A Deep Learning Approach to Image-Based Malware Analysis

Chapter · March 2020

DOI: 10.1007/978-981-15-2414-1_33

3 authors, including:

Gurumayum akash Sharma
National Institute of Technology, Manipur
1 PUBLICATION **1** CITATION

SEE PROFILE

A Deep Learning Approach to Image-Based Malware Analysis

Gurumayum Akash Sharma, Khundrakpam Johnson Singh and Maisnam Debabrata Singh

Abstract Malicious software also referred to as "Malware" is one of the serious threats on the Internet today since it has been growing exponentially over the last decade according to research, causing substantial financial trouble to various organizations. Different security companies have been proposing different techniques to defend from this threat which is a major challenge on the complexity and growing volumes. Recently, malware communities and researchers have begun to apply machine learning and deep learning model to detect potential threats. We propose a malware classification model that takes advantage of the potential of deep learning (DL) models using the convolutional neural network (CNN) and combination of machine learning classifier with CNN such as support vector machine (SVM) for classifying their families. Detection of newly released malware using such models would be possible through mathematical function. That is, $f{:}n \rightarrow z$, where n is the given malware and z is their corresponding malware family. Malimg dataset is used to perform the experiment which contains malware image of 25 malware families and 9339 malware samples. CNN has outperformed the CNN-SVM with a test accuracy of 97.5%.

Keywords Artificial intelligence · Deep learning · Machine learning · Malware classification · Support vector machine · Convolutional neural networks

1 Introduction

Malicious software also referred to as "Malware" is one of the serious threats on the Internet today since it has been growing exponentially over the last decade according to research, causing substantial financial trouble to various organizations. Over 350,000 new malware and potentially unwanted applications (PUA) have registered every day as per the latest report of AV-TEST Institute [7]. According to AV-TEST

G. A. Sharma (✉) · K. J. Singh · M. D. Singh
Department of Computer Science and Engineering, NIT Manipur, Imphal, India
e-mail: g_akash@nitmanipur.ac.in

H. Das et al. (eds.), *Progress in Computing, Analytics and Networking*,
Advances in Intelligent Systems and Computing 1119,
https://doi.org/10.1007/978-981-15-2414-1_33

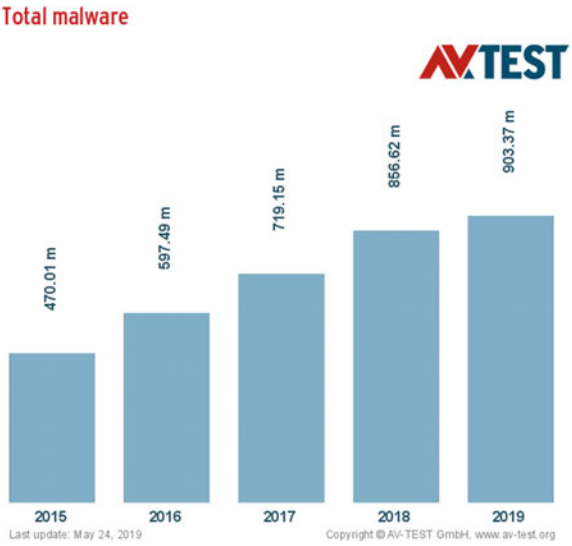

Fig. 1 Yearly survey report of malware (2015–2019) [7]

latest report, during this past 5 years, i.e., 2015–2019, it was observed that the number of malware reported has been increasing over 100 million in number each year, which is a major concern for the information security community. The detailed survey report is shown in Fig. 1.

Traditionally, pattern matching against signatures obtained from known malware was used for the detection of malware which can be easily defeated by many well-known strategies [14]. So, different security organizations have been proposing different techniques to defend from such threats which is a major challenge on the complexity and growing volumes of malware [11, 15]. In this paper, deep learning approach such as CNN and CNN-SVM will be used to perform the experiment for classification of different malware families using Malimg dataset which contains images of malware from 25 different families.

