Malware Detection and Classification

Using Convolutional Neural Networks

(December 2020)

Egehan ORTA, Hasan Hüseyin KAÇMAZ, Mürüvvet BOZKURT, Yiğitcan ÇOBAN

Abstract—Each day, countless new malware is created. Classic antivirus scanners are using signatures of software to detect malware. In that case, antivirus producers need to analyze malware to add their signature to their database. Since it is virtually impossible to analyze all malware and it is easy to trick classic antivirus scanners using various techniques, it is necessary to come up with a solution without supervising every file one by one. We believe that, currently the best fit for that approach is using ML algorithms to detect malware. We examined various academic papers to create an insight about different kinds of solutions that academicians offered. During that research we concluded that using convolutional neural networks (CNN) is a consistent and easy to implement solution for that purpose. Using convolutional neural networks, we have created a new way to detect malware types. In this paper, we examined the success rate of using the classification algorithm to detect malware.

Index Terms— Malware Detection, Machine Learning, Image Recognition, Convolutional Neural Network, Deep Learning

---

- *Egehan Orta Author is with the Computer Engineering Department, Istanbul Technical University, ortae16@itu.edu.tr*
- *Hasan Hüseyin Kaçmaz Author is with the Computer Engineering Department, Istanbul Technical University, kacmazh17@itu.edu.tr*
- *Mürüvvet Bozkurt Author is with the Computer Engineering Department, Istanbul Technical University, bozkurtm16@itu.edu.tr*
- *Yiğitcan Çoban Author is with the Computer Engineering Department, Istanbul Technical University, cobany16@itu.edu.tr*

1 Introduction

In the digital world, fraud has become a problem just like the real world. Therefore, in the new cyber world, people have to face new threats that they need to deal with. Since the internet is a space that everyone can wander around freely, communications between people are increasing. As a result, there are plenty of hackers who are trying to steal others' information. Hackers are using malicious software as a hacking vector to reach our information. That malicious software is called *malware*. Hackers are gaining the ability to execute their algorithms on users' computers using malware they devised. They are able to steal information as well as manipulate others' computers to do whatever they want. As a result, there are huge security issues related to malware problem.

Just like there are security forces in the real world to protect people against threats, there is an equivalent of it called *antiviruses*. Antiviruses ensure people surf safely on the internet by minimizing the fraud rate.

For malware detection, old antiviruses that we use today are using software signatures to detect malware [18]. This method basically creates a signature from a part of the code in any software and looks it up from a database that has been created by the antivirus producer by investigating malware manually. This old way cannot reach the speed of production of new malware. In 2016, more than 7,100,000 of new malware have been detected and that number is 47% higher than the previous year [1]. To bypass this technique, hackers are generating several malware which act the same as others. Therefore, there are hundreds of thousands of malicious software that have identical functionality but with different signatures. It has been seen that malware are using techniques called polymorphism to not get caught by old fashioned antiviruses by changing their software's signatures with these techniques [14]. Given these points, it is clear that there is a need for a new approach to detect malware. Most commonly debated way to do it is using machine learning and deep learning algorithms.

Artificial Intelligence methods are gaining popularity day by day and they are being used to solve various problems, including cyber security. One of the branches of AI is machine learning. Machine learning can process the data previously given to it and learn the pattern to predict the next step. By using these algorithms, systems can create rules to calculate results of certain states. Deep learning models are calculating results of inputs by using ANN algorithms which have more layers than traditional machine learning algorithms to create continuous learning schemes. Another feature of deep learning is using hidden layers to understand much more complex states than regular machine learning models [3].

In this paper we will be using Convolutional Neural Networks (CNN) to detect and classify malware. CNN is a neural network which consists of convolutional layers for filtering. CNN is widely used for image processing and recognition. After the filters, isolated inputs are sent to the neural network. The neural network computes the data and produces an output.

This method has been used widely by other researchers. Vinayakumar et al. have used both images and static analysis data as an input to their CNN algorithm in their ScaleMalNet [2]. Another example is, in their work Huang et al. created images from static and dynamic analysis data. In this process, they have colored the image by categorizing Application Programming Interface calls of the Windows system. This hybrid analysis approach caused their work to produce better results as they have stated with 91% detection accuracy [4].

Our solution is using images as inputs as well. We have produced images of malicious software by analyzing their hex codes. By using hex codes, we have produced grayscale [23] images of the malware then trained our model with those images. So as an input our model accepts images of malware as well.

It is important to detect malware with efficient ways since it is a huge threat to the cyber world. Classical algorithms are not sufficient enough to detect all of the new malware so it is necessary to devise new ways. Therefore, solutions like machine learning algorithms are on their way to be the norm of cyber security. With our work, we aimed to show the performance of malware classification algorithms on malware detection domain and we tried to evaluate our CNN model.

2 Related Works

We were able to find various kinds of work in this particular area since it is a popular topic and the need for new technology is in high demand.

Sewak, Sahay and Rathore have used objdump of files and deep neural networks to create a malware detection system. They have used around 11.000 malicious files and 2800 clean files. By using auto encoder as their feature extractor and 4 level Deep Neural Networks, their system reached 99,21% accuracy rate with 0,12% false positive rate. Interestingly their deeper Deep Neural Networks had a worse result [5].

On another work, researchers have used CNN just like our research to detect malicious PDF files. For that experiment 19830 malicious files and 116695 benign files have been used. They have experimented on classification of malicious files with CNN too. They have created 2 models for detection algorithms. Models are different from each other in terms of layer count and complexity. Model A was simpler. It consists of a single convolutional layer, followed by a global max pool and finishing with a linear layer and a sigmoidal gate. Model B also had a fully connected layer. Just like Sewak, Sahay and Rathore's results, a somewhat simpler layer had almost the same or better results against a complicated layer. Their convolutional neural network also did better than against antiviruses they have tested in terms of classification [6].

Xiang et al. proposes a new method based on deep learning and visualization of malware on malware detection topic. They, firstly, generate several images from static analysis. Then, they run all of these malicious or benign files on CuckooBox which is a sandbox, works on Windows Platforms and they get reports generated from CuckooBox and turn into images. Lastly, they merge 2 images into 1 image which contains static visualization image and dynamic analysis image. They trained their model with these hybrid images and got 94.7% accuracy rate [7].

