Image-based Malware Detection Using Deep-learning Approaches

Mohammad Hadi Alaeiyan[a], Pouria Lakzian[1]

[a]*Comp, , Earth, , ,*

Abstract

With the advent of the Internet of Things (IoT), the threat of malware and cyber-attacks is growing day by day. Considering advances in IoT, our personal belongings and gadgets can connect to and transmit data through the internet. Since the Internet is now available to almost everyone these days, the safety of our digital persona is of great importance. Given the rapid growth in complexity and sophistication of malware files, traditional approaches to malware detection no longer ensure our digital safety. In recent years, with the remarkable advances in AI, researchers have begun to address this threat with newer approaches, such as machine learning algorithms; more recently, deep learning techniques have been applied to the field of malware detection with promising results. In this paper, we used the Grad-CAM technique on the malware images to gain insight into how the black-box CNN models make predictions on this type of data. We also proposed a CNN architecture designed based on our findings from the Grad-CAM results that outperforms existing models for malware detection and classification, and compared it with well-known CNN models all of which were trained and tested on malware image datasets such as Malimg, Dumpware10, MaleVis, and Binary. The architecture proposed in this paper achieved a significant overall accuracy of 99.58% on the Malimg dataset which was higher than the proposed CNN in the research of Puneeth et al, which had an overall accuracy of 99.26% on the malimg dataset.

Keywords: Deep learning, Classification, Malware, Dynamic analysis, Memory dump image

1. Introduction

In recent years, the progression of malware has posed a significant cyber-security threat, necessitating the development of advanced detection and classification techniques. Traditional malware detection methods, which often rely on signature-based approaches, have proven inadequate in the face of increasingly sophisticated and rapidly evolving malware variants(11). Consequently, there has been a growing interest in leveraging deep learning techniques to enhance malware detection and classification.

A transformation approach (13) converts malware binaries into images, enabling the application of image classification techniques to identify and categorize malware, which capitalizes on the ability of convolutional neural networks (CNNs) to extract and learn hierarchical features from image data, thus facilitating the accurate classification of malware samples.

Malware analysis can be performed in two manners: static and dynamic, both of which are carried out in this paper. The static analysis as shown inFigure 1 is the conversion of malware binaries from portable executable files into 8-bit vectors which then form the pixels of each image(12). In the dynamic analysis, the image is obtained by capturing the malware memory dump and converting it to an image.

In this paper, we propose a novel CNN architecture specifically designed for malware image classification and test it on both static and dynamic analysis approaches. Our architecture aims to improve classification accuracy by incorporating advanced techniques such as residual connections, and attention techniques. Through extensive experiments and evaluations, we show that our proposed CNN is able to segregate malware from benign instances with high accuracy of detection, which was the result of using the Grad-CAM method to gain insight into how the model has been driven to make a certain classification decision. The baseline for accuracy in this paper was set by training pre-trained models such as Xception and VGG19 on our datasets.

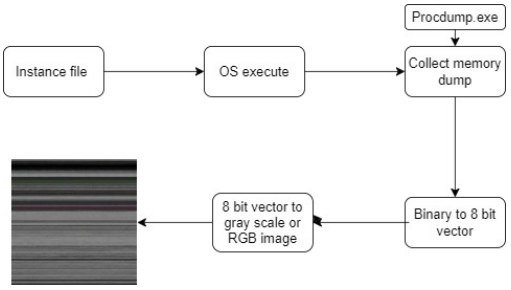

Figure 1: converting malware binary to image

The rest of this paper consists of the following sections: Related works, which discuss previous efforts in the field of malware detection using traditional ML methods and the newer DL approaches. The proposed model section includes an in-depth description of the proposed architecture as well as the reason behind each feature; The datasets and implementation reviews the datasets on which the research was carried out as well as the method of implementation. In the results section, the overall comparison of the proposed model with other models is provided as well as the ablation study for the proposed model. The summary and conclusions presents the future direction of this

research and the final assessment of the model.

