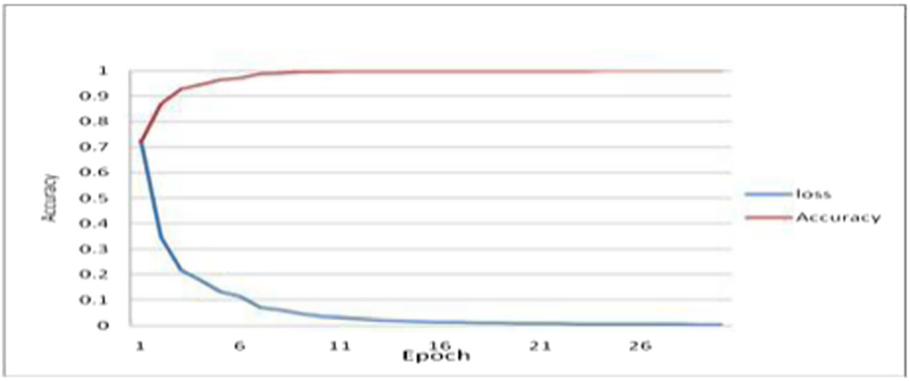


Fig. 6. Training loss and training accuracy comparison of our proposed model

results shown below it can be observed that the proposed framework is highly accurate and can be used for real-time malware classification on Microsoft malware classification dataset. Figure 6 showing the training loss, training accuracy and Fig. 7 showing the validation loss and validation accuracy of our proposed method.

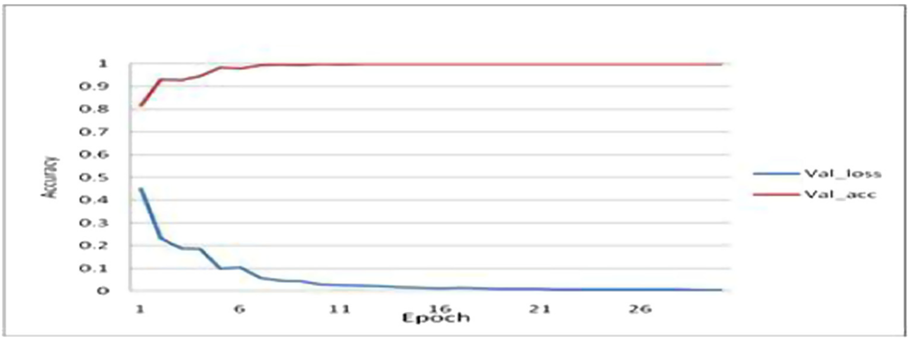


Fig. 7. Validation loss and validation accuracy comparison of our proposed model

6 Conclusion and Future Scope

This paper proposed a novel fused-triple convolutional neural network (fCNN)-based malware detection framework where first byte and assembly information are converted into image data and then VGGNet inspired fused-triple CNN is employed for classification of malwares. Performance evaluation of the proposed framework is evaluated using Microsoft malware dataset. Performance Comparisons of the proposed framework are done using five existing state-of-art techniques. Result analysis confirms the superiority of the proposed framework over the existing state-of-art framework. Proposed framework has achieved 99.98% accuracy. In future, the proposed framework is planned to be evaluated on other datasets like Android malware datasets, iOS malware datasets and network malware datasets. Converting byte data to images requires proper image sizing, which might create accuracy limitations while designing for larger datasets. To remove this limitation, the work can also be extended by addition of gated recurrent unit (GRU) and long-short-term-memory (LSTM) models, which will allow the system to skip byte to image conversion, and obtain the final classes with increased accuracy.

References

1. Namanya, A.P., Cullen, A., Awan, I.U., Disso, J.P.: The world of malware: an overview. In: 2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud), pp. 420–427. IEEE, August 2018
2. Aslan, Ö.A., Samet, R.: A comprehensive review on malware detection approaches. IEEE Access **8**, 6249–6271 (2020)
3. Gibert, D., Mateu, C., Planes, J.: HYDRA: a multimodal deep learning framework for malware classification. Comput. Secur. **95**, 101873 (2020)
4. Ren, Z., Chen, G., Lu, W.: Malware visualization methods based on deep convolution neural networks. Multimedia Tools Appl. **79**(15–16), 10975–10993 (2019). https://doi.org/10.1007/s11042-019-08310-9
5. Sun, J., Luo, X., Gao, H., Wang, W., Gao, Y., Yang, X.: Categorizing malware via a Word2Vec-based temporal convolutional network scheme. J. Cloud Comput. **9**(1), 1–14 (2020). https://doi.org/10.1186/s13677-020-00200-y