Yeo et al. proposed a malware detection method that consists of investigation of network flow. They have downloaded 5 types of malware (Neris, rbot, Virut, Murlo, NSIS) to train their model and a clean state to compare them. They have used convolutional neural network (CNN), multi-layer perceptron (MLP), support vector machine (SVM) and random forest (RF) algorithms. Their

experiments resulted with superiority of CNN and RF algorithms in terms of both accuracy and performance [8].

Choudhary and Sharma point out that machine learning algorithms have more potential than traditional malware detection algorithms. They reached out at a 93% rate to figure out if an executable file is malware or not with K-Nearest Neighbors and Support Vector Machine algorithms. On the other hand, Naive Bayes algorithm gives 53% rate to detect malware [9].

Zhao et al. proposed a solution to detect malware by using both dynamic and static analysis. They got features from static analysis with Portable Executable Files and Application Programming Interface call sequence from dynamic analysis. Then they used SVM and Naive Bayes to classify the executable files with 97% accuracy rate [10].

Vinayakumar et al. has created some composite systems that are using multiple machine learning algorithms to increase its performance. They have evaluated generic algorithms firstly in their paper. Results showed us that the best Deep Neural Networks and Convolutional Neural Networks did the best work. Convolutional Neural Networks and Deep Neural Networks accomplished 98,8% and 99% accuracy respectively. Researchers also stated that deep learning algorithms were better than classical machine learning algorithms. Lastly, they have formed images of files similar to our work and the best results for classification was the product that system. End product of this paper was a complex program which uses static, dynamic analysis and image processing. [2]

Zhang proposed that the multi-layer perceptron neural network model gave good results to detect PDF-based malware. This algorithm uses stochastic gradient descent on the backpropagation algorithm for updating the model. This model works better than eight well-known antiviruses with a 95% true positive rate [11].

Vasan et al. has created a system to detect malwares that targets IoT malware. Their model can detect ARM, x86_64, i386, MIPS malware. Their work has accomplished an astonishing 99.98% accuracy on ARM architecture. They constructed their system using two deep learning algorithms, Recurrent Neural Network and Convolutional Neural Network. They analyzed sequences of opcodes and the binary itself distinctly and pipelined their outputs to an Advanced Ensemble Learning method [15]. Another paper which proposes a similar architecture is Lu, Li and Zhu's process-based malware detection method [20].

Past works show that malware detection using artificial intelligence has started to be used recently. Thus, it also shows that deep learning algorithms can be integrated into the cyber security domain too. After researching, we concluded that convolutional neural network algorithms, which is a sub-method of deep learning algorithms, can be used for malware detection problems. Work of Xiang et al. is a clear example of this. They have converted Windows Application Programming Interface call routines to RGB images and use this input for their CNN based model [7]. We created images from binary malware files and produced a highly precise CNN model.

The basis of our thinking was that if we can use CNN for both detection and classification separately, we can use it for both simultaneously as well. Our work differentiates from other works from that point. Our model takes clean files with malware as input so that we can accomplish detection alongside classification.

3 Proposed Solution

In the light of all the articles we reviewed, using the Convolutional Neural Network to detect malware has caught our attention. The dataset that we used in our model which is described on part 4, was already in image form. However, we want to test our model by adding some benign software. In order to do that, we read benign software as binary and convert all lines to hexadecimal strings. Then we read these hexadecimal files and convert these hexes into grayscale images. Related function's pseudo code can be seen in Figure 1.

```
function ConvertToImage(directory, name)

    a, b is integer values

    for each binary in directory do

        for each line in binary do

            INSERT line[:16] to array_binary

        array_image <- array_binary

        a           <- sqrt(array_image.height)

        a           <- square((log(a) / log(2)) + 1)

        b           <- (array_image.height * 16) / a

        array_image <- array_image[:a*b // 16, :]

        array_image <- reshape.array_image

        image       <- to_array(array_image)

        save.image
```

Fig. 1. Pseudo code of conversation binary data to grayscale image.

To balance the dataset among distinct malware families, various methods are used. The bat algorithm is used to data equilibrium approach or a dynamic resampling method etc. [16]. Besides, the bat algorithm combined with the CNN is also used to improve the accuracy of the model. Additionally, NSGA-II approach resolves balance problems of the dataset [17]. In our solution, we trained our model with weighted classes instead of using these kinds of methods due to unbalanced data.