2. Related Work

In this section, we discuss the efforts of other researchers in the field of malware detection using machine-learning and deep-learning-based methods, as well as their upsides and downsides. Kumar (10) proposed a method for static analysis of malware using deep transfer learning for image classification using well-defined models such as VGG16 and VGG19 which was done on malimg dataset. The research conducted by Bensaoud et al. (12), compared the performance of several well-known CNN models such as InceptionV3, VGG16, and ResNet50 to classify malware binaries converted to images. In this research, they stated that VGG16 and ResNet50 models showed low performance since they were both designed to classify RGB images. In the results of our research, VGG16 and ResNet50 showed good overall performance in malware classification and detection on the malimg dataset. What we did was take the single grey-scale input and copy it three times across the RGB channel. It is not stated how they obtained these results but one possible way could be that they replaced other input channels with zeros, hence the low performance. The model proposed by Puneeth et al. (11) was a huge inspiration for our research as it was a balanced combination of 2D convolutions and depth-wise convolutions which resulted in good performance and memory efficiency. With subtle modifications and use of uneven kernel sizes, we were able to reduce the number of trainable parameters of the model and achieve greater results on the same datasets for both dynamic and static analysis. Gibert et al.(9) proposed a CNN approach for classification of malware represented as images with an overall classification accuracy on malimg of 98.48% which was stated to be higher than that of Nataraj (13), 97.18%.

2.1. Machine learning and Deep learning methods:

The use of traditional machine learning methods to detect malware has been investigated in several papers(8) (7). The methods used include decision trees, SVM, and Naive Bayes with boost. Makandar et al, (6) used multi-class SVM for malware image classification and achieved an average accuracy of 96.35%. Nataraj et al.(13) used KNN to classify 9342 gray-scale images of malware files belonging to 25 families by which they obtained 98.08% overall accuracy. Kalash et al. (5) proposed a CNN-based architecture to classify malware binaries converted to grayscale images. They achieved overall accuracy of 98.52% and 99.97% on the Malimg and Microsoft datasets respectively. In the researchs mentioned above, there is a lack of using multiple datasets to both test the models on static and dynamic analysis of malware, also there is no previous endeavor to introduce explainability to the models trained on malware images. The approach of this researh was to use attetion methods such as Grad-CAM to better understand how black-box models make predictions.

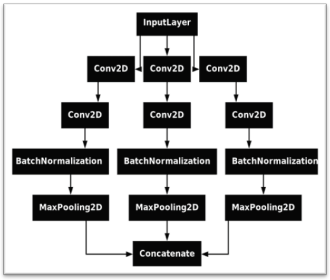

Figure 2: The first stage of the proposed model

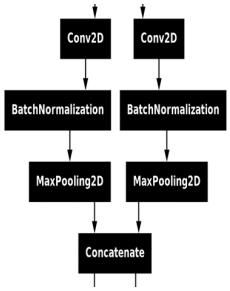

Figure 3: The second stage of the proposed model

3. The Proposed model:

In this paper, we propose a novel network architecture that excels at detection and classification of malware images using key characteristics of well-known image classification models along with subtle modifications to maximize the performance. This architecture comprises four stages of feature extraction from the malware image, all of which are unique from one another to obtain features on different levels. The use of varying kernel sizes and use of depth-wise convolutions is one of the innovations used in this architecture. Through extensive experimentation, the optimal four-stage architecture was developed to ensure it neither underfits nor overfits.

3.1. First stage

The first stage as shown in Figure 2 extracts features from the given data using two 2D convolutions of depth 64 in three paths each of which has different kernel sizes of (3 * 12) and (3 * 9) in the first path, (3 * 9) and (3 * 6) in the second path and lastly (3 * 6) and (3 * 3) in the third path to extract different patterns from the data followed by batch normalization and Max pooling to down-sample the features and then concatenated to be fed to the next stage.

3.2. Second stage

The second stage as shown in Figure 3 includes two paths of 2D convolution with depth of 128 to and kernel sizes of (3*6) and (3*3) in order to make the model capable of capturing more complex features and patterns from the data. Batch normalization is used after the convolution layers to help mitigate overfitting as we go deeper.