6. Masabo, E., Kaawaase, K.S., Sansa-Otim, J., Ngubiri, J., Hanyurwimfura, D.: Improvement of malware classification using hybrid feature engineering. SN Comput. Sci. **1**(1), 1–14 (2019). https://doi.org/10.1007/s42979-019-0017-9
7. Gibert, D., Mateu, C., Planes, J.: The rise of machine learning for detection and classification of malware: Research developments, trends and challenges. J. Netw. Comput. Appl. **153**, 102526 (2020)
8. Bae, S.I., Lee, G.B., Im, E.G.: Ransomware detection using machine learning algorithms. Concurr. Computat. Pract. Exp. **32**(18), e5422 (2020)
9. Lu, J., Gu, F., Wang, Y., Chen, J., Peng, Z., Wen, S.: Static detection of file access control vulnerabilities on windows system. Concurr. Comput. Pract. Exp., e6004 (2020). https://doi.org/10.1002/cpe.6004
10. Ahmadi, M., Ulyanov, D., Semenov, S., Trofimov, M., Giacinto, G.: Novel feature extraction, selection and fusion for effective malware family classification. In: Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy, pp. 183–194, March 2016
11. Zhang, Y., Liu, Z., Jiang, Y.: The classification and detection of malware using soft relevance evaluation. IEEE Trans. Reliab., 1–12 (2020). https://doi.org/10.1109/TR.2020.3020954
12. Hashemi, H., Azmoodeh, A., Hamzeh, A., Hashemi, S.: Graph embedding as a new approach for unknown malware detection. J. Comput. Virol. Hacking Tech. **13**(3), 153–166 (2016). https://doi.org/10.1007/s11416-016-0278-y
13. Singh, P., Tapaswi, S., Gupta, S.: Malware detection in PDF and office documents: a survey. Inf. Secur. J. Glob. Perspect. **29**(3), 134–153 (2020)
14. Egitmen, A., Bulut, I., Aygun, R., Gunduz, A.B., Seyrekbasan, O., Yavuz, A.G.: Combat mobile evasive malware via skip-gram-based malware detection. Secur. Commun. Netw. **2020**, article ID 6726147, 10 p. (2020). https://doi.org/10.1155/2020/6726147
15. Yuan, B., Wang, J., Liu, D., Guo, W., Wu, P., Bao, X.: Byte-level malware classification based on Markov images and deep learning. Comput. Secur. **92**, 101740 (2020)
16. Sahay, S.K., Sharma, A.: Grouping the executables to detect malware with high accuracy. arXiv preprint arXiv:1606.06908 (2016)
17. Roseline, S.A., Geetha, S., Kadry, S., Nam, Y.: Intelligent vision-based malware detection and classification using deep random forest paradigm. IEEE Access **8**, 206303–206324 (2020)
18. Darabian, H., et al.: A multiview learning method for malware threat hunting: windows, IoT and android as case studies. World Wide Web **23**(2), 1241–1260 (2020). https://doi.org/10.1007/s11280-019-00755-0
19. Khan, R.U., Zhang, X., Kumar, R.: Analysis of ResNet and GoogleNet models for malware detection. J. Comput. Virol. Hacking Tech. **15**(1), 29–37 (2018). https://doi.org/10.1007/s11416-018-0324-z
20. Zhang, Z., Cheng, Y., Gao, Y., Nepal, S., Liu, D., Zou, Y.: Detecting hardware-assisted virtualization with inconspicuous features. IEEE Trans. Inf. Forensics Secur. **16**, 16–27 (2020)
21. Bai, J., Shi, Q., Mu, S.: A malware and variant detection method using function call graph isomorphism. Secur. Commun. Netw. **2019**, article ID 1043794, 12 p. (2019). https://doi.org/10.1155/2019/1043794
22. Gao, X., Hu, C., Shan, C., Liu, B., Niu, Z., Xie, H.: Malware classification for the cloud via semi-supervised transfer learning. J. Inf. Secur. Appl. **55**, 102661 (2020)
23. Narouei, M., Ahmadi, M., Giacinto, G., Takabi, H., Sami, A.: DLLMiner: structural mining for malware detection. Secur. Commun. Netw. **8**(18), 3311–3322 (2015)
24. Tien, C.W., Huang, T.Y., Tien, C.W., Huang, T.C., Kuo, S.Y.: KubAnomaly: anomaly detection for the Docker orchestration platform with neural network approaches. Eng. Rep. **1**(5), e12080 (2019)
25. Ronen, R., Radu, M., Feuerstein, C., Yom-Tov, E., Ahmadi, M.: Microsoft malware classification challenge. arXiv preprint arXiv:1802.10135 (2018)

26. Sharma, S., Krishna, C.R., Sahay, S.K.: Detection of advanced malware by machine learning techniques. In: Ray, K., Sharma, T., Rawat, S., Saini, R., Bandyopadhyay, A. (eds.) Soft Computing: Theories and Applications. AISC, vol 742, pp. 333–342. Springer, Singapore (2019). https://doi.org/10.1007/978-981-13-0589-4_31
27. Ding, H., Sun, W., Chen, Y., Zhao, B., Gui, H. Malware detection and classification based on parallel sequence comparison. In: 2018 5th International Conference on Systems and Informatics (ICSAI), pp. 670–675. IEEE, November 2018