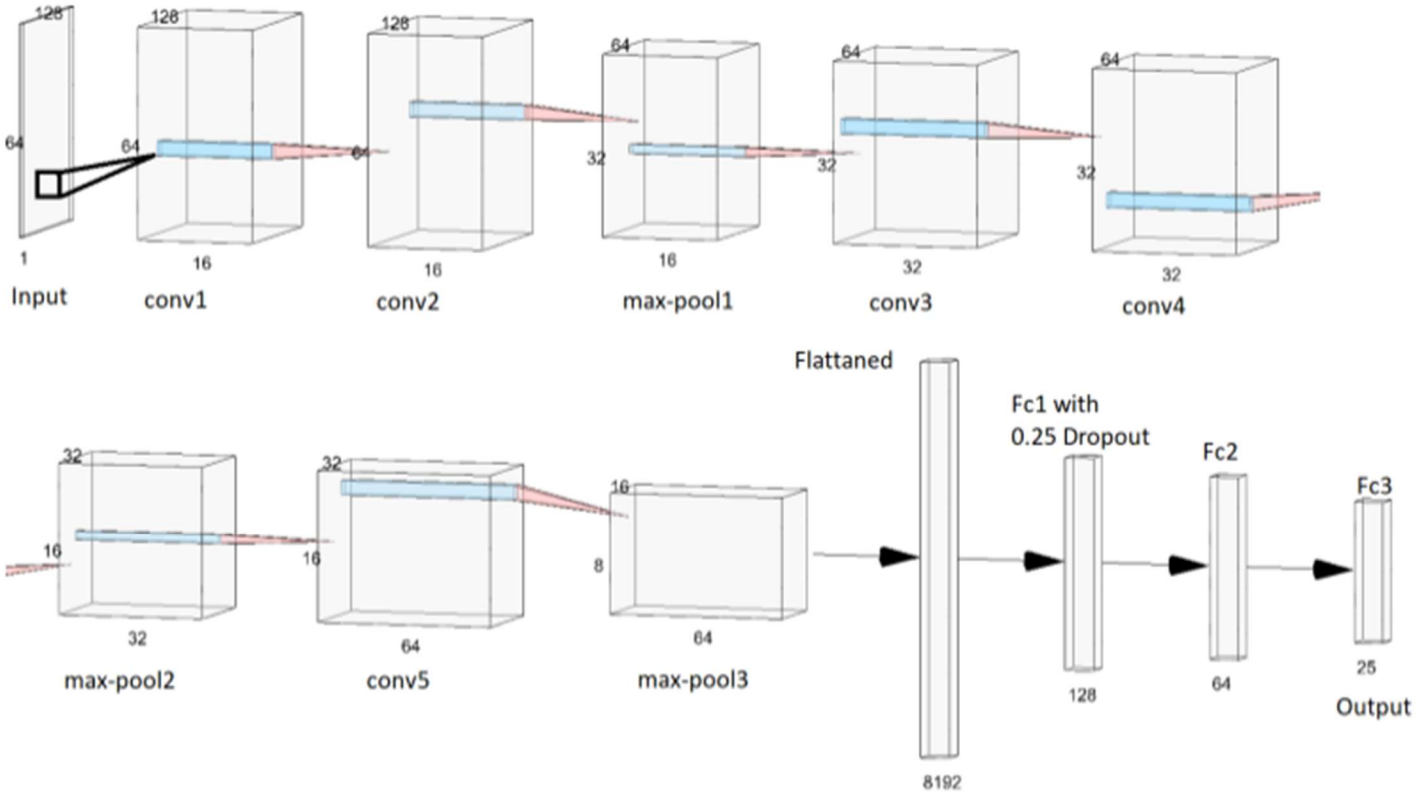


Fig. 2. Train dataset split of classes. The distribution of train dataset which belongs to Malware Classification Model.

After data generation processes, images are read and they are converted to 128x64 grayscale labeled images. Subsequently, 20% of total images are selected randomly as a test dataset. The rest 80% data is separated as validation and training dataset. For the validation dataset we used 20% of the rest data, which is 16% in total. Finally, the training dataset is created using the rest which is 64% in total.

We tested our dataset in numerous CNN models. As we observed, complex models with more convolutional layers than ours have generally performed worse than our model. Also, there were some good models which performed nearly the same as our model, but since they had more features and were more complex than our model we did not select them. Rather than complex models, we tried simpler ones and we see that performance is at very low levels in simpler models. Comparing all these models we have tried, we selected a model which has 5 convolutional layers connected to a neural network that has 2 depths. An example for a 5 convolutional layer malware detection model which uses CNN is FireEye [19].

Our selected model takes 128x64 grayscale images as input. In the first two convolutional layers, kernel with size 3x3 convolution applied with the same padding for 16 filters by using ReLu as activation function. Then a max pooling layer with 2x2 window size and 2x2 strides is applied. After max pooling, two convolution layers with kernel size 3x3 applied with the same padding for 32 filters by using ReLu as activation function. A max pooling layer with 2x2 window size and 2x2 strides followed these convolution layers again. As a last convolution layer, 3x3 convolution applied with the same padding for 64 filters by using ReLu as activation function. After that data is flatted and an 8192x1 vector is sent to the neural network. The neural network has three layers. First and second layers have a 25% dropout rate between them, second and third layers are fully connected. First layer of neural network has 128 neurons with ReLu activation function, second layer has 64 neurons with ReLu activation function and the last layer has 25 neurons with Softmax activation function for malware classification and 26 neurons with Softmax activation function for malware and clean data classification. Visual representation can be seen at Figure 2.

Total trainable parameters of the model were 1.093.449 at the end. We selected Adam Optimizer as model optimizer and Categorical Cross Entropy as loss function for model's accuracy.

4 DATASET

We used MalImg Dataset for this model [12] [13]. This dataset has 9339 grayscale images belonging to 25 different malware classes. Also, we created 578 images as clean data. Clean data is created with randomly selected Linux and Windows system files and scripts.

For our malware classification model, we split the data as train, validation and test 64%, 16% and 20% respectively. At the end malware classification data was split into 5976 train samples, 1495 validation samples and 1868 test samples. The distribution of first model's train, validation and test dataset can be seen in Figure 3, Figure 4 and Figure 5 respectively.

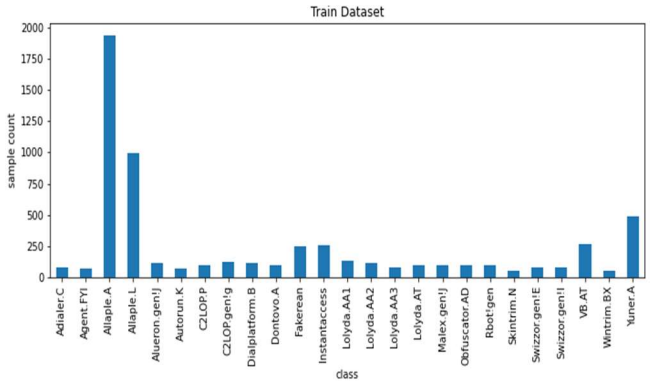


Fig. 3. Train dataset split of classes. The distribution of train dataset which belongs to Malware Classification Model.