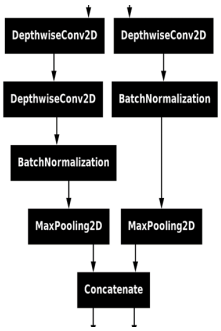

Figure 4: The third stage of the proposed model

3.3. Third stage

The third stage of the proposed model is made up of two paths of Depth-wise convolutions for parameter efficiency. One path takes the output of the previous stage and performs Depth-wise convolutions with kernel sizes of (3*3) two time followed by a Batch normalization layer and a max pooling layer to down sample the features. The other path uses Depth-wise convolution once followed by Batch normalization and max pooling layers. The output of the two paths is then concatenated to be fed to the next stage as shown in Figure 4. The whole process is repeated two times as using depth-wise convolutions makes it so efficient to use them multiple times and achieve performance comparable to standard convolutions. This stage maintains expressive performance while having relatively low computational cost.

3.4. Fourth stage

The fourth stage of the model as shown in Figure 5 is where we feed the output of the previous stages to a convolution layer of depth 256 and a kernel size of 3*3 to capture hierarchical representations and learn abstract features by combining lower-level features of the previous stages. After the convolution, there is a max pooling layer to down-sample the features and the data is flattened to be passed dense layers of 2048 neurons each to aggregate the local features extracted by the convolution layers across the entire spatial extend of the data and achieve global feature combination. We added dropout regularization after each dense layer with a rate of 0.4 to prevent overfitting. The final dense layer is the classification layer which was modified according to the number of classes of the data. This model takes inspiration from the proposed model from the paper (11) with modifications to improve the performance. Differences being the depth and kernel size in convolution layers along with added paths to the initial stage.

3.5. Reason for un-even kernels

The main motivation for using un-even rectangular kernel sizes of (3 * 12), (3 * 9), and (3 *6) is the results of the Grad-CAM (1) experiments we ran on the malware images to gain better insight into the data at hand. The Grad-CAM provides visual explanations for decisions made by the model which uses gradients from the final convolutional layer to highlight the important regions in an image. In simple words, it demonstrates

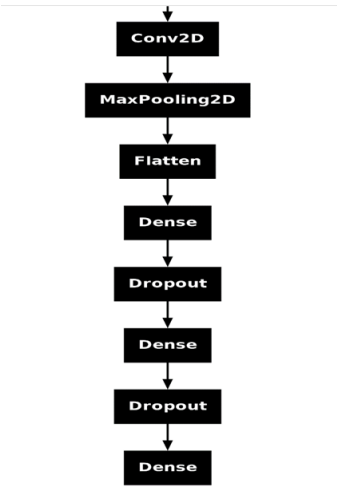

Figure 5: The fourth stage of the proposed model

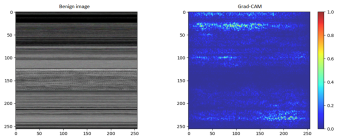

Figure 6: Grad-CAM for a benign image

what pixels in an image contribute most to the decision made by the model in the classification. In our experiment with Grad-CAM on the dataset we found out the most highlighted areas form rectangular shapes as shown in Figure 6
Figure 7 Figure 8 Figure 9, thus the use of un-even kernel sizes was put to test for better capturing patterns and features.

3.6. Conclusion

Overall, given the results from the Grad-CAM which was performed on the binary dataset samples, we proposed an DCNN architecture to capture features relevant to the data at hand, this model is computationally efficient and performs better than the existing malware detection frameworks.

4. Datasets and implementation

The data sets used in this research are the leading datasets in the field of malware detection, which are Malimg(3), Dumpware10(4), MaleVis(2), and a binary dataset.

