International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 11 Issue: 11 | Nov 2024 www.irjet.net p-ISSN: 2395-0072

Malware Classification Using Deep Learning

Ankit Das[1], Sujal Jadhav[2], Mohd Saad Khan[3], Dhruv Patel[4], Keshav Halder[5]

[1] *Student, Dept. of Computer Science Engineering (Cyber Security), Thakur College of Engineering and Technology, Mumbai, Maharashtra*
[2]*Student, Dept. of Computer Science Engineering (Cyber Security), Thakur College of Engineering and Technology, Mumbai, Maharashtra*
[3]*Student, Dept. of Computer Science Engineering (Cyber Security), Thakur College of Engineering and Technology, Mumbai, Maharashtra*
[4]*Student, Dept. of Computer Science Engineering (Cyber Security), Thakur College of Engineering and Technology, Mumbai, Maharashtra*
[5]*Student, Dept. of Computer Science Engineering (Cyber Security), Thakur College of Engineering and Technology, Mumbai, Maharashtra*

Abstract - *Malware attacks are increasing and causing lots of harm to both organizations and normal users. In this study we have categorized malware according to the family to which it belongs. We evaluated the efficacy of six deep learning models—CNN, VGG16, VGG19, RestNet50, Xception and MobileNet—in classifying malware by examining Malimg dataset and examining classes of Malware. MobileNet outperformed other models according to our results, with the maximum accuracy of 98.74%. This illustrates how sophisticated deep learning techniques may be used to improve the classification of malware and emphasizes the necessity of ongoing model adaptation and improvement in order to combat changing cyberthreats.*

Key Words: Malware, Cybersecurity, AI, Deep Learning, CNN, VGG16, VGG19, MobileNet, Xception, ResNet50, Accuracy, Precision.

1.INTRODUCTION

Malware has become less of a novelty and more of a reality for billions of people and organizations around the world in recent years [4]. Malware attacks rose by 11 % to 13,44,566 in 2024 from 12,13,528 in 2023 [5]. Any software that damages, disrupts, gains access to, or steals user data and/or money is referred to as malware. Malware can result in a variety of problems, such as losing control over data, identity theft, private life eavesdropping, hardware failure, service denial, and more. Malware attacks target both normal users and commercial organizations. Whether the malware is infecting a personal computer or a business network, the damage it causes varies depending on the malware type and its intention. Among the most well-known forms of malware include trojans, viruses, worms, ransomware, spyware, and adware [4]. Tools such as Virus Total and PEview are used for the detection and classification of malware. Malware creates a threat to the integrity and authenticity of data.

Researchers have found the potential of deep learning, a powerful branch of machine learning, as a feasible approach to enhance malware detection and classification. [2] [1] [3]. The research done in recent times has increased its focus on using deep learning models to classify malware into specific families [1] [2] [3]. We will be using converting malware binaries into visual representations, image-based malware classification is a novel technique that enables efficient analysis and classification using deep learning models, especially Convolutional Neural Networks (CNNs). Deep learning models can remain relevant in the face of quickly changing threats because of their capacity to continuously learn from fresh data. Deep learning models are appropriate for real-time detection applications because, once trained, they can categorize new samples rapidly [7] [8]. The shortcomings of conventional malware detection methods, which frequently rely on static or dynamic analysis, are addressed by this approach. By checking for the unique qualities and patterns in malware samples, these models aim to precisely determine which particular malware belongs to which family. This research project is going to compare multiple deep learning models to solve the challenge of malware family categorization. This project can help develop new defenses against emerging malware and enable rapid responses to evolving malware attacks.

This project will involve:

Collecting and preprocessing the malware image data: We would be using a Dataset which contains 25 malware families/classes. Implementing and Training various deep learning models: We have trained this data on CNN, VGG16, VGG19, RestNet50, Xception, Mobilenet. Evaluating model performance using appropriate metrics: The effectiveness of each model will be measured using metrics such as accuracy and loss. Analyzing results to identify the most effective approaches: Depending on the requirements and evaluation metrics, the best suited model would be selected amongst the other model trained.