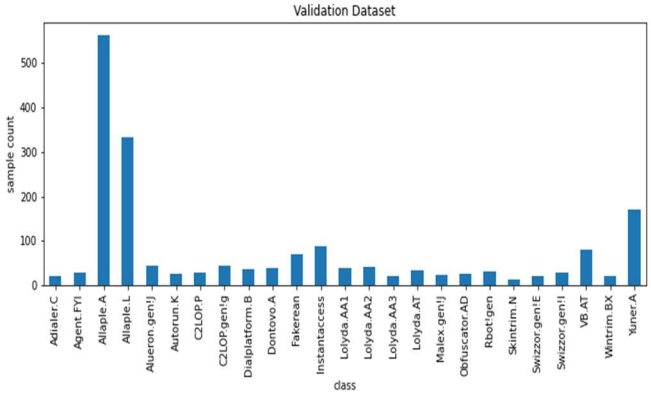


Fig. 4. Validation dataset split of classes. The distribution of validation dataset which belongs to Malware Classification Model.

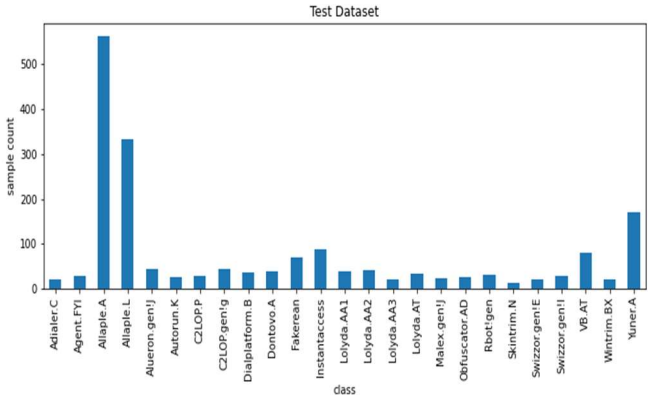


Fig. 5. Test Dataset split of classes. The distribution of Test dataset which belongs to Malware Classification Model.

For our malware and clean data classification model we split the data as train, validation and test 64%, 16% and 20% respectively. At the end malware classification data was split into 6356 train samples, 1591 validation samples and 1964 test samples. The distribution of first model's train, validation and test dataset can be seen in Figure 6, Figure 7 and Figure 8 respectively.

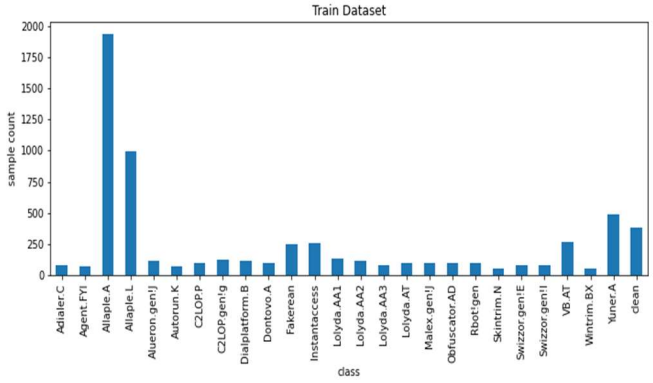


Fig. 6. Train dataset split of classes. The distribution of train dataset which belongs to Malware and Clean Data Classification Model.

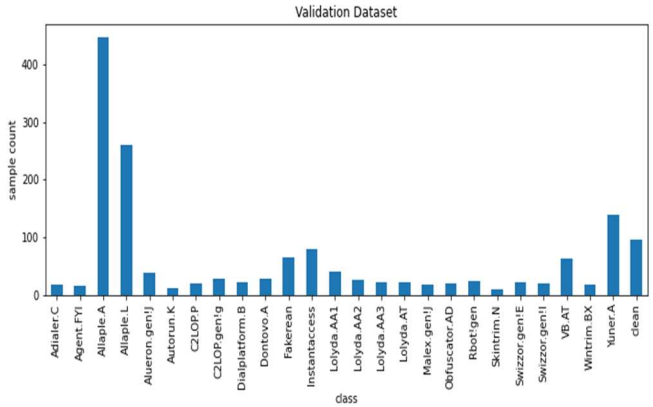


Fig. 7. Validation dataset split of classes. The distribution of Validation dataset which belongs to Malware and Clean Data Classification Model.

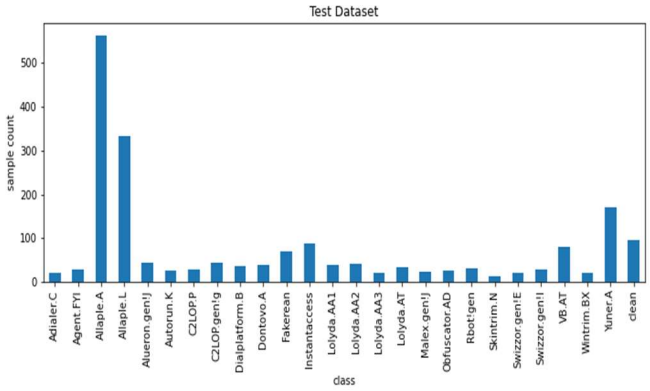


Fig. 8. Test dataset split of classes. The distribution of test dataset which belongs to Malware and Clean Data Classification Model.

The whole dataset of our model consists of 26 different classes including clean samples. Some samples from our dataset can be seen in Figure 9.

As we described before, we used 2 different datasets for our models. One of them has got only 25 class of malwares to measure the performance of model on malware classification. Other one has got an additional class named clean which we generated. Total number of samples can be clearly seen in Table 1.

TABLE 1
TOTAL NUMBER OF SAMPLES FOR MODELS

	Malware Classification	Malware and Clean Data Classification
Train	5976	6356
Test	1868	1964
Validation	1495	1591