4.1. Malimg dataset

The Malimg dataset consists of 9339 malware images from 25 malware families and was constructed by converting malware binaries into grayscale images, this makes it a dataset

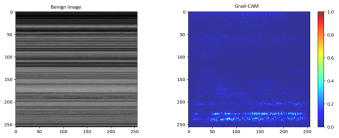

Figure 7: Grad-CAM for a benign image

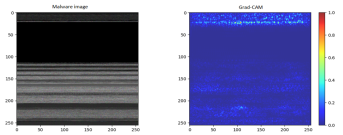

Figure 8: Grad-CAM for a malware image

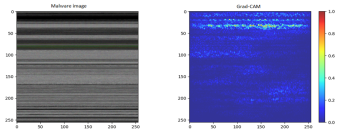

Figure 9: Grad-CAM for a malware image

for static malware analysis. Samples from malimg dataset are shown in Figure 10

4.2. Dumpware10 dataset

The dumpware10 dataset consists of 4294 RGB images, 3686 of them are malware images of 10 classes. This dataset was constructed by running malware files and collecting the memory dump, which is a good data set for dynamic analysis. Samples of Dumpware10 dataset are shown in Figure 11.

4.3. MaleVis dataset

This dataset was developed to carry benchmark on proposed models in the malware detection field and consists of 9100 images for the train set and 5126 images for the test set, that belong to 25 families of malwares and an additional class of benign images. The dataset was constructed by converting malware binaries to RGB images using bin2png script. The dataset has 350 and 1482 benign samples in train and test set respectively to evaluate the models on their ability to detect legitimate images from malware images. Samples are shown in Figure 12.

4.4. Binary dataset

This dataset consists of 19796 RGB images, 5209 benign images and 13587 malware images. The ProcDump utility was used to capture malware memory dumps and convert them into RGB images. This dataset was generated as a tool for dynamic analysis of malware files. A few samples of the binary dataset are shown in Figure 13.

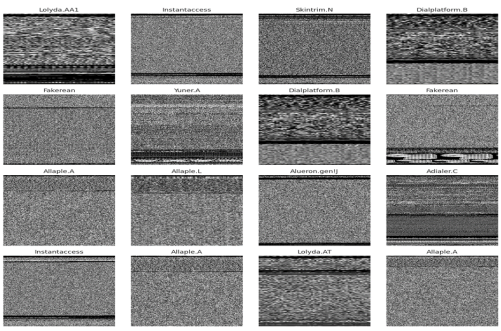

Figure 10: Malimg dataset samples

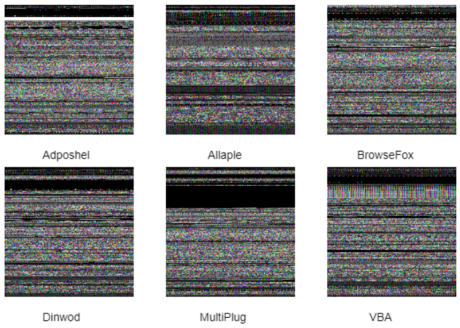

Figure 11: Dumpware10 dataset samples

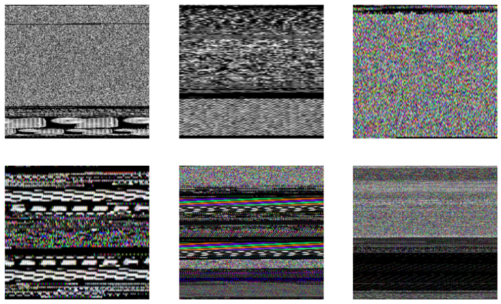

Figure 12: MaleVis dataset samples

5. Results

The model proposed in this paper was benchmarked on different datasets and was compared to other existing architectures. The proposed model in this paper achieved greater results in all benchmarks and demonstrated competence in both static and dynamic malware analysis. The static analysis results on the Malimg dataset and MaleVis dataset are shown in Table 2

