Malware Detection via Memory Dump Images: Investigating the Role of Uneven Kernel Filters in CNNs with Visual Explainability

Mohammadhadi Alaeiyan[a], Pooria Lakzian[b]

[a] *Faculty of Computer Engineering, K. N. Toosi University of Technology, Seyed Khandan, Shariati Ave, Tehran, 16317-14191, Tehran, Iran*
[b] *School of Mathematics and Computer Science, Iran University of Science and Technology, Narmak, Tehran, 16844, Tehran, Iran*

Abstract

With the widespread availability of the Internet, safeguarding our digital identities has become increasingly critical. Our devices are now constantly connected to the Internet, transmitting data and making them vulnerable to various threats. Traditional malware detection methods are no longer sufficient to ensure digital safety due to the rapid increase in the complexity and sophistication of malware. Researchers have recently turned to artificial intelligence, particularly machine learning, and deep learning techniques, to combat these threats with promising results. One emerging approach involves converting malware binaries or memory dumps into images. However, these malware images are not recognizable by the human eye and, therefore, remain unexplained as to how the CNNs perceive this data type. This paper applies the Grad-CAM technique to malware images collected from memory dumps to understand better how black-box CNN models make predictions with this data. Based on insights from the Grad-CAM analysis, we propose a novel CNN architecture with uneven kernel sizes that outperforms existing malware detection and classification models. We compare its performance against well-known CNN architectures, all trained and tested on malware image datasets such as Malimg, Dumpware10, MaleVis, and MalBen. Our proposed architecture achieved an impressive accuracy of 99.58% on the Malimg dataset with fewer training parameters, surpassing previous models, which achieved a maximum accuracy of 99.26% on the same dataset.

Keywords: Deep learning, Classification, Malware, Dynamic analysis, Memory dumps.

1. Introduction

In recent years, the rapid evolution of malware has presented a significant challenge to cybersecurity, demanding more sophisticated detection and classification techniques. Traditional methods often depend on signature-based detection and have proven insufficient against increasingly complex and fast-adapting malware variants [1, 2, 3, 4]. As a result, there has been a growing interest in employing deep learning techniques to improve the accuracy and efficiency of malware detection and classification [5, 6, 7].

In general, CNN [8], LSTM [], transfer learning [], and classical machine learning methods [] are applied to distinguish between malware and benign instances. As proposed by Nataraj et al., [8], a malware visualization approach converts malware binaries into images, applying image classification techniques to identify and categorize malware. Also, Bensaoud et al. [9] applied some Convolutional Neural Networks (CNNs) methods on images created from binary files. However, their proposed method is dataset-dependent. Moreover, prior works ignored the role of kernel size while applying CNNs and lacked attention to offering an applicable method for both gray-scale and RBG images.

Our proposed method utilizes CNNs to extract and learn hierarchical features from image data, allowing for the accurate classification of malware samples. However, it is considered a black-box approach because it is unclear which regions of the malware image the model identified as crucial for prediction and lacks interpretability. In our proposed model, we fine-tuned our CNN using insights from Grad-CAM to identify essential regions, resulting in significantly improved results. This type of visual explanation has also been applied to medical images.

This paper includes both static and dynamic analysis of malware. The dynamic analysis shown in 1 is the conversion of collected memory dumps of Portable Executable (PE) files by the Procdump program [10] into RGB or gray-scale images [9]. In the static analysis, the PE file is converted to 8-bit vectors, which then take a value between 0-255, representing one pixel of the image as grey-scale or 24-bit vectors as RGB images [9, 11].

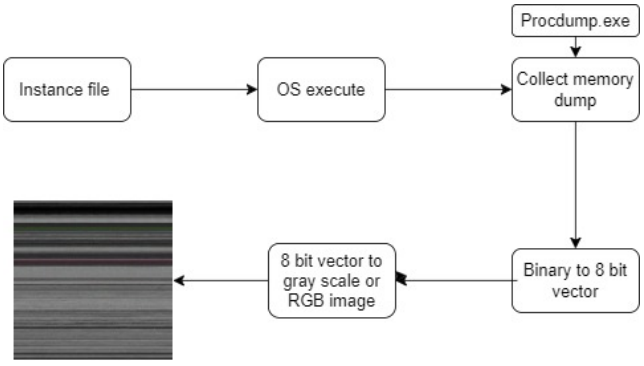


Figure 1: Extraction of memory dump from a PE file and conversion into image.