2. LITERATURE REVIEW

1. "Malware Classification with Deep Convolutional Neural Networks" [10]

In order to classify malware binaries using Deep CNNs, Kalash et al.'s paper takes a novel technique by turning them into grayscale photos. CNNs can efficiently extract malware-specific patterns thanks to the visual transformation, which plays to CNN's advantages in image processing. The suggested model's high accuracy using the Microsoft and Malimg datasets highlights CNN's versatility in cybersecurity scenarios where visual representation increases the effectiveness of malware discovery.

Key Results: Shows the potential of CNNs for high-precision malware classification, achieving up to 99.97% accuracy on malware datasets. Additionally, the visual-based method improves interpretability by offering a mechanism to comprehend the structure of malware.

Cons: The method's high processing needs make it difficult to use in real-time application scenarios, and it has trouble generalizing to larger, more diverse malware datasets, which may limit its applicability.

2. "Image-Based Malware Classification using Deep CNN and Transfer Learning" [11]

The study by Pant and Bista uses CNN transfer learning to categorize viruses by altering images. Because pretrained models eliminate the need for big, malware-specific datasets, the method can be used even with constrained data resources. This approach highlights the potential of transfer learning in cybersecurity applications by improving efficiency and facilitating rapid model change.

Important Results: Transfer learning reduces computational requirements, allowing for smaller datasets and shorter model training times without sacrificing appreciable accuracy.

Cons: The accuracy of the approach may be limited for extremely subtle or uncommon malware kinds due to pretrained models' potential lack of specificity in identifying subtle virus changes.

3. "Malware Classification Framework using Convolutional Neural Network" [12] uses CNN's feature extraction capabilities to turn malware binaries into photos. This technique shows CNNs' potential in cybersecurity by allowing them to identify virus patterns in visual data, leading to correct categorization. However, the resource-intensive nature of the method might make it difficult to scale for large datasets.

Key Results: High classification accuracy is made possible by efficient feature extraction, offering a reliable technique for detecting malware in cybersecurity.

Cons: The framework's high processing requirement makes it difficult to use in dynamic contexts and restricts its scalability for large-scale or real-time malware investigation.

4. "Malware Classification Using Customized CNN with Leaky RELU" [13] Komatwarand Kokare's study improves malware classification by addressing the "dying ReLU" problem by customizing CNNs with Leaky ReLU. By concentrating on visual patterns unique to malware, this method improves CNN's feature extraction and achieves more accurate classification.

Key Outcomes: Leaky ReLU improves CNN responsiveness to malware features, resulting in increased accuracy and decreased overfitting, making it perfect for complicated malware classification.

Cons: Its applicability and generalizability across various malware kinds are limited by the necessity for extensive model tuning, which necessitates particular adaptations for various datasets.

5. "Image-Based Malware Classification using Ensemble of CNN Architectures (IMCEC) [14]

The IMCEC model classifies malware using an ensemble of CNNs, using different CNN architectures to capture a range of malware attributes. With improved accuracy and lower misclassification rates, this approach performs better than individual CNN models. However, ensemble methods can be unfeasible in real-time or resource-constrained environments due to their high resource requirements.

Key Results: The ensemble model improved detection across complicated malware categories by improving classification accuracy and robustness.

Cons: The model's high processing demand restricts its use for real-time applications and makes it difficult to implement in high-speed or low-power settings.

6. "Malware Classification Using Convolutional Fuzzy Neural Networks with Feature Fusion" [15] Lin, Huang, and Lee optimize feature selection using the Taguchi technique by combining CNNs with fuzzy neural networks. The feature fusion of this model provides good accuracy across a variety of datasets and allows for the categorization of intricate malware patterns. Additionally, the incorporation of fuzzy neural networks improves the model's adaptability in virus detection.

Key Outcomes: Feature fusion yields high accuracy, and the model can be used to malware with a variety of traits, making it practical for a range of applications.

Cons: Its usage in real-time malware detection is limited by its increased complexity and processing resource needs, which make deployment difficult in dynamic contexts.