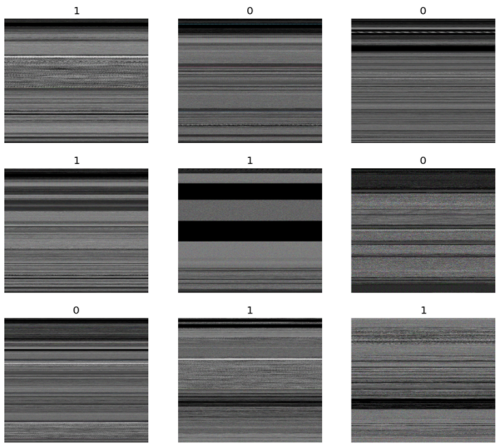

Figure 13: Bianry dataset samples

Table 1: Achieved performance of the proposed model on binary dataset

Model	Accuracy	Precision	Recall	F1
VGG16	0.9154	0.9045	0.9266	0.9154
VGG19	0.9259	0.9200	0.9311	0.9255
Xception	0.9327	0.9279	0.9376	0.9327
Proposed Model	**0.9478**	**0.9363**	**0.9592**	**0.9476**

Table 2: Achieved performance of the proposed model on Malimg dataset

Model	Accuracy	Precision	Recall	F1
VGG16	0.9676	0.9520	0.9676	0.9578
VGG19	0.9896	0.9903	0.9896	0.9892
ResNet50	0.9707	0.96127	0.97074	0.96533
MobileNet	0.98015	0.98233	0.98015	0.97970
Xception	0.97597	0.97069	0.97597	0.97215
InceptionV3	0.97074	0.96041	0.97074	0.96436
Proposed model	**0.99582**	**0.99638**	**0.99582**	**0.99578**

and Table 4 and the dynamic analysis results of Binary dataset and Dumpware10 dataset are shown in Table 1 and Table 3.

5.1. Ablation Study

The ablation study for this case was carried out by testing the proposed model with all the same specifications except for the change in the kernel sizes. The results of the ablation study as shown inTable 6 demonstrate significant improvement in the performance of the model on the test set.

To prove the superiority of the proposed model we tested different kernel sizes and benchmarked the model on the same datasets to give a fair verdict for the proposed architecture. As we can see in Figure 17 and Figure 18, the test accuracy for the model with all 3*3 kernels on datasets Malimg and MaleVis was 96.1337 and 86.3636, respectively. Also for Dumpware10, it achieved 98.1298 in test accuracy.

5.2. Efficiency:

As regards the efficiency of the proposed model, the model is computationally efficient compared to other reviewed and proposed models. As we can see from Table 7, the number of trainable parameters in our proposed model is almost half the parameter count in the RDMNet of the paper (11) with without sacrificing performance.

Table 3: Achieved performance of the proposed model on Dumpware10 dataset

Model	Accuracy	Precision	Recall	F1
VGG16	0.7844	0.9323	0.7844	0.8039
VGG19	0.9164	0.9416	0.91	0.9147
ResNet50	0.9945	0.9947	0.9946	0.9945
MobileNet	0.9987	0.9978	0.9978	0.9978
Xception	0.9923	0.9925	0.9923	0.9923
InceptionV3	0.9945	0.9946	0.9948	0.9945
Proposed model	**0.9989**	**0.9989**	**0.9990**	**0.9989**

Table 4: Achieved performance of the porposed on MaleVis dataset

Model	Accuracy	Precicion	Recall	F1
VGG16	0.8926	0.9123	0.8962	0.8999
VGG19	0.8622	0.9002	0.8622	0.8694
ResNet50	0.9323	0.9420	0.9323	0.9348
MobileNet	0.9004	0.9417	0.9004	0.9124
Xception	0.9109	0.9165	0.9109	0.9115
InceptionV3	0.8986	0.9127	0.8986	0.9022
Proposed model	**0.9400**	0.9432	0.9405	0.9396