This paper proposes a novel CNN architecture designed explicitly for malware image classification and tests it on static and dynamic analysis approaches. Our architecture aims to improve classification accuracy by incorporating techniques such as residual connections to improve convergence and address the problem of vanishing gradients, Depth-wise convolutions for memory efficiency while retaining performance, and kernel optimization using insights from Grad-CAM visualization of malware images. Through extensive experiments and evaluations, we show that our proposed CNN can segregate malware from benign instances with high detection accuracy, which resulted from using the Grad-CAM method to gain insight into how the model has been driven to make a confident classification decision. The baseline accuracy in this paper was established using the performance of pre-trained models like Xception (91.09% on MaleVis) and VGG19 (98.96% on Malimg). We assessed the accuracy of these models on each dataset and then compared their performance to the accuracy achieved by our approach.

Contribution: The key contributions of our proposed memory dump malware detector are as follows:

- Improving classification accuracy by incorporating techniques such as residual connections to improve convergence and address the problem of vanishing gradients and depth-wise convolutions for memory efficiency while retaining performance.
- Higher detection accuracy while having a fewer number of parameters.
- Using the Grad-CAM visualization method on malware images for more explainability and finding significant kernel filter sizes for better feature extraction.
- Publishing dynamically analyzed malicious and legitimate memory dump images as a dataset.

Remaining: The remainder of this paper is organized as follows: Section 2 reviews previous research in malware detection, focusing on traditional machine learning methods and recent deep learning approaches. Section 3 provides a detailed description of the architecture and the rationale behind each design choice. Section 4 discusses the datasets used in this study and outlines the implementation process. In Section 5, we comprehensively compare the proposed model with other state-of-the-art models, including an ablation study of our architecture. Finally, the Summary and Conclusions section outlines the potential future directions of this research and provides a final evaluation of the model in Section 6.

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 using binary [9, 11] or memory dump images [1], as well as their upsides and downsides.

Kumar and Janet [11] introduced a method for static malware analysis by leveraging deep transfer learning for image classification, utilizing established models such as VGG16 and VGG19. Their approach was applied to the Malimg dataset, which comprises grey-scale images representing 25 malware families [12]. Similarly, research by Bensaoud et al. [9] evaluated the performance of six popular CNN models, including InceptionV3, VGG16, and ResNet50, for static malware classification on the same Malimg dataset. They observed that the VGG16 and ResNet50 models performed poorly on grey-scale images from Malimg, likely due to their original design for RGB image classification. Contrary to these findings, our study demonstrated that VGG16 and ResNet50 achieved solid malware classification and detection performance when applied to the Malimg dataset.

The model proposed by Puneeth et al. [1] was a huge inspiration for our research. It was a balanced combination of 2D convolutions and depth-wise convolutions, resulting in good performance and memory efficiency. With subtle modifications and the use of uneven kernel sizes, we reduced the number of trainable parameters of the model. We achieved more significant results on the same datasets for both dynamic and static analysis. Gibert et al.[13] proposed a CNN approach for classifying malware images with a higher overall classification accuracy on Malimg than that of Nataraj [8].

The use of traditional machine learning methods to detect malware has been investigated in several papers [14, 15]. The techniques used in these papers include decision trees, SVM, and Naive Bayes with boost. Makandar et al., [16] used multiclass SVM for malware image classification and achieved an average accuracy of 96.35%. Nataraj et al. [8] 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. [17] proposed a CNN-based architecture for detecting and classifying malware images, which achieved an overall accuracy of 98.52% and 99.97% on the Malimg [12] and Microsoft datasets [18], respectively.

In the research mentioned above, multiple datasets are not used to 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. Our research used methods such as Grad-CAM to understand better how black-box models make predictions and fine-tune our model based on these understandings.

3. The Proposed model

This paper proposes a novel network architecture that excels at detecting and classifying malware using critical characteristics of binary or memory dump image classification models and subtle modifications to maximize performance. This architecture comprises four feature extraction stages from the collected malicious and legitimate images, all unique to obtain different-level features. Using varying kernel sizes and depth-wise convolutions is one of the innovations used in this architecture. The optimal four-stage architecture was developed through extensive experimentation by adding and removing different layers to ensure it neither underfits nor overfits.

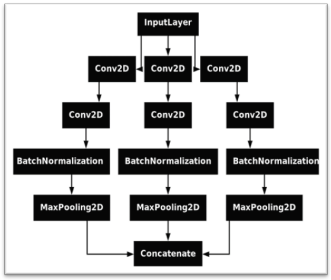


Figure 2: The first stage of our proposed model.

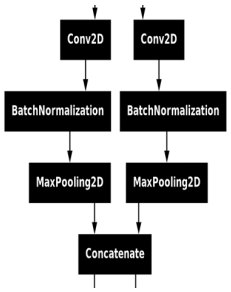


Figure 3: The second stage of the proposed model.