3. METHODOLOGY

The fundamental technique for classifying malware families entails gathering a dataset of malware images, identifying pertinent attributes that can point to harmful intent, and then classifying which malware images are members of which malware families using deep learning models.

3.1 Dataset:

A deep learning model's prediction quality is closely correlated with the caliber of its training data. Since CNN and other Deep Learning models perform well on image data, we chose the Malimg Dataset from [16] and performed data preprocessing on it to ensure smooth operation. There are 9339 malware images in the Malimg Dataset, which are divided into 25 families and classes. An overview of the dataset is provided in the table below.

	Class	Family	#
1.	Worm	Allaple.L	1591
2.	Worm	Allaple.A	2949
3.	Worm	Yuner.A	800
4.	PWS	Lolyda.AA 1	213
5.	PWS	Lolyda.AA 2	184
6.	PWS	Lolyda.AA 3	123
7.	Trojan	C2Lop.P	146
8.	Trojan	C2Lop.gen!g	200
9.	Dialer	Instantaccess	431
10.	TDownloader	Swizzot.gen!I	132
11.	TDownloader	Swizzor.gen!E	128
12.	Worm	VB.AT	408
13.	Rogue	Fakerean	381
14.	Trojan	Alueron.gen!J	198
15.	Trojan	Malex.gen!J	136
16.	PWS	Lolyda.AT	159
17.	Dialer	Adialer.C	125
18.	TDownloader	Wintrim.BX	97
19.	Dialer	Dialplatform.B	177
20.	TDownloader	Dontovo.A	162
21.	TDownloader	Obfuscator.AD	142
22.	Backdoor	Agent.FYI	116
23.	Worm:AutoIT	Autorun.K	106
24.	Backdoor	Rbot!gen	158
25.	Trojan	Skintrim.N	80

Table -1: Malware's family and its quantity [16]

3.2 Data Preprocessing

The malware dataset that we are using is a processed version of Malimg dataset. Conversion of byte files to png files was done beforehand [16]. However, the method to visualize malware used by Fu et al. [17] proves to give better results as compared to the script that was used to convert the .byte files. In the method that Fu et al. [17] proposed, the malware is visualized through RGB images, with byte values, entropy, and section size mapped to the red, green, and blue channels. Their technique captures detailed global and local malware features, including Gray Level Co-occurrence Matrix (GLCM) textures, color moments, and distinct ASCII-convertible byte sequences, improving classification accuracy. In contrast, the script that was used generates simpler grayscale images by directly converting hexadecimal bytes into pixel intensities, capturing basic structural information without additional feature extraction or color channel separation, thus limiting classification detail.

A. Generating the Dataset

The malware images are classified based on the family or class they belong to into subfolders [18]. The process involved creating batches of normalized image tensors from directories representing malware families, enabling efficient processing for model training and testing. By setting the target size to 64x64 pixels, all images were resized to uniform dimensions, facilitating standardized input. With a dataset containing 9,339 images, a batch size of up to 9,339 ensured each batch could contain the entire dataset if necessary. Data was split into training and testing sets in a 70:30 ratio, with the method successfully recognizing 25 distinct classes based on folder names. Although visual inspection revealed differences among files, accurately distinguishing malware families visually remained challenging, underscoring the need for automated classification methods.

B. Balancing the Dataset

To address the class imbalance in our dataset, where over 30% of images belong to class 2 (Allaple.A) and 17% to class 3 (Allaple.L). We employed a strategy that adjusts class weights, assigning greater weight to minority classes and lower weight to the majority class. This adjustment is crucial to ensure the model does not become biased toward the majority class, thereby improving its ability to correctly classify the minority classes. We began by calculating the weights inversely proportional to class frequencies, which allowed us to set balanced class weights. This step is

essential to counter the imbalance and ensure that each class is fairly represented in the training process. The target labels in our training data were transformed to a single label format to fit this approach, facilitating the application of the calculated class weights.

C. Feature Extraction

Classifying malware based on grayscale imaging is a cutting-edge method that has demonstrated its effectiveness as a static analysis tool (Kancherla and Mukkamala, 2013; Nataraj et al., 2014). A grayscale image portrays different shades of gray, ranging from white to black, divided into 256 levels based on a logarithmic scale. This grayscale variation corresponds to distinct physical information in the images, with textures reflecting these grayscale differences in the visual field.