2 Related Works

Natraj et al. proposed a simple method of malware classification using a technique of visualizing and classify using image processing techniques [1, 12]. Malware binaries were converted to gray-scale images. For classification, malware code did not need to be executed or disassembled. For computing texture features, the GIST descriptor has been used which uses wavelet decomposition of an image. For classification, k-nearest neighbors with Euclidean distance have been used. For performance evaluation, bi-gram distributions were calculated from the raw data without disassembly. Classification accuracy of 0.98 was found using bi-gram distributions as a feature vector and it took 56 s to classify a sample in comparison to using the GIST feature,

which took only 1.4 s for the overall classification as feature vector length used was 320 but 65 K elements were used for the distribution-based analysis using bi-grams.

Yichuan Tang proposed a deep learning model that replaced the softmax activation function and minimized cross-entropy loss with a linear support vector machine [2]. Using L2-SVMs instead of softmax has shown a significant gain on popular deep learning datasets such as MNIST and CIFAR-10. Convolutional neural network (CNN) using SVM has a lesser error rate (11.9%) as compared to conventional convolutional neural network (CNN) using softmax which has an error rate of 14.0% on MNIST and CIFAR-10 dataset.

K. Kosmidis et al. proposed an automated framework for the identification of unknown vulnerabilities using current neural network techniques such as computer vision and image classification [3]. For feature engineering, malware binaries were converted to an 8-bit vector and then gray-scale images as preparation for new training set for the machine learning algorithm. Malimg dataset was used for all experiments such as training and testing of different classification algorithms. Using the nearest centroid, the average training time of 0.218 s and testing time of 0.0211 s had an average accuracy of 0.0856 which was the least accurate result. However, using the random forest, the average training time of 1.72 s and testing time of 0.0063 s had an average accuracy of 0.0916 which is the highest accuracy result, whereas using the classification algorithm, such as decision tree, stochastic gradient, perceptron, and multilayer perceptron have accuracy results of 0.088, 0.087, 0.0905, and 0.087, respectively. However, stochastic gradient and perceptron have misclassification.

Abien Fred et al. proposed a different deep learning model using the support vector machine (SVM) classifier for the classification of malware using the Malimg dataset [4]. The classification was done on different deep learning models such as multilayer perceptron (MLP), convolutional neural network (CNN), and gated recurrent unit (GRU). Empirical evidence had shown that GRU had outperformed the other deep learning models with a predictive accuracy of $\approx 84.92\%$.

Jiawei et al. proposed a lightweight approach for detecting distributed denial of service (DDoS) malware in IoT environments [5]. For classifying malware families using image recognition techniques, malware binaries were converted into grayscale images and fed into fine-tuned convolutional neural networks. The converted binary images were fed into the machine learning classifier into local devices, and a suspicious file was then submitted to a remote cloud server which was then used for further classification. However, the signature matching system has a large database as it contains details of each malware sample that are not efficient for IoT devices as IoT devices have limited resources. In the case of machine learning, only a small set of training data was needed for classification once trained. A small, two-layer shallow convolutional neural network has been used to have a lightweight detection system. The proposed system can achieve an average accuracy result of 94.0% on benign and malicious classification. To obtain more representative features from malware image, a new malware image extraction method may be needed.

3 Proposed Methodology

3.1 Machine Learning Library

Keras which runs on top of Google TensorFlow [13] is used to implement the deep learning algorithm, with the help of other libraries such as Matplotlib [8] which is used for plotting graph, numpy [10] which is used for scientific computing, and scikit-learn [9].

3.2 The Dataset

In this study, the Malimg dataset would be used for the evaluation of deep learning models which consists of 9,339 malware samples from 25 different malware families [1]. The frequency distribution of malware families and their variants in the Malimg dataset are shown in Table 1.