This model draws inspiration from the proposed architecture in [1], incorporating several modifications to improve performance with fewer parameters. Key differences include using different kernels in each convolutional layer, using more convolutional layers with fewer filters, using Dropout layers to reduce overfitting, and adding multiple paths at the initial stage.

3.1. First stage

The first stage, as shown in Figure 2, has an adjustable input layer depending on the input type, whether it is grey-scale or RGB, and its resolution; it takes the input and feeds it to the network to extract 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 a Batch Normalization layer to normalize the data and speed up the training process, 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, takes the concatenated data from the previous stage and passes it through two paths of 2D convolution with a depth of 128 and kernel sizes of 3×6 and 3×3 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 by normalizing the data after each computational layer and producing smoother gradients as we go deeper.

3.3. Third stage

The third stage comprises two paths of depth-wise convolutions for parameter efficiency. One path takes the previous stage's output. It performs depth-wise convolutions with kernel sizes of 3×3 twice, followed by a batch normalization layer and a max pooling layer to downsample the features. The other path uses depth-wise convolution, 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.

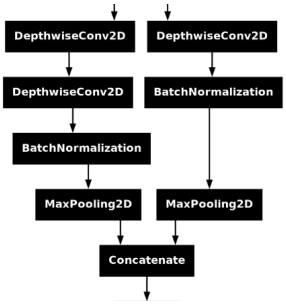


Figure 4: The third stage of the proposed model.

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 earlier stages. After the convolution, a max pooling layer is used to down-sample the features. The data is flattened to be passed through dense layers of 2048 neurons each to aggregate the local features extracted by the convolution layers across the entire spatial extent of the data and achieve a 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 based on the number of classes in the dataset. The activation function for binary classification was set to Sigmoid; for multi-class classification, it was Softmax.

3.5. Reason for uneven kernels

The primary motivation for using uneven rectangular kernel sizes of 3×12, 3×9, and 3×6 is the results of the Grad-CAM [19] 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 critical regions in an image. Simply, it demonstrates 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 MalBen dataset, we found out the most highlighted areas form rectangular shapes as shown in Figure 6; One possible explanation for this would be the sequential nature of the binary data before forming the pixels of the image. Thus, the use of uneven

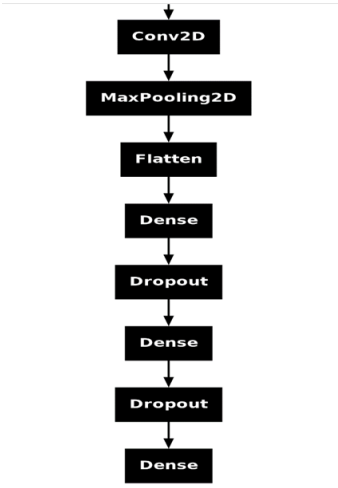


Figure 5: The fourth stage of the proposed model

kernel sizes was put to the test for better capturing patterns and features.

3.6. Findings

Given the Grad-CAM results performed on the MalBen dataset samples, we proposed a deep CNN architecture to capture features relevant to the data at hand. This model is computationally efficient and performs better than the existing malware detection frameworks. We assume that due to the sequential nature of the data, it is better to use rectangular filters in Convolution operations than square filters.

4. Datasets and Descriptions

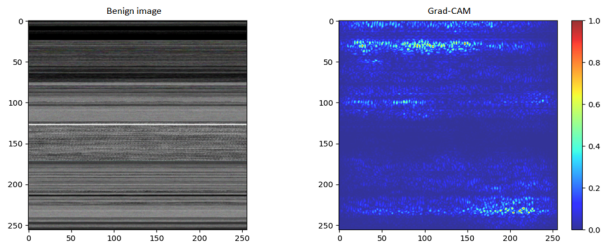
The datasets used in this research are the leading datasets in the field of malware detection, which are Malimg[12], Dumpware10[20], and MaleVis[21]. Also, we collected a large-scale dataset, named MalBen [22], included from Virusshare [23]. Then, we extracted their memory dumps and relevant images.

4.1. Malimg dataset

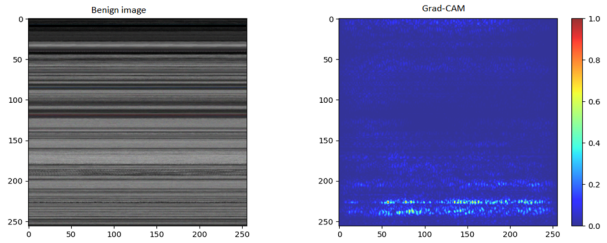
The Malimg dataset [12] consists of 9339 malware images from 25 malware families and was constructed by converting malware binaries into grey-scale images, making it a static malware analysis dataset. Samples from the Malimg dataset are shown in Figure 7. So far, the highest accuracy reported on the Malimg dataset belongs to the [24]. This was achieved using a method called SE-AGM.