Numerous features are frequently used in texture analysis, with frequency content analysis being one of the most popular techniques. This method entails splitting a texture block's frequency domain into discrete areas, usually wedges that represent various orientations and rings that capture scale. To further describe the texture, characteristics are calculated within each region. Because they can capture significant patterns and structures within textures that may be difficult to see in the spatial domain alone, these frequency-based techniques are commonly used.

The efficacy of this method is supported by psychophysical research, which demonstrates how the human visual system spontaneously decomposes images according to frequency and direction in order to perceive textures. By removing directional features and spatial frequency information, this decomposition is consistent with how our perception interprets minute variations in texture. Algorithms can more accurately simulate human perception by modeling texture analysis similarly, allowing for more precise texture-based classifications and analyses. As a result, frequency domain analysis with scale and orientation segmentation has emerged as a key instrument in texture research, with several applications in domains like remote sensing, computer vision, and medical imaging.

Texture analysis, which finds unique patterns in the visual structure of malware images, is the basis of this classification method. Because Gabor filters function in both the spatial and frequency domains, they are renowned for their resilience in texture segmentation and classification. This makes it possible to analyze the frequency content of the image and extract detailed texture details. By breaking down the image into different scales and orientations, gabor filters capture distinct texture elements that highlight variations in malware structure.

Nataraj et al. in [16] came up first with the approach of visualizing malware as gray-scale images. In that work, they represented a malware as a gray-scale image in the range of [0, 255], where 0 being for black and 255 for white [17]. They could observe that the obtained images presented different sections which in turn represented different information about the malware.
We can identify key textural features specific to particular malware families by dissecting the image into their constituents. their features may include common coding patterns or differences in code structure across malware belonging to the same family.

This method uses GIST features, which are obtained via wavelet decomposition of the images, in addition to Gabor filtering [16]. Through the application of a steerable pyramid across many orientations and scales, GIST features efficiently capture both local and global attributes [16]. A local representation is computed for every image location, capturing minute features in structure and texture. A more compact representation is obtained by averaging these local properties across broader spatial regions. The result is a condensed feature vector that lowers dimensionality while preserving key aspects of the image [16]. With its structured, high-level overview of image content, this feature vector is ideal for classification tasks since it improves pattern identification and facilitates precise class differentiation.

CNN

We created a three-layer deep Convolutional Neural Network (CNN) specifically designed for classification tasks, with Rectified Linear Units (ReLU) on each activation layer, as used in [19]. In order to provide necessary nonlinearity into the network, the model starts with a two-dimensional convolutional layer and then moves on to a nonlinear activation layer in each instance. The network generates features that draw attention to spatial correlations in the input data by performing element-wise multiplication and summing in each convolutional layer. Because of its computing efficiency and capacity to resolve the vanishing gradient problem, which frequently impacts the bottom layers of deep networks, ReLU was chosen over alternative non-linear functions like tanh or sigmoid. Without sacrificing model performance, this decision enhances gradient flow and expedites training, enabling faster convergence.

Each convolutional layer is followed by a max-pooling layer in addition to ReLU to enhance feature extraction. This layer selects the maximum value within each sub-region that the filter covers by sliding a filter with a given stride over the input. Max pooling minimizes the output size by maintaining these maximum values, which lowers computation demands while keeping the most important features. The network's overall classification accuracy is increased by this simplified representation, which also improves the network's capacity to identify patterns.