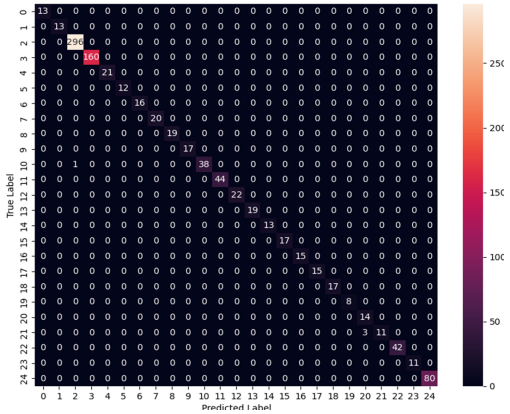

Figure 14: Confusion matrix of the porposed model on Malimg dataset

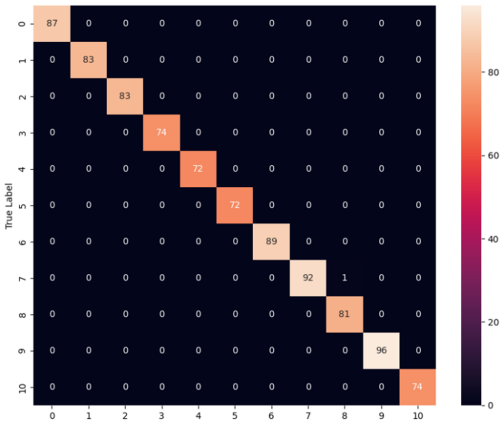

Figure 15: Confusion matrix of the proposed model on Dumpware10 dataset

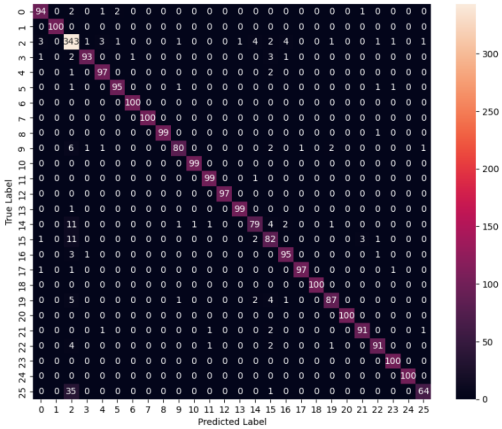

Figure 16: Confusion matrix of the proposed model on MaleVis dataset

	Predicted Positive	Predicted Negative
Actual Positive	1178	48
Actual Negative	83	1174

Table 5: Confusion Matrix of the propsoed model on Binary dataset

Table 6: Ablation study of different kernel sizes

Dataset	3*3 kernels	un-even kernels
Malimg	96.13	99.58
Dumpware10	98.12	99.89
MaleVis	86.36	94.00

5.3. Contradictions found:

In reviewing (12), we found contradictions with our findings regarding the performance of the VGG16 and ResNet50 models on the classification of malware images. It is stated that VGG16 and ResNet50 showed low performance compared to the other models since both of these models were designed to recognize colored images that require RGB format. Therefore, both give low accuracies when tested on grayscale images. It is not stated how the testing was done but one probable way is to just use one channel of malimg and fill the other two channels with sparse zeros which gives poor results. What we did was copy the vectorized data of the image across the three channels so that we have three channels for VGG16 and ResNet50 which gives realistic results as shown in Table 8.

6. Summary and conclusions

In this paper a novel CNN architecture was proposed based on insights from Grad-CAM results of malware images. In addition, other well-known pre-trained models were also benchmarked on the datasets to compare the accuracy of the proposed model with existing models. There were also contradictions found with the results of other papers which were addressed.

References

[1] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh and D. Batra, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization," 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 2017, pp. 618-626, doi: 10.1109/ICCV.2017.74.

[2] Bozkir, Ahmet, Cankaya, Ahmet, Aydos, Murat. (2019). Utilization and Comparision of Convolutional Neural Networks in Malware Recognition. 10.1109/SIU.2019.8806511.

[3] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, S. Venka traman, Robust intelligent malware detection using deep learning, IEEE Access 7 (2019) 46717–46738. doi:10.1109/access.2019.2906934.