4.2. Dumpware10 dataset

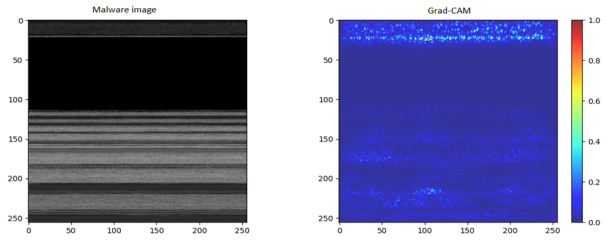
The Dumpware10 dataset [20] consists of 4294 RGB images, 3686 of which are malware images of 10 classes. This dataset was constructed by running malware files and collecting the memory dump, making it a good dataset for dynamic analysis. Samples of the Dumpware10 dataset are shown in Figure 8. The highest accuracy reported on the Dumpware10 dataset so far was by [25], which was 99.60%.



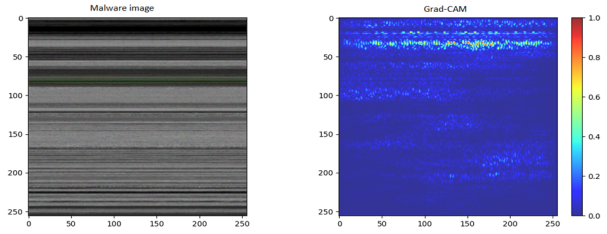
(a) Grad-CAM for a benign image



(b) Grad-CAM for a benign image



(c) Grad-CAM for a malware image



(d) Grad-CAM for a malware image

Figure 6: Grad-CAM visualizations of MalBen samples using 3×3 kernels.

4.3. MaleVis dataset

The MaleVis dataset [21] 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 malware and an additional class of benign images. The dataset was constructed by converting malware binaries to RGB images using the bin2png script [26]. The dataset has 350 and 1482 benign samples in train and test sets to evaluate the models' ability to detect legitimate images from malware images. Samples are shown in Figure 9. Regarding the problem as a close set form (i.e., excluding the legitimate samples), in [21], Densenet-based convolutional neural networks have achieved an accuracy of 97.48% on the MaleVis validation set. However, our proposed model was trained and tested on an open set with legitimate samples.

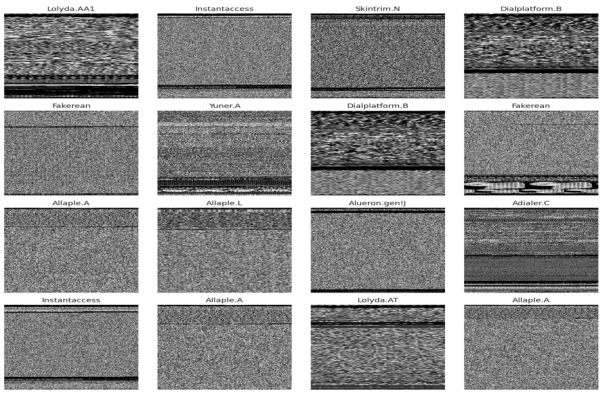


Figure 7: Malimg dataset samples [12].

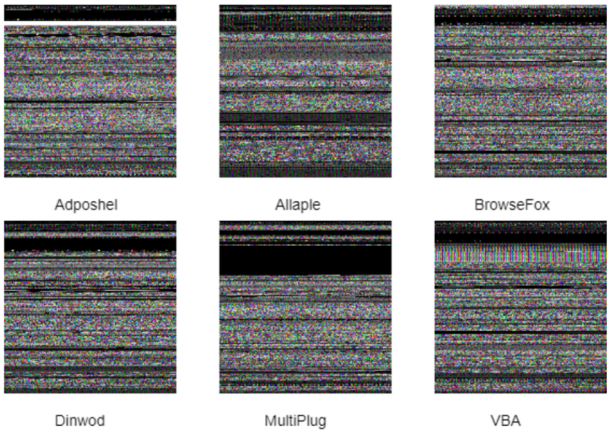


Figure 8: Dumpware10 dataset samples [20].

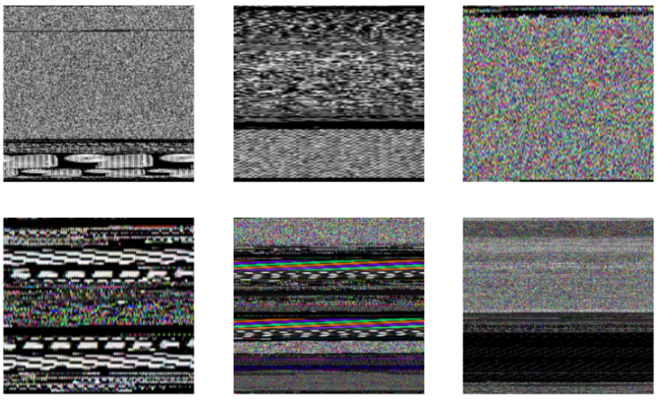


Figure 9: MaleVis dataset samples [21].