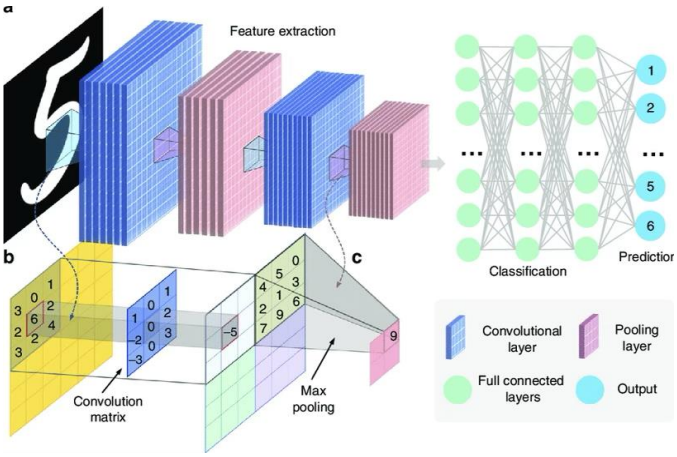

Figure -1: The typical CNN architecture for image-classification tasks. (L. Nataraj, 2011) [20].

VGG-16

We concentrate on choosing and getting the VGG-16 model ready for our malware image classification task in this step [21]. Deep convolutional neural network design VGG-16 is well known for its ability to perform well in image classification tasks, especially in the ImageNet competition. The architecture can learn rich feature representations from images since it has 16 learnable layers, including 3 fully connected layers and 13 convolutional layers.

We use a pre-trained version of VGG-16 in order to take advantage of its capabilities without having to train the model from scratch. A sizable dataset (ImageNet) with millions of photos in thousands of categories was used to train this model. Using a pre-trained model has the benefit of allowing it to extract significant features from photos because of the substantial training it has received. Transfer learning is the method by which we modify the acquired features to fit our particular malware classification goal.

Several fully connected (FC) layers at the conclusion of the original VGG-16 architecture are intended to classify images into ImageNet's 1,000 categories. We eliminate these FC levels since our job is to categorize malware photos into a varied number of classes (e.g., distinct malware families).

Adding Custom Classifier

After removing the FC layers, we add new layers tailored to our problem:

a) Global Average Pooling Layer - This layer takes the role of the final convolutional layer and averages across all spatial dimensions to reduce each feature map to a single value. This results in a more generalized representation and less overfitting.

b) Dense Layer - The number of neurons in one or more dense layers is equivalent to the number of malware classes we are classifying. For example, this layer will have 25 neurons if there are 25 malware families.

c) Activation Function - In order to interpret the outputs as class probabilities, a softmax activation function is applied to the output layer, converting raw scores into probabilities that add up to one.

Lastly, we use a suitable optimizer (such Adam or SGD) and loss function (categorical cross entropy for multi-class classification) to assemble the model. Additionally, we designate criteria for tracking throughout training, such as accuracy.

We successfully leverage VGG-16's potent feature extraction capabilities while customizing it for identifying various malware types based on their visual characteristics by following these thorough methods when customizing it for our malware classification project.

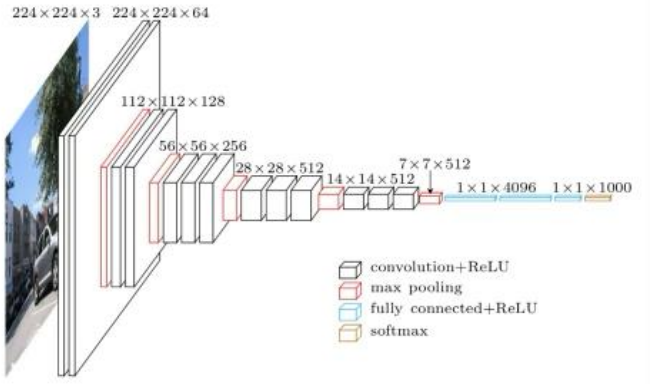

Figure -2: Architecture of VGG16 [22].

VGG-19

We employ a structured methodology akin to that of VGG-16 to apply VGG-19 for malware image classification, but we make use of the extra depth and capabilities of the VGG-19 architecture. VGG-19 can recognize more intricate patterns in photos because it has 19 layers with learnable parameters. A thorough explanation of our project's use of VGG-19, including implementation methods, is provided below.

Three fully connected layers and sixteen convolutional layers make up the 19 learnable parameter layers that make up VGG-19. Small receptive fields (3x3 filters) and a uniform architecture, in which the same filter size is applied across the network, are characteristics of the design. VGG-19 is very useful for picture classification jobs because of its design, which enables it to capture complex patterns and textures in photos.