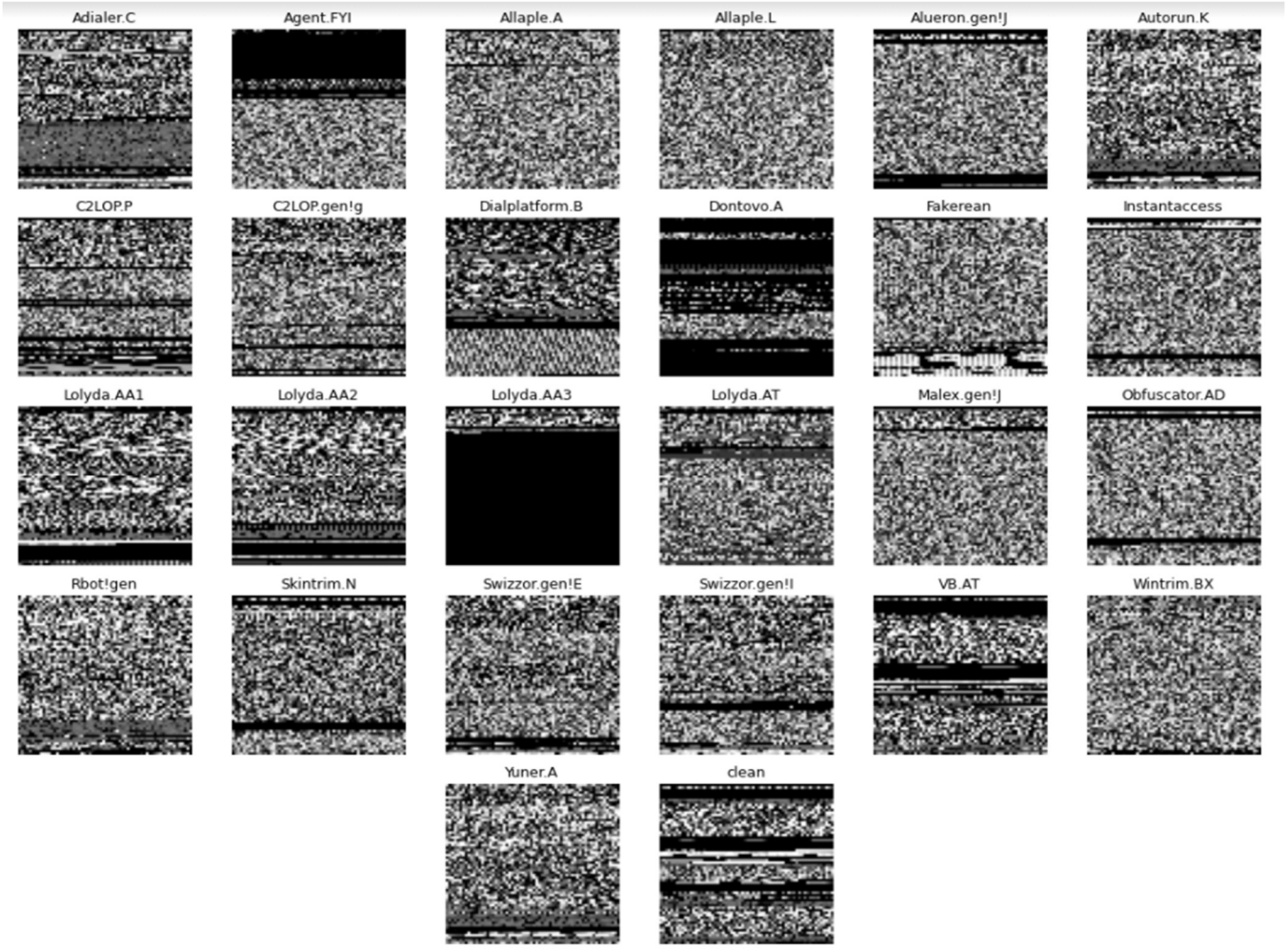


Fig. 9. Sample images that belongs to various classes of our dataset.

Finally, total number of samples that belongs to the classes and detailed overview of our dataset with malware family names and types can be seen in Table 2.

TABLE 2
MALWARE DATASET SPECIFICATIONS

	Family / Class	Type	Count
1	Adialer.C	Dialer	122
2	Agent.FYI	Backdoor	116
3	Allaple.A	Worm	2949
4	Allaple.L	Worm	1591
5	Alueron.gen!J	Worm	198
6	Autorun.K	Worm:AutoIT	106
7	C2LOP.P	Trojan	146
8	C2LOP.gen!g	Trojan	200
9	Dialplatform.B	Dialer	177
10	Dontovo.A	Trojan Downloader	162
11	Fakerean	Rogue	381
12	Instantaccess	Dialer	431
13	Lolyda.AA1	PWS	213
14	Lolyda.AA2	PWS	184
15	Lolyda.AA3	PWS	123
16	Lolyda.AT	PWS	159
17	Malex.gen!J	Trojan	136
18	Obfuscator.AD	Trojan Downloader	142
19	Rbot!gen	Backdoor	158
20	Skintrim.N	Trojan	80
21	Swizzor.gen!E	Trojan Downloader	128
22	Swizzor.gen!I	Trojan Downloader	132
23	VB.AT	Worm	408
24	Wintrim.BX	Trojan Downloader	97
25	Yuner.A	Worm	800
26	clean	clean	578

5 Results and Analysis of Proposed Solution

For malware classification, we trained this model with 25 epochs and weighted classes due to unbalanced data. Our training loss was ~0.02 and training accuracy was ~0.99 at the end of 25th epoch. Our validation loss was ~0.10 and validation accuracy was ~0.98 at the end of 25th epoch. Training and validation loss growth can be seen in Figure 10 and accuracy growth can be seen in Figure 11.

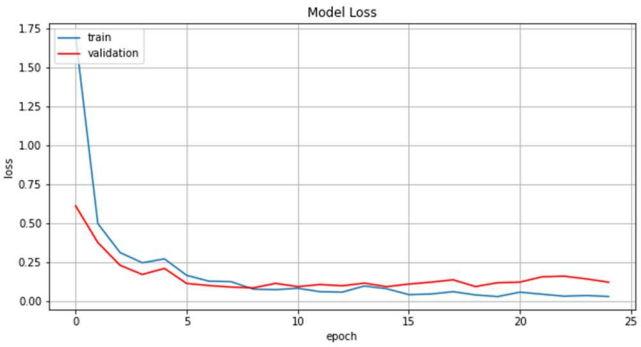


Fig. 10. Change in validation and train loss. Line graph of progression of validation and train loss as epoch increases.