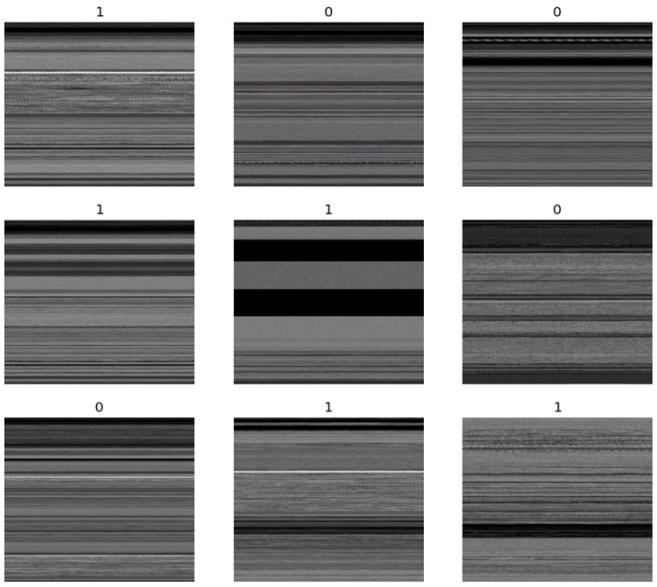


Figure 10: MalBen dataset samples.

4.4. MalBen dataset

This dataset comprises 19796 RGB images, including 5209 benign and 13587 malware images [22]. The ProcDump utility captured and converted malware memory dumps into RGB images. We generated this dataset as a tool for the dynamic analysis of malware files, and it is publicly available for future research use. A few samples of the MalBen dataset are shown in Figure 10.

5. Results

The model proposed in this paper was benchmarked on different datasets, as mentioned in Section 4, and compared to other existing architectures. For each dataset, 80% of the data was used for training, 10% for validation, and 10% for the test. The training process was carried out with RMSprop as the optimizer with a learning rate of $\alpha = 0.0001$ and early stopping with the patience of 5 epochs monitoring validation accuracy. The proposed model in this paper achieved more significant results in all benchmarks and demonstrated competence in static and dynamic malware analysis. The results of the static analysis on the Malimg dataset and MaleVis dataset are shown in Table 1 and Table 2. The dynamic analysis results of our collected memory dump MalBen and Dumpware10 datasets are shown in Table 3 and Table 4.

The confusion matrices of datasets Malimg, Dumpware10, MaleVis, and MalBen are shown in 11, 12, 13, and 14, respectively.

We evaluated our model's performance using key metrics across multiple classification and detection scenarios, and the experimental results demonstrate robust performance. In the case of the MalBen dataset, the model shows slight asymmetry between precision and recall, which indicates better performance in reducing false negatives and suggests that the model prioritizes capturing positive cases.

Table 1: The performance of the proposed model on the 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
RMDNet [1]	0.9926	0.9837	0.9812	0.9825
Proposed model	**0.99582**	**0.99638**	**0.99582**	**0.99578**

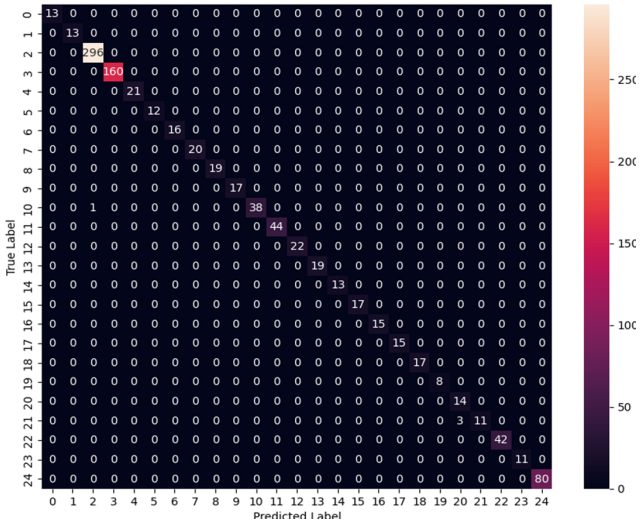


Figure 11: Confusion matrix of the proposed model on the Malimg dataset.

Table 2: The performance of the proposed model on the 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**

Table 3: The performance of the proposed model on our collected memory dumps dataset MalBen.