Malimg dataset was created by Nataraj et al. [1] by converting malware binaries into an 8-bit unsigned integer composing a matrix $M \in R^{m \times n}$. The matrix can be visualized as a grayscale image having values in the range of [0, 255], with 1 representing *white* and 0 representing *black* (Fig. 2).

3.3 Dataset Preprocessing

For preprocessing of the dataset such as the conversion of raw pixel to numpy array and generating labels, Keras preprocessing tools have been used [1]. The Malimg dataset has 25 classes so it is converted to the binary number of 0–24 that allows the representation of categorical data to more expressive; this process is also called as one hot encoding. Categorical data cannot be applied to the machine learning algorithm directly so it is be converted to numbers as shown in Table 1.

3.4 CNN Concepts

3.4.1 Input/Output Volumes

CNN is usually applied to image data. Every image is a matrix of pixels values. We encode each pixel as 8 bits. Each malware image is interpreted as [0, 255]. Thus the image develops a one-dimensional structure called the input volume ($64 \times 64 \times 1$) as the image we used is a grayscale image.

Table 1 Details of Malimg dataset

No	Family	Family name	No of variants
01	Backdoor	Agent.FYI	116
02	Backdoor	Rbot!gen	158
03	Dialer	Adialer.C	122
04	Dialer	Dialplatform.B	177
05	Dialer	Instantaccess	431
06	PWS	Lolyda.AA 1	213
07	PWS	Lolyda.AA 2	184
08	PWS	Lolyda.AA 3	123
09	PWS	Lolyda.AT	159
10	Rogue	Fakerean	381
11	Trojan	Alueron.gen!J	198
12	Trojan	C2Lop.P	146
13	Trojan	C2Lop.gen!G	200
14	Trojan	Malex.gen!J	136
15	Trojan	Skintrim.N	80
16	Trojan downloader	Dontovo.A	162
17	Trojan downloader	Obfuscator.AD	142
18	Trojan downloader	Swizzor.gen!E	128
19	Trojan downloader	Swizzor.gen!I	132
20	Trojan downloader	Wintrim.BX	97
21	Worm	Allaple.A	2949
22	Worm	Allaple.L	1591
23	Worm	VB.AT	408
24	Worm	Yuner.A	800
25	Worm:autoit	Autorun.K	106

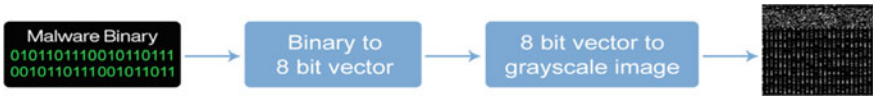

Fig. 2 Visualizing malware binary as a grayscale image

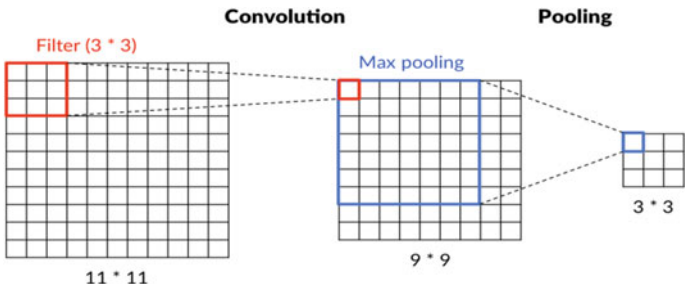

Fig. 3 Example of convolution and pooling operation diagram (source)

3.4.2 Features

A distinct and useful pattern obtained from the input data, i.e., image that helps in performing the desired image analysis is term as a feature. CNN gets the features from the input data which refers to the malware image in this study.

3.4.3 Filters (Convolution Kernels)

A filter which also refers as kernels is a small-size matrix in comparison to the input size of the image.