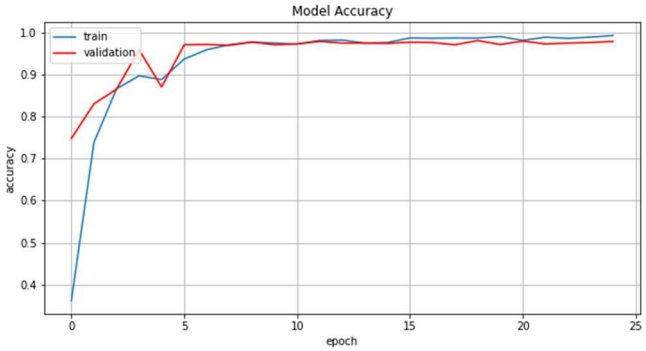


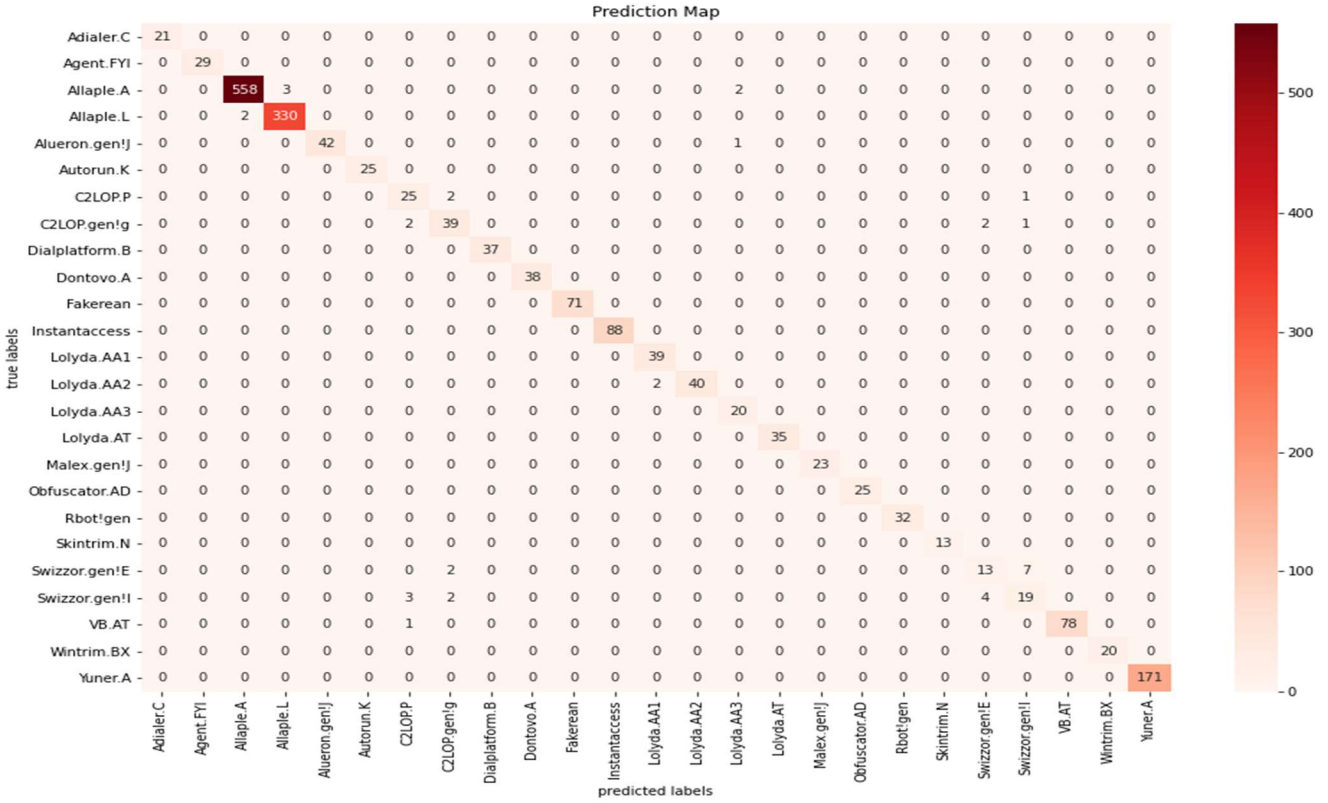
Fig. 11. Change in validation and train accuracy. Line graph of progression of validation and train accuracy as epoch increases.

After the training part we tested our model. The test results were very close to the validation scores. This model performed with loss 0.0835 and with accuracy 0.9802 in testing. According to test results 37 samples are misclassified while 1458 samples are correctly predicted. In Table 3, true labels and predicted labels of the test dataset can be seen.

Upon investigating, we realized that most of the false decisions that model made were between the same types of malware. Since the same type of malware has similarities, the model made wrong decisions. If we subtract errors among the same type of malware, which is 12, from total error count, which is 37, we will have 15 errors left from 1495 samples. The data can be seen on Table 3.

TABLE 3

Prediction Counts of Each Class for Malware Classification



For malware and benign classification, we trained model with 25 epochs and weighted classes to avoid unbalanced classes. The training loss was ~0.02 and training accuracy was ~0.99 at the end of 25th epoch. The validation loss of model was ~0.09 and accuracy was ~0.98 at the end of 25th epoch. Training and validation loss growth can be seen in Figure 12 and accuracy growth can be seen in Figure 13.

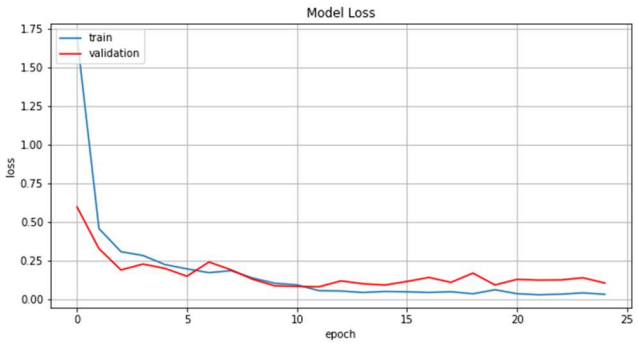


Fig. 12. Change in validation and train loss. Line graph of progression of validation and train loss as epoch increases.