To preserve the learnt weights from the ImageNet dataset during training, we first freeze the convolutional base of VGG-19. This indicates that this phase will just update the

weights of the recently added layers. By doing this, the model may take advantage of the rich feature representations that VGG-19 has already acquired, preventing overfitting and improving training efficiency. Without changing the pre-trained model's proven feature extraction capabilities, freezing the layers enables us to concentrate on optimizing the classifier for our particular malware classification task. This tactic guarantees a more reliable and efficient training procedure.

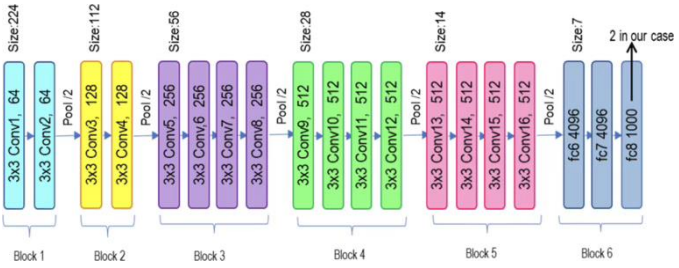

Figure -3: VGG-19 Architecture [23]

ResNetV2

Deep convolutional neural networks (CNNs) present issues [24], therefore we use ResNetV2.ResNetV2 is an extension of ResNet designed to solve those problems. It can maintain efficiency while reducing computing complexity by adding "bottleneck" blocks that compress feature maps. Batch normalization and ReLU activation are placed before convolutions in "reactivation" modules to minimize degradation and speed up convergence during training[24]. ResNetV2 outperforms its predecessor, exhibiting improved training efficacy and precision, particularly in more intricate network topologies. Applications such as object recognition, semantic segmentation, and picture classification have made extensive use of ResNetV2, a groundbreaking architecture in computer vision research. To preserve the learnt weights from ImageNet during training, we first freeze the convolutional basis of ResNetV2. This implies that the pre-trained weights will remain unchanged during training, with only the weights of our new layers being updated. This method expedites training and helps avoid overfitting.

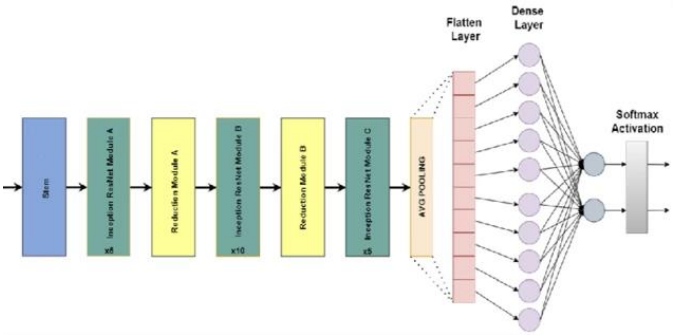

Figure -4: Inception-ResNet-V2 architecture designed for binary classification.

MobileNet Small

MobileNet Small is an adaptation of the MobileNet architecture intended for efficient deep learning on low-resource devices [24]. By employing depth-wise separable convolutions, it drastically reduces the model size and computational complexity while guaranteeing superior performance on embedded systems and mobile devices.

Despite its efficacy, MobileNet Small maintains a fair level of accuracy in tasks such as image classification and object detection [24]. This design choice, which permits real-time processing and lowers computational and energy costs, makes it the perfect choice for on-device AI applications.

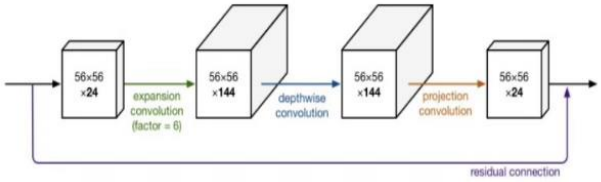

Figure -5: MobileNet Architecture [24].

4. RESULT AND DISCUSSION

We evaluated the performance of five deep learning models for malware classification in our research - CNN, VGG16, VGG19, MobileNet, Xception, ResNet50. The accuracy of these models was used to gauge their performance.