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**

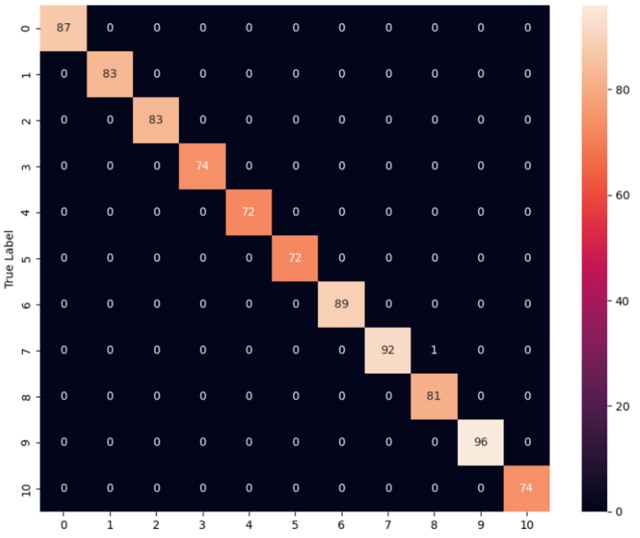


Figure 12: Confusion matrix of the proposed model on the Dumpware10 dataset.

Table 4: The performance of the proposed model on the 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**

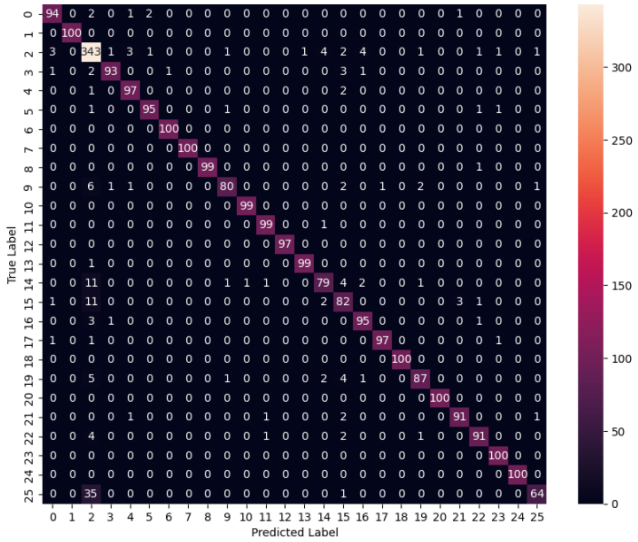


Figure 13: Confusion matrix of the proposed model on the MaleVis dataset.

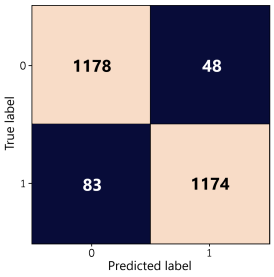


Figure 14: Confusion matrix of the proposed model on the MalBen dataset

Table 5: Ablation study of different kernel sizes.

Dataset	3×3 kernels	uneven kernels
Malimg	96.13	99.58
Dumpware10	98.12	99.89
MaleVis	86.36	94.00
MalBen	91.21	94.78

Table 6: Parameter comparison on Malimg dataset with [1].

Architecture	Proposed Model	RDMNet	VGG19
Total parameters	33,006,041	70,805,273	134,268,738

5.1. Effect of Kernel Size on Model Generalization

The ablation study for this case was carried out by testing the proposed model with all exact specifications except for the change in kernel sizes. 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 on the proposed architecture. The results of the ablation study, as shown inTable 5, demonstrate a significant improvement in the model's performance on the test set. The learning curve graphs of the model on these datasets using uneven and 3×3 kernels show a smoother curve when using the uneven kernels. It is also evident that the problem of overfitting after a certain number of epochs is solved using uneven kernels.

Using uneven kernel sizes led to achieving a smoother learning curve and better generalization on the data than the 3×3 kernels.

5.2. Efficiency:

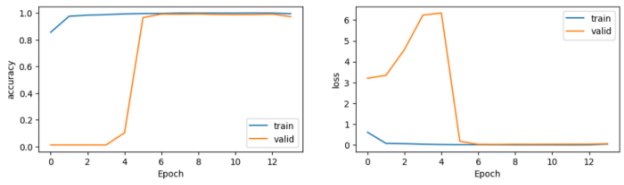
The proposed model is computationally efficient compared to other reviewed and proposed models. As we can see from Table 6, the number of trainable parameters in our proposed model is almost half the parameter count in the RDMNet of the paper [1] without sacrificing performance.