3.5 CNN Architecture

3.5.1 Convolution Layer

In our case, the filter size is $5 \times 5 \times 1$ and slide over input array with a stride of 1. In each step of the slide, each filter element and element of each subarea of the input array took the dot product. The result of each dot product is a scalar. Feature map is the total result of each unique position where the filter can be put on the image. For example, 64×64 unique position feature map is a $64 \times 64 \times 1$ array. We will get a smaller feature map if the stride is larger than 1.

3.5.2 The Pooling Layer

The pooling layer is used to reduce the spatial size progressively to reduce the number of features and computational complexity of the network. The main reason for the pooling layer is to avoid the model from overfitting. The most commonly used approach is max-pooling.

3.5.3 The Fully Connected Layer

Fully connected layers will perform high-level reasoning in the neural network after several convolutional and max-pooling layers. Just like regular artificial neural networks (non-convolutional), all activations in the previous layers are connected to neurons in fully connected layers.

3.6 Support Vector Machine

Vapnik originally developed support vector machines (SVMs) for binary classification [6]. To classify two classes in a given dataset with features $c \in R^m$, SVM will find the optimal hyperplane f(w, x) = w.x + b. SVM learns the parameters w and b by solving the following constrained optimization problem:

$$\text{Loss} = min\frac{1}{p}w^T w + C\sum_{i=1}^{p}\xi_i \tag{1}$$

Such that $\forall n\ w^T x_n t_n \geq 1 - \xi_n$

$$\forall n\ \xi_n \geq 0$$

where $w^T w$ is the Manhattan norm (also known as L1 norm), C is the penalty parameter (maybe an arbitrary value or a picked value using hyperparameter tuning), and ξ is the cost function.

$$loss = min\frac{1}{p}w^T w + C\sum_{i=1}^{p} max(0, 1 - y_i'(w^T x_i + b)) \tag{2}$$

This equation is known as L1-SVM, with the standard hinge loss. Its differentiable equivalent, L2-SVM (given by Eq. 3), provides more stable results [2].

$$loss = min\frac{1}{p}||w||_2^2 + C\sum_{i=1}^{p} max(0, 1 - y_i'(w^T x_i + b))^2 \tag{3}$$

where $||w||_2$ is the Euclidean norm (also known as L2 norm), with square hinge loss.

Despite intended for binary classification, SVM may be utilized for multinomial classification as well. The use of kernel tricks is one approach to accomplish this, which transforms a linear model to a nonlinear one by implementing kernel functions such as radial basis function (RBF). We then employed the one-versus-all (OvA) strategy, which treats a given class c_i as the positive class, and others as a negative class.

With OvA strategy, the L2-SVM serves as the classifier of the deep learning model in this study (CNN). That is, SVM learned the parameter weight and bias of the model.

3.7 Softmax

For classification problems using deep learning techniques, applying softmax or 1-of-K encoding at the top is the standard. For example, 25 classes in this study, the softmax layer has 25 nodes denoted by p_i, where $i = 1, 2, \ldots, 25$. p_i specifies a discrete probability distribution, therefore, $\sum_i^{25} p_i = 1$.

Let W be the weight connecting the penultimate layer to the softmax layer, h be the activation of the penultimate layer nodes, then the total input into a softmax layer, given by a, is

$$a_i = \sum_k h_k W_{ki} \tag{4}$$

Then we have

$$p_i = \frac{exp(a_i)}{\sum_j^{10} exp(a_j)} \tag{5}$$

The predicted class $\hat{i}$ would be

$$\hat{i} = arg_i \, max \; p_i = arg_i \, max \; a_i \tag{6}$$

3.8 Multiclass SVMs

Using the so-called one-versus-rest strategy, SVMs can extend for multiclass problems which seem to be the simplest way [6]. N linear SVMs will be trained independently, for N-class problems where the data from the other classes from the negative cases.