Table -2: Model Result

MODEL	ACCURACY	LOSS
CNN	.9607	.1588
VGG16	.9408	.2262
VGG19	.9236	.3176
RestNet50	.9600	.1600
Xception	.9853	.04956
MobileNet	0.9874	.0386

To evaluate the efficacy of various models in malware classification, six architectures were tested, yielding substantial performance differences. MobileNet achieved the highest classification accuracy at 98.74% with a loss of 0.0386, suggesting both robustness and efficiency. Xception followed closely, with 98.53% accuracy and a slightly higher loss of 0.04956, marking it as another strong choice for accurate malware categorization.

Other deep learning models also performed well, particularly CNN and ResNet50, achieving accuracies of 96.07% and 96.00%, with losses of 0.1588 and 0.1600 respectively. These

results indicate a high level of reliability, though slightly less precise than MobileNet and Xception. The VGG family models, VGG16 and VGG19, showed respectable performance with accuracies of 94.08% and 92.36% and higher losses of 0.2262 and 0.3176, respectively. While these accuracies are promising, the greater loss values suggest VGG models may be less effective for malware classification compared to the lightweight and efficient architectures. These findings underscore the advantage of using MobileNet and Xception for high-accuracy malware classification with minimal error, particularly beneficial for real-time detection applications where both speed and precision are critical

5. CONCLUSIONS

Our study explores a novel method of malware classification by using common image processing techniques to visualize malware binaries as grayscale images. Using several machine learning models, we were able to categorize malware samples with excellent accuracy due to the visual similarities within malware families. In particular, we used six models: CNN, VGG16, VGG19, ResNet50, Xception, and MobileNet. These models achieved classification accuracies ranging from 92.36% to 98.74%, with the best accuracy and lowest loss being shown by MobileNet and Xception. These models demonstrated the resilience of our method even against obfuscation techniques such as section encryption by successfully capturing unique image textures and patterns across malware families. Our approach is a computationally effective and scalable malware classification solution since it does not require code execution or disassembly. Additionally, the results indicate that by offering a dependable, texture-based classification strategy, this image-based approach can support conventional malware investigation tools. To further increase resilience against changing malware obfuscation techniques, future research could investigate the use of localized characteristics and adaptive structures. All things considered, this method shows encouraging promise for improving automated malware categorization and strengthening cybersecurity defenses.

REFERENCES

[1] Naway, Abdelmonim & LI, Yuancheng. (2018). A Review on The Use of Deep Learning in Android Malware Detection. 10.48550/arXiv.1812.10360.

[2] J. W. Stokes, D. Wang, M. Marinescu, M. Marino and B. Bussone, "Attack and Defense of Dynamic Analysis-Based, Adversarial Neural Malware Detection Models," MILCOM 2018 - 2018 IEEE Military Communications Conference (MILCOM), Los Angeles, CA, USA, 2018, pp. 1-8, doi: 10.1109/MILCOM.2018.8599855.

[3] Kolosnjaji, Bojan & Demontis, Ambra & Biggio, Battista & Maiorca, Davide & Giacinto, Giorgio & Eckert, Claudia & Roli, Fabio. (2018). Adversarial Malware Binaries: Evading Deep Learning for Malware Detection in Executables. 533-537. 10.23919/EUSIPCO.2018.8553214.

[4] Abdel Ouahab, I.B., Bouhorma, M., El Aachak, L. and Boudhir, A.A., 2022. Towards a new cyberdefense generation: Proposition of an intelligent cybersecurity framework for malware attacks. Recent Advances in Computer Science and Communications (Formerly: Recent Patents on Computer Science), 15(8), pp.1026-1042.

[5] News article from india news https://www.business-standard.com/india-news/malware-attacks-in-india-increase-by-11-22-jump-seen-in-ransomware-124080100800_1.html

[6] Bensaoud, A., Kalita, J. and Bensaoud, M., 2024. A survey of malware detection using deep learning. Machine Learning With Applications, 16, p.100546.