5.3. Contradictions found

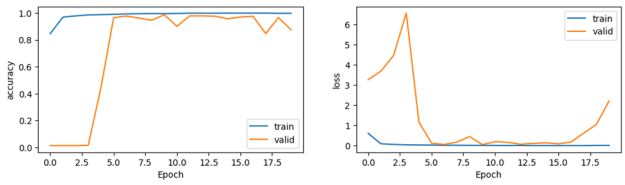
In reviewing [9], we found contradictions with our findings regarding the performance of the VGG16 and ResNet50 models on the classification of malware images. It is stated that VGG16 and ResNet50 showed low performance compared to the other models since both of these models were designed to recognize colored images that require RGB format. Therefore, both give low accuracies when tested on grey-scale images. It is not stated how the testing was done in their research, but one probable way is to use one channel of malimg and fill the other two channels with sparse zeros, which gives poor results. We copied the vectorized data of the image across the three channels so that we have three channels for VGG16 and ResNet50, which offers realistic results as shown in Table 7.

6. Summary and conclusions

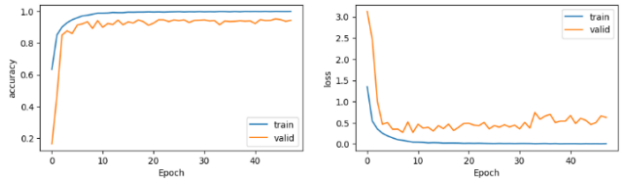
In this paper, we proposed a CNN architecture for robust malware detection based on the insights from Grad-CAM vi-



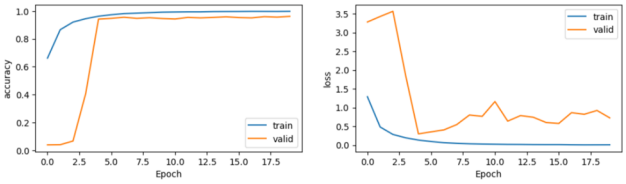
(a) Uneven kernels on Malimg dataset



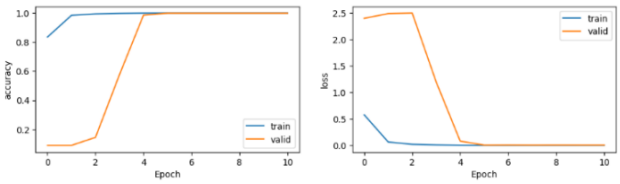
(b) 3×3 kernels on Malimg dataset



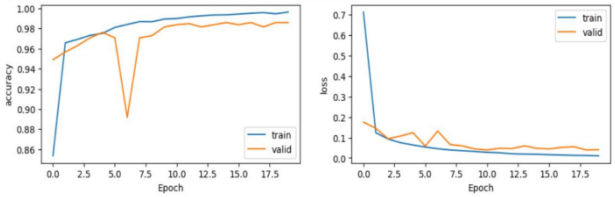
(c) Uneven kernels on MaleVis dataset



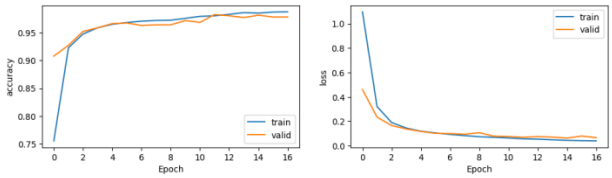
(d) 3×3 kernels on MaleVis dataset



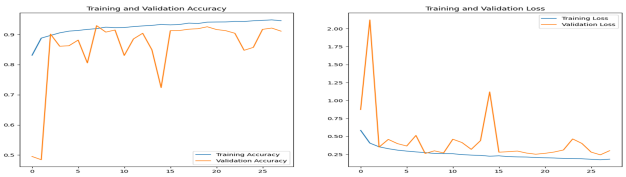
(e) Uneven kernels on Dumpware10 dataset



(f) 3×3 kernels on Dumpware10 dataset



(g) Uneven kernels on MalBen dataset



(h) 3×3 kernels on MalBen dataset

Figure 15: Comparison of uneven and 3×3 kernels on different datasets.

Table 7: Contradiction in results of [9] on malimg dataset

Model	Our try	Reported Results [9]
VGG16	96.76	14.31
ResNet50	97.07	26.66

sualization of malware images. We fine-tuned our model using uneven kernels of sizes 3×12, 3×9, 3×6, and 3×3 to capture hierarchical features at all levels, leading to significant detection improvement. The proposed CNN was compared with other well-defined CNN architectures using transfer learning methods on malware image datasets and showed superior performance. We also introduced a new MalBen malware image dataset gathered from memory dumps. We converted them to RGB images and made them publicly available for future studies on image-based malware detection.

References

[1] S. Puneeth, S. Lal, M. Pratap Singh, and B. S. Raghavendra. RMDNet-Deep Learning Paradigms for Effective Malware Detection and Classification. *IEEE Access*, 12:82622–82635, 2024.