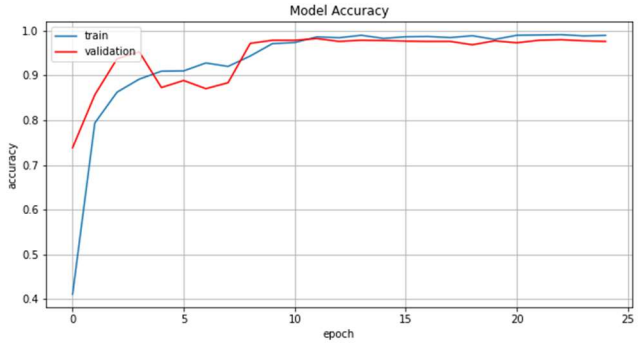


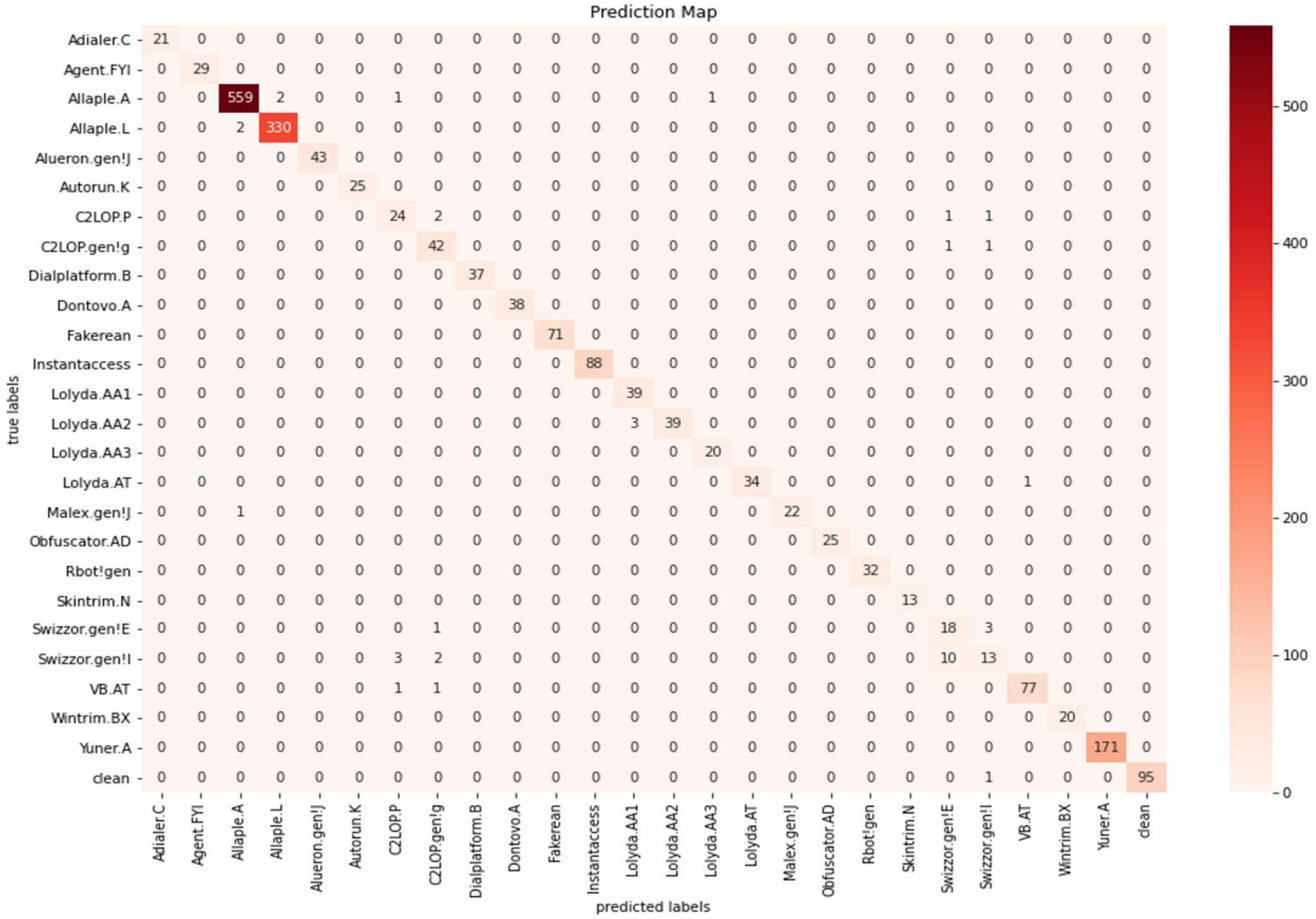
Fig. 13. Change in validation and train accuracy. Line graph of progression of validation and train accuracy as epoch increases.

After the training part we tested our model. Test results for this model were as good as malware classification. This model performed with loss 0.0841 and with accuracy 0.9801 in testing. According to test results 39 samples are misclassified while 1925 samples are correctly predicted. In Table 4, true labels and predicted labels of the test dataset can be seen.

If we examine Table 4, the same type of error occurrences happened there as well. Same types of malware families have been misclassified in several events. If we subtract that number of errors from the actual error count, we will get a total error count of 19 instead of 39. Also, only 1 clean software has been misclassified as malware.

TABLE 4

PREDICTION COUNTS OF EACH CLASS FOR MALWARE AND CLEAN DATA CLASSIFICATION



6 Conclusion

In this work, we have proposed a new way to detect malwares instead of today's antiviruses. Contrary to classical antivirus software which are using software signatures to detect malicious code parts, our system does not need software signatures. In this new product, we converted malware files to the images using their binary values and then created a model which is using convolutional neural network algorithms. Convolutional neural networks are widely used for image classification and object detection, yet we have achieved both malware detection and classification with this method we devised. We converted malware to images by using hexadecimal output to create grayscale images. Our work has achieved 98,8% success rate in both detection and classification. Other works show that, by adding system calls and opcodes to the structure of the algorithm, there will be more precision to detect malwares [21] [22]. In the future, we are aiming to strengthen our algorithm by creating images with colors based on opcodes thus convolutional neural network algorithms can be more precise.