[7] Mcafee blog https://www.mcafee.com/blogs/other-blogs/mcafee-labs/the-rise-of-deep-learning-for-detection-and-classification-of-malware/

[8] Bensaoud, A., Kalita, J. and Bensaoud, M., 2024. A survey of malware detection using deep learning. Machine Learning With Applications, 16, p.100546.

[9] M. Kalash, M. Rochan, N. Mohammed, N. D. B. Bruce, Y. Wang, and F. Iqbal, "Malware Classification with Deep Convolutional Neural Networks," IEEE Conference Publication. [Online]. Available: IEEE Xplore.

[10] M. Kalash, M. Rochan, N. Mohammed, N. D. B. Bruce, Y. Wang, and F. Iqbal, "Malware Classification with Deep Convolutional Neural Networks," IEEE Conference Publication. [Online]. Available: IEEE Xplore.

[11] D. Pant and R. Bista, "Image-based Malware Classification using Deep Convolutional Neural Network and Transfer Learning," in Proceedings of the 2021 3rd International Conference on Advanced Information Science and System (AISS 2021), Sanya, China, Nov. 2021, doi: 10.1145/3503047.3503081.

[12] "Malware Classification Framework using Convolutional Neural Network," in 2020 International Conference on Cyber Warfare and Security (ICCWS), Oct. 2020, doi: 10.1109/ICCWS48432.2020.9292384.

[13] R. Komatwar and M. Kokare, "Malware Classification Using Customized Convolutional Neural Networks by Leaky ReLU Activated Function," Journal of Xi'an Shiyou University, Natural Science Edition, vol. 18, no. 3, pp. 186-200, 2020.

[14] D. Vasan, M. Alazab, S. Wassan, B. Safaei, and Q. Zheng, "Image-Based Malware Classification using Ensemble of CNN Architectures (IMCEC)," Computer Security Research.

[15] C. J. Lin, M. S. Huang, and C. L. Lee, "Malware Classification Using Convolutional Fuzzy Neural Networks Based on Feature Fusion and the Taguchi Method," International Journal of Advanced Computer Science and Applications.

[16] S. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, "Malware Images: Visualization and Automatic Classification," *Proceedings of the 8th International Symposium on Visualization for Cyber Security* (VizSec '11), 2011, pp. 1–7.

[17] J. Fu, J. Xue, Y. Wang, Z. Liu, and C. Shan, "Malware Visualization for Fine-Grained Classification," *IEEE Access*, vol. 6, pp. 14510–14523, 2018, doi: 10.1109/ACCESS.2018.2805301.

[18] R. Suresh, "Malware Classification Using Convolutional Neural Networks: Step-by-Step Tutorial," *Towards Data Science*, Apr. 17, 2019. [Online].

[19] Kabanga, E.K. and Kim, C.H., 2017. Malware images classification using convolutional neural network. *Journal of Computer and Communications*, *6*(1), pp.153-158.

[20] Zuo, Chao & Qian, Jiaming & Feng, Shijie & Yin, Wei & Li, Yixuan & Fan, Pengfei & Han, Jing & Qian, Kemao & Chen, Qian. (2022). Deep learning in optical metrology: a review. Light: Science & Applications. 11. 39. 10.1038/s41377-022-00714-x.

[21] Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv preprint arXiv:1409.1556.

[22] Step by step VGG16 implementation in Keras for beginners. https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c

[23] Khattar, Anuradha & Quadri, Syed. (2022). "Generalization of convolutional network to domain adaptation network for classification of disaster images on twitter". Multimedia Tools and Applications. 81. 10.1007/s11042-022-12869-1.

[24] Sharma, A.K.V., Swamy, H., Shetty, A. and Sathyanarayana, A.B., 2023. Malware Classification using Deep Neural Networks: Performance Evaluation and Applications in Edge Devices. *arXiv preprint arXiv:2310.06841*.

[25] Anis, Shazia & Xuan, Tan & Chuah, Joon Huang & Usman, Juliana & Qian, Pengjiang & Lai, Khin Wee. (2021). A comparative study of multiple neural network for detection of COVID-19 on chest X-ray. EURASIP Journal on Advances in Signal Processing. 2021. 10.1186/s13634-021-00755-1.