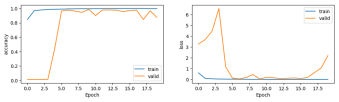

Figure 17: Malimg benchmark with 3*3 kernels

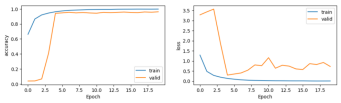

Figure 18: MaleVis benchmark with 3*3 kernels

Table 7: Parameter comparison on Malimg dataset with paper (11)

Architecture	Proposed Model	RDMNet
Total parameters	33,006,041	70,805,273

[4] Bozkir, A. S., Tahillioglu, E., Aydos, M., & Kara, I. (2021). Catch them alive: A malware detection approach through memory forensics, manifold learning and computer vision. *Computers & Security*, *103*, 102166. https://doi.org/10.1016/j.cose.2020.102166

[5] M. Kalash, M. Rochan, N. Mohammed, N. D. B. Bruce, Y. Wang and F. Iqbal, "Malware Classification with Deep Convolutional Neural Networks," *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, Paris, France, 2018, pp. 1-5, doi: 10.1109/NTMS.2018.8328749.

[6] A. Makandar and A. Patrot, "Malware analysis and classification using Artificial Neural Network," *2015 International Conference on Trends in Automation, Communications and Computing Technology (I-TACT-15)*, Bangalore, India, 2015, pp. 1-6, doi: 10.1109/ITACT.2015.7492653.

[7] Dube, Thomas, Richard Raines, Bert Peterson, Kenneth Bauer, and Steven Rogers. "An investigation of malware type classification." In *International Conference on Cyber Warfare and Security*, p. 398. Academic Conferences International Limited, 2010.

[8] Kolter, J.Z. and Maloof, M.A. (2006) Learning to Detect and Classify Malicious Executables in the Wild. Journal of Machine Learning Research, 7, 2721-2744.

[9] Gibert, Daniel, Mateu, Carles, Planes, Jordi, Béjar, Javier, Vicens, Ramon, Solis, Daniel. (2017). Convolutional Neural Networks for Classification of Malware Assembly Code. 10.3233/978-1-61499-806-8-221.

[10] Sanjeev Kumar, B. Janet, DTMIC: Deep transfer learning for malware image classification, Journal of Information Security and Applications, Volume 64, 2022, 103063, ISSN 2214-2126, https://doi.org/10.1016/j.jisa.2021.103063

[11] S. Puneeth, S. Lal, M. Pratap Singh and B. S. Raghavendra, "RMDNet-Deep Learning Paradigms for Effective Malware Detection and Classification," in IEEE Access, vol. 12, pp. 82622-82635, 2024, https://doi.org/10.1109/ACCESS.2024.3403458

[12] Bensaoud, A., Abudawaood, N., Kalita, J. (2020). Classifying malware images with convolutional neural network models. International Journal of Network Security, 22(6), 1022-1031. https://doi.org/10.6633/IJNS.202011_22(6).17.

[13] Lakshmanan Nataraj, Sreejith Karthikeyan, Gregoire Jacob, and BS Manjunath. Malware Images: Visualization And Automatic Classification. In Proceedings of the 8th International Symposium on Visualization for Cyber Security, page 4, 2011.

[14] Ali Mirza, Qublai Khan, Irfan Awan, Muhammad Younas. (2017). CloudIntell: An intelligent malware detection system. Future Generation Computer Systems. 86. https://doi.org/10.1016/j.future.2017.07.016.

[15] Chollet, F. (2017). Deep learning with python. Manning Publications.

[16] Ahmet Selman Bozkir, Ersan Tahillioglu, Murat Aydos, Ilker Kara, Catch them alive: A malware detection approach through memory forensics, manifold learning and computer vision, Computers & Security, Volume 103, 2021, 102166, ISSN 0167-4048, https://doi.org/10.1016/j.cose.2020.102166.

[17] Bozkir, Ahmet, Cankaya, Ahmet, Aydos, Murat. (2019). Utilization and

Table 8: Contradiction in results of (12) on malimg dataset

Model	Our try	Reported Results
VGG16	96.76	14.31
ResNet50	97.07	26.66

Comparision of Convolutional Neural Networks in Malware Recognition. 10.1109/SIU.2019.8806511.