Denoting the output of nth SVM as

$$a_n(x) = w^T x \tag{7}$$

The predicted class is

$$arg_n max \; a_n(x) \tag{8}$$

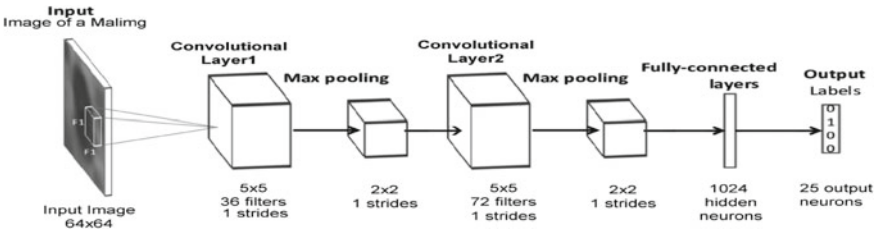

Fig. 4 CNN architecture

3.9 Proposed CNN Model

Convolutional neural network (CNN) has hidden layers of neurons with "learnable" parameters which make CNNs similar to feedforward neural networks. Those neurons receive input and perform a dot product then follow it with a nonlinearity such as *sigmoid* or *tanh* (Fig. 4).

The alteration proposed in the architecture design was the size of the input layer and output layer such as the input of $64 \times 64 \times 1$ and the output of 25 classes as shown in Fig. 3. And the introduction of SVM as the network classifier instead of conventional softmax function for the CNN-SVM model. Actually, Tang first presented this paradigm of combining CNN and SVM [2]. In this paper, we analyze the comparison of conventional CNN and CNN-SVM using the Malimg dataset.

4 Results and Discussion

In this study, all experiments were implemented using a personal computer with AMD Ryzen 5 2600x @ 3.5 GHz × 6, 16 GB of DDR4 RAM, and NVIDIA GeForce GTX 1050 ti 4 GB GPU.

Each deep learning (DL) model was trained on ≈70% of the preprocessed Malimg dataset [1]. The results for the different DL models are summarized in Table 2 where testing is done with ≈ 30% of the dataset. To evaluate our model like previous

Table 2 Experimental results of deep learning models

Variables	CNN	CNN-SVM
Train accuracy	97.58%	89%
Epochs	25	25
Data points	2798	2798
F1	0.96	0.86
Precision	0.96	0.83
Recall	0.97	0.89

methods, we use accuracy which indicates the percentage of malware samples labeled correctly in the test data.

We train our proposed model CNN and CNN-SVM for 25 epochs with a batch size of 256 with Malimg dataset. Table 2 shows the performance of different methods on this dataset. Our method CNN achieves the predictive accuracy of 97.58%, whereas CNN-SVM has predictive accuracy of 89%. However, the CNN model turns out to have the best performance with an accuracy of 97.58%.

Figures 5 and 6 show the experimental training and testing accuracy curves of the deep learning models (CNN and CNN-SVM) for 25 epochs (equivalent to 625 steps, since $6400 \times 25 \div 256 = 625$). The graph of learning curve for all the models shown in figures is plotted using Matplotlib [8].

Figure 7 shows the testing performance of CNN model in multinomial classification on malware families. The mentioned model has precision of 0.96, recall of 0.97, and F1 score of 0.96.

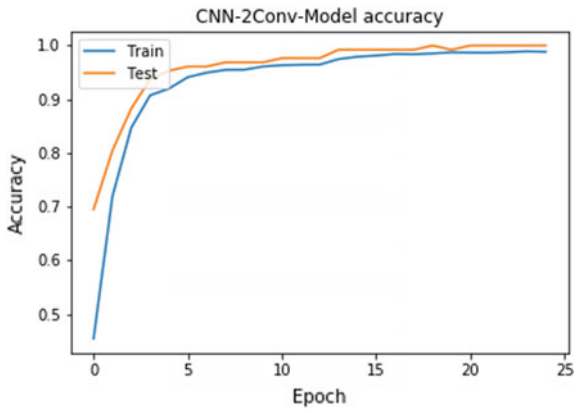

Fig. 5 Accuracy curve of CNN showing training and testing accuracy according to epoch