[2] Mohammadhadi Alaeiyan, Saeed Parsa, and Mauro Conti. Analysis and classification of context-based malware behavior. *Computer Communications*, 136:76–90, 2019.

[3] Mohammadhadi Alaeiyan, Ali Dehghantanha, Tooska Dargahi, Mauro Conti, and Saeed Parsa. A multilabel fuzzy relevance clustering system for malware attack attribution in the edge layer of cyber-physical networks. *ACM Transactions on Cyber-Physical Systems*, 4(3):1–22, 2020.

[4] Mohammadhadi Alaeiyan, Saeed Parsa, and P Vinod. Sober: Explores for invasive behaviour of malware. *Journal of Information Security and Applications*, 74:103451, 2023.

[5] Meysam Ghahramani, Rahim Taheri, Mohammad Shojafar, Reza Javidan, and Shaohua Wan. Deep image: A precious image based deep learning method for online malware detection in iot environment. *Internet of Things*, 27:101300, 2024.

[6] Hassan Naderi, P Vinod, Mauro Conti, Saeed Parsa, and Mohammad Hadi Alaeiyan. Malware signature generation using locality sensitive hashing. In *Security and Privacy: Second ISEA International Conference, ISEA-ISAP 2018, Jaipur, India, January, 9–11, 2019, Revised Selected Papers 2*, pages 115–124. Springer, 2019.

[7] Francesco Mercaldo, Fabio Martinelli, and Antonella Santone. Deep convolutional generative adversarial networks in image-based android malware detection. *Computers*, 13(6), 2024.

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

[9] A. Bensaoud, N. Abudawaood, and J. Kalita. Classifying malware images with convolutional neural network models. *International Journal of Network Security*, 22(6):1022–1031, 2020.

[10] Microsoft. Procdump. `https://learn.microsoft.com/en-us/sysinternals/downloads/procdump`, 2024.

[11] Sanjeev Kumar and B. Janet. DTMIC: Deep transfer learning for malware image classification. *Journal of Information Security and Applications*, 64:103063, 2022.

[12] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, and S. Venkatraman. Robust intelligent malware detection using deep learning. *IEEE Access*, 7:46717–46738, 2019.

[13] Daniel Gibert, Carles Mateu, Jordi Planes, Javier Béjar, Ramon Vicens, and Daniel Solis. Convolutional Neural Networks for Classification of Malware Assembly Code. In *International Conference on Artificial Intelligence Research and Development*, 2017.

[14] J. Z. Kolter and M. A. Maloof. Learning to Detect and Classify Malicious Executables in the Wild. *Journal of Machine Learning Research*, 7:2721–2744, 2006.

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

[16] A. Makandar and A. Patrot. Malware analysis and classification using Artificial Neural Network. In *International Conference on Trends in Automation, Communications and Computing Technology*, pages 1–6, 2015.

[17] M. Kalash, M. Rochan, N. Mohammed, N. D. B. Bruce, Y. Wang, and F. Iqbal. Malware Classification with Deep Convolutional Neural Networks. *IFIP International Conference on New Technologies, Mobility and Security*, pages 1–5, 2018.

[18] R Ronen. Microsoft malware classification challenge. *arXiv preprint arXiv:1802.10135*, 2018.

[19] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. *IEEE International Conference on Computer Vision (ICCV)*, pages 618–626, 2017.

[20] A. S. Bozkir, E. Tahillioglu, M. Aydos, and I. Kara. Catch them alive: A malware detection approach through memory forensics, manifold learning and computer vision. *Computers & Security*, 103:102166, 2021.

[21] Ahmet Bozkir, Ahmet Cankaya, and Murat Aydos. Utilization and Comparison of Convolutional Neural Networks in Malware Recognition. *Signal Processing and Communications Applications Conference*, 2019.

[22] Mohammadhadi Alaeiyan. Malware dataset. `http://alaeiyan.ir/MalwareDataset.html`, 2024. Accessed: 2024-10-20.

[23] VirusShare. Virusshare.com. `https://www.virusshare.com`, 2024. Accessed: 2024-10-20.

[24] Pratyush Panda, Om Kumar C U, Suguna Marappan, Suresh Ma, Manimurugan S, and Deeksha Veesani Nandi. Transfer learning for image-based malware detection for iot. *Sensors*, 23(6), 2023.

[25] Adem Tekerek and Muhammed Mutlu Yapici. A novel malware classification and augmentation model based on convolutional neural network. *Comput. Secur.*, 112(C), January 2022.

[26] E. Sultanik. bin2png. `https://github.com/ESultanik/bin2png`, 2024.