References

[1] de Lima, S.M.L., Silva, H.K.d.L., Luz, J.H.d.S. *et al.* Artificial intelligence-based antivirus in order to detect malware preventively. *Prog Artif Intell* (2020). https://0-doi-org.divit.library.itu.edu.tr/10.1007/s13748-020-00220-4

[2] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran and S. Venkatraman, "Robust Intelligent Malware Detection Using Deep Learning," in *IEEE Access*, vol. 7, pp. 46717-46738, 2019, doi: 10.1109/ACCESS.2019.2906934.

[3] J. Ferdaos, C. Vaya, A. Bhalla, A. Tharayil and M. El Barachi, "Smart Malware Detection: From Signatures to Artificial Intelligence," *2020 5th International Conference on Smart and Sustainable Technologies (SpliTech)*, Split, Croatia, 2020, pp. 1-6, doi: 10.23919/SpliTech49282.2020.9243841.

[4] Huang, X., Ma, L., Yang, W. *et al.* A Method for Windows Malware Detection Based on Deep Learning. *J Sign Process Syst* (2020). https://doi.org/10.1007/s11265-020-01588-1

[5] Sewak, Mohit & Sahay, Sanjay & Rathore, Hemant. (2018). An investigation of a deep learning based malware detection system. 1-5. 10.1145/3230833.3230835.

[6] Fettaya, Raphael & Mansour, Yishay. (2020). Detecting malicious PDF using CNN.

[7] Huang, Xiang & Ma, Li & Yang, Wenyin & Zhong, Yong. (2020). A Method for Windows Malware Detection Based on Deep Learning. Journal of Signal Processing Systems. 1-9. 10.1007/s11265-020-01588-1.

[8] M. Yeo *et al.*, "Flow-based malware detection using convolutional neural network," *2018 International Conference on Information Networking (ICOIN)*, Chiang Mai, 2018, pp. 910-913, doi: 10.1109/ICOIN.2018.8343255.

[9] S. Choudhary and A. Sharma, "Malware Detection & Classification using Machine Learning," *2020 International Conference on Emerging Trends in Communication, Control and Computing (ICONC3)*, Lakshmangarh, Sikar, India, 2020, pp. 1-4, doi: 10.1109/ICONC345789.2020.9117547.

[10] Zhao, Jingling & Zhang, Suoxing & Liu, Bohan & Cui, Baojiang. (2018). Malware Detection Using Machine Learning Based on the Combination of Dynamic and Static Features. 1-6. 10.1109/ICCCN.2018.8487459.

[11] Zhang, Jason. (2018). MLPdf: An Effective Machine Learning Based Approach for PDF Malware Detection.

[12] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath. 2011. Malware images: visualization and automatic classification. In *Proceedings of the 8th International Symposium on Visualization for Cyber Security* (*VizSec '11*). Association for Computing Machinery, New York, NY, USA, Article 4, 1–7. DOI: https://doi.org/10.1145/2016904.2016908

[13] MalImg Dataset: https://www.dropbox.com/s/ep8qjakfwh1rzk4/malimg_dataset.zip?dl=0

[14] Mohammed, Mohssen & Chan, H Anthony & Ventura, Neco & Hashim, Mohsin & Bashier, Eihab. (2011). An Automated Signature Generation Approach for Polymorphic Worms Using Factor Analysis. International Journal of Information Security. 1. 45-52.

[15] D. Vasan, M. Alazab, S. Venkatraman, J. Akram and Z. Qin, "MTHAEL: Cross-Architecture IoT Malware Detection Based on Neural Network Advanced Ensemble Learning," in *IEEE Transactions on Computers*, vol. 69, no. 11, pp. 1654-1667, 1 Nov. 2020, doi: 10.1109/TC.2020.3015584.

[16] Z. Cui, F. Xue, X. Cai, Y. Cao, G. Wang and J. Chen, "Detection of Malicious Code Variants Based on Deep Learning," in IEEE Transactions on Industrial Informatics, vol. 14, no. 7, pp. 3187-3196, July 2018, doi: 10.1109/TII.2018.2822680.

[17] Cui, Zhihua & Du, Lei & Wang, Penghong & Cai, Xingjuan & Zhang, Wensheng. (2019). Malicious code detection based on CNNs and multi-objective algorithm. Journal of Parallel and Distributed Computing. 129. 10.1016/j.jpdc.2019.03.010.

[18] M. Alazab, S. Venkataraman and P. Watters, "Towards Understanding Malware Behaviour by the Extraction of API Calls," 2010 Second Cybercrime and Trustworthy Computing Workshop, Ballarat, VIC, 2010, pp. 52-59, doi: 10.1109/CTC.2010.8.

[19] Bose S, Barao T, Liu X. Explaining AI for Malware Detection: Analysis of Mechanisms of MalConv. *2020 International Joint Conference on Neural Networks (IJCNN), Neural Networks (IJCNN), 2020 International Joint Conference on*. July 2020:1-8. doi:10.1109/IJCNN48605.2020.9207322

[20] Lu S (1,2), Li Q (1), Zhu X (1). Stealthy Malware Detection Based on Deep Neural Network. Journal of Physics: Conference Series. 1437(1). doi:10.1088/1742-6596/1437/1/012123

[21] H. Kim, J. Kim, Y. Kim, I. Kim, K. J. Kim, and H. Kim, "Improvement of malware detection and classification using api call sequence alignment and visualization,"Cluster Computing, pp. 1-9, 2017. [Online]. Available: http://dx.doi.org/10.1007/s10586-017-1110-2

[22] J. Saxe, D. Mentis, and C. Greamo, "Visualization of shared system call sequence relationships in large malware corpora,"Proc. Int. Symp. Visual.Cyber Security. ACM, Seattle, WA, United states, 2012, pp. 33-40. [Online]. Available: http://dx.doi.org/10.1145/2379690.2379695

[23] Nataraj, L., Karthikeyan, S., Jacob, G., & Manjunath, B. S. (2011, July). Malware images: Visualization and automatic classification.In Proceedings of the 8th international symposium on visualization for cyber security (pp. 1-7).