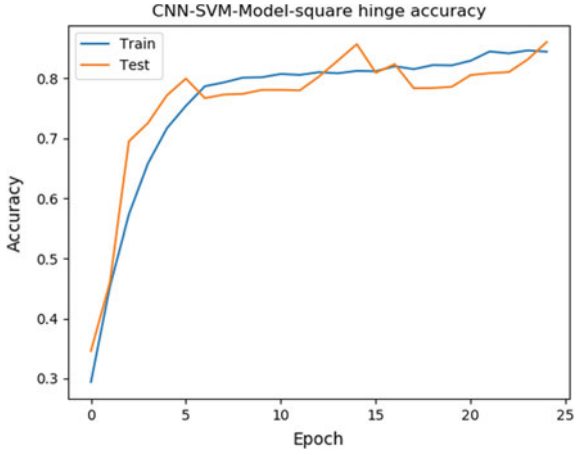

Fig. 6 Accuracy curve of CNN-SVM showing training and testing accuracy according to epoch

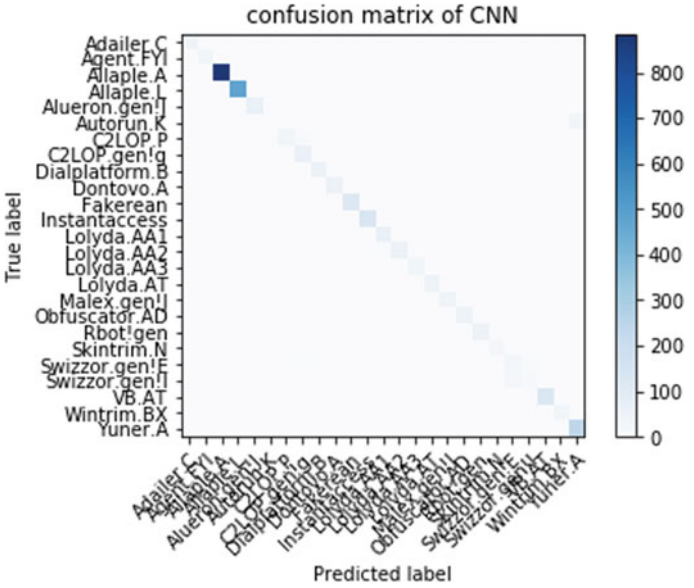

Fig. 7 The figure showing Confusion matrix of CNN

Figure 8 shows the testing performance of CNN-SVM model using SVM as final classifier. The mentioned model had a precision of 0.83, a recall of 0.89, and F1 score of 0.86.

Comparison of proposed methods with various related works done in the survey with Malimg dataset in terms of methods and accuracy is shown in Table 3, while our methods (CNN) have achieved the best performance among all the others.

5 Conclusion and Future Work

Malimg dataset is used for the experiment in this paper, which consists of malware images for the purpose of malware family classification [1]. We implemented different deep learning models with L2-SVM, and softmax as their final layer in the multinomial classification task. The experimental results show that CNN using the softmax model had the highest predictive accuracy among other deep learning models (CNN-SVM), having a test accuracy of $\approx 97.58\%$.

Improving the architectural design of the deep learning models by adding more layers and adding better non-linearities and/or using an optimized dropout may get a better result on malware classification. Other deep learning models such as MLP and RNN for image classification can be applied to experiments on malware classification.

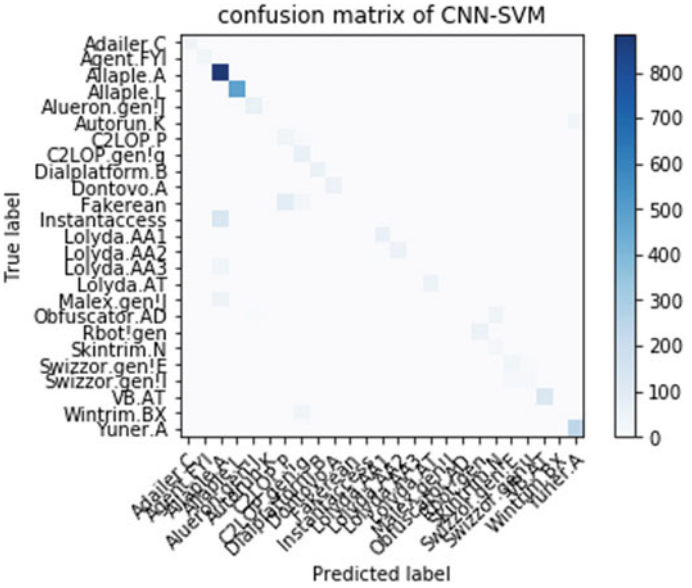

Fig. 8 The figure showing Confusion matrix of CNN-SVM

Table 3 Comparison of accuracy results for Malimg dataset

Paper	Methods	Accuracy
Kosmidis et al. [3]	MLP	≈ 87%
Agarap et al. [4]	GRU-SVM	≈ 84.92%
Agarap et al. [4]	MLP-SVM	≈ 80.46%
Agarap et al. [4]	CNN-SVM	≈ 77.23%
Proposed method	CNN	≈ 97.58%
Proposed method	CNN-SVM	≈0 89%

References

1. Nataraj, L., Karthikeyan, S., Jacob, G., Manjunath, B.: Malware Images: Visualization and Automatic Classification (2011)
2. Tang, Y.: Deep Learning using Linear Support Vector Machines (2013)
3. Kosmidis, K., Kalloniatis, C.: Machine Learning and Images for Malware Detection and Classification (2017)
4. Agarap, A.F., Pepito, F.J.H.: Towards Building an Intelligent Anti-Malware System: A Deep Learning Approach using Support Vector Machine (SVM) for Malware Classification (2017)
5. Su, J., Danilo Vasconcellos, V., Prasad, S., Daniele, S., Feng, Y., Sakurai, K.: Lightweight classification of IoT malware based on image recognition. In: 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), pp. 664–669. Tokyo (2018)
6. Cortes, C., Vapnik, V.N.: Support vector networks. Mach. Learn. (1995)
7. AVG-Test Institute: (2019). https://www.av-test.org/en/statistics/malware/. Accessed 20 May 2019

8. Hunter, J.D.: Matplotlib: A 2D graphics environment. Comput. Sci. Eng. **9**(3), 90–95 (2007). https://doi.org/10.1109/MCSE.2007.55
9. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**(2011), 2825–2830 (2011)
10. van der Walt, S., Colbert, S.C., Varoquaux, G.: The NumPy array: a structure for efficient numerical computation. Comput. Sci. Eng. **13**(2), 22–30 (2011)
11. Vinod, P., Jaipur, R., Laxmi, V., Gaur, M.: Survey on malware detection methods. In: Proceedings of the 3rd Hackers' Workshop on Computer and Internet Security, pp. 74–79 (2009)
12. Nataraj, L., Yegneswaran, V., Porras, P., Zhang, J.: A Comparative Assessment of Malware Classification Using Binary Texture Analysis and Dynamic Analysis, pp. 21–30 (2011)
13. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jozefowicz, R., Jia, Y., Kaiser, L., Kudlur, M., Levenberg, J., ManÃl', D., Schuster, M., Monga, R., Moore, S., Murray, D., Olah, C., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., ViÃl'gas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: large-scale machine learning on heterogeneous systems Software (2015). Available from www.tensorflow.org
14. Gandotra, E., Bansal, D., Sofat, S.: Malware analysis and classification: a survey. J. Inf. Secur. **5**, 56–64 (2014)
15. Udayakumar, N., Saglani, V.J., Cupta, A.V., Subbulakshmi, T.: Malware classification using machine learning algorithms. In: 2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI), pp. 1–9. Tirunelveli (2018)