2024 International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE) | 979-8-3503-0641-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/IITCEE59897.2024.10467665

Deep Learning for Cyberthreats: Performance Analysis and Application of Malware Classification in Edge Computing

Akhil M R
Dept. of Computer Science
PES University
Bengaluru, India
akhil.mr2001@gmail.com

Adithya Krishna V Sharma
Software Engineer
Red Hat
Bengaluru, India
aadithya794@gmail.com

Abhijith Nadig
Dept. of Elec. and Comm.
JSS Academy of Technology
Bengaluru, India
abhijith.nadig@gmail.com

Pavan A
Dept. of Computer Science
PES University
Bengaluru, India
pavanappanna7@gmail.com

Ashray Shetty
Dept. of Computer Science
PES University
Bengaluru, India
ashrayshetty09@gmail.com

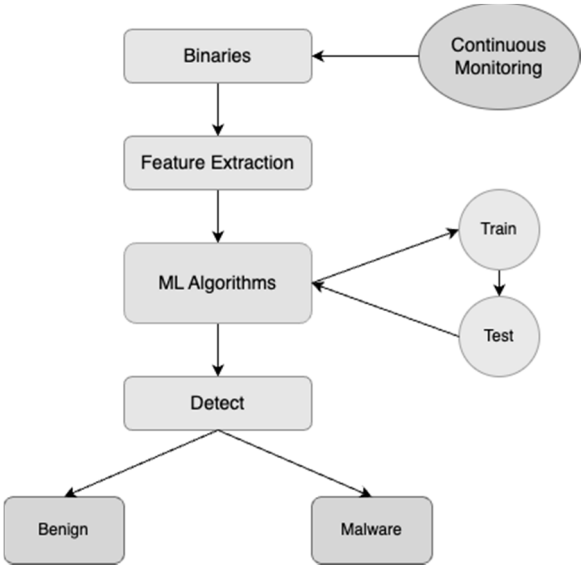
Sumathi A
Dept. of ECE
BNM Institute of Technology
Bengaluru, India
sumathia@bnmit.in

Abstract* — *In the context of the escalating malware threat landscape and the ongoing challenge of identifying modern malware, this research investigates the utility of Deep Neural Networks (DNNs) for malware classification. Various DNN architectures are explored, revealing their ability to accurately classify malware across different types with consistently high accuracy rates. The study also examines the feasibility of deploying DNN models on edge devices for real-time classification in resource-constrained settings, emphasizing the importance of efficient performance optimization. This research contributes to advancing malware detection techniques and highlights the significance of early malware detection for preventing adverse consequences. It additionally addresses the distribution of security tasks to edge devices to maintain the integrity and availability of large-scale IoT systems, fostering a more resilient and secure digital ecosystem.

Keywords— Cybersecurity, Data Protection, Deep Neural Networks, IoT Security, Malware Classification, Performance Evaluation

I. INTRODUCTION

Combating the constant spread of malware continues to be a major concern in the constantly changing world of cybersecurity. The integrity, confidentiality, and availability of digital assets are seriously threatened by malicious software, or malware, making accurate malware categorization a critical component of contemporary security systems. Long used for malware detection, traditional signature-based methods are ineffective against fast-evolving and zero-day malware. This calls for the adoption of more advanced methodologies. Deep neural networks (DNNs), a particularly noteworthy development in deep learning in recent years, have shown significant promise in several areas, including image identification, natural language processing, and autonomous systems. In the field of malware categorization, its capacity to automatically learn sophisticated patterns and features from raw data has attracted interest. DNNs can dramatically improve the precision and effectiveness of malware detection by sifting through malware samples and deriving useful representations.



Fig–1: ML based Malware Classification Flow

IoT devices are complex in nature and are subject to a wide variety of cyber-attacks with malware attacks being one of the prominent ones. Additionally with the increasing adoption of IoT devices in industries, these IoT systems will experience a rise in cyber-attacks. Therefore, it is deemed necessary to deploy efficient methodologies to detect and mitigate the adverse effects which would otherwise be caused by such malware attacks. According to Quoc-Dung Ngo et al. IoT malware detection techniques can be broadly classified into two domains, namely static analysis, and dynamic analysis [5].

Dynamic analysis includes having to execute the binaries and monitor for any malicious activity which could potentially infect the real time execution environment. In contrast, static analysis involves analyzing the binaries without executing them. [5] The methodologies explored in this paper leverage deep learning techniques to identify patterns and classify malware binaries without having to execute them.

Additionally, we cover a crucial topic of implementing advanced malware classification algorithms in contexts with limited resources. The computational and memory resources

of edge devices, such as Internet of Things (IoT) gadgets and low-powered computer systems, are constrained. For effective and real-time malware detection at the network edge, it is critical to assess the applicability of our deep neural network approach in such devices. Therefore, the computation time or the latency to classify malware binaries is measured once the trained model is obtained. This research intends to advance cybersecurity procedures by examining the functionality and applicability of our DNN-based malware classification methodology. By providing security experts with a cutting-edge tool for malware detection that is early and accurate, our research has the potential to increase the resilience of digital ecosystems to the ever-increasing cyberthreats

II. RELATED WORK

Based on previous research approaches of malware classification can be divided into two main categories: they are,

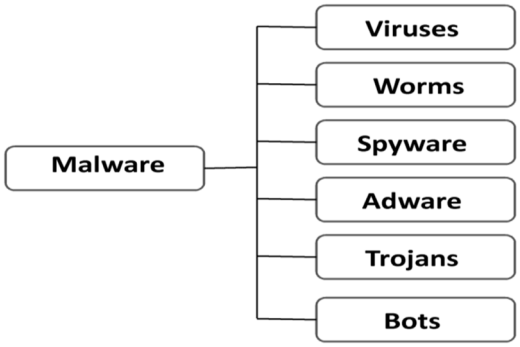


Fig –2: Types of Malwares

A. Non-Machine Learning Models

Traditionally, malware detection involved analyzing their signatures and traces either by applying principles of static analysis or dynamic signature analysis. Static malware analysis typically involves matching the semantics and structure of the malware executable with known pieces of malware executables in order to effectively classify the executable as malicious; without actually executing the malware executable file. Dynamic signature analysis on the other hand, typically involves the execution of the malware in a controlled and virtual environment such that there is no possibility of the malware to spread to other devices. In this scenario, the behavior of the suspected malware is analyzed in order to determine whether the executable is malicious or not. Although this approach is promising, it is complex, consumes extra resources and time. However, there are trade-offs between the two approaches and the drawbacks of static analysis include its lack of scalability and inability to detect new and unforeseen malware. As a result, researchers have turned to intelligent machine learning algorithms as an alternative approach.

Dynamic Analysis: Significant progress has been made by various researchers to propose malware detection methods which are primarily behavior-based that capture program behavior at runtime. One approach is to monitor the interactions between the operating system and the program by analyzing the of API calls. Some studies consider additional semantic information like the sequence of API calls and the use of graph representations to develop effective and robust systems. These approaches mainly scrutinize the chronological sequence of API calls, assess the impact of API calls on registers or extract behavioral graphs based on dependencies between API call parameters. In opposition to program-centric approaches, global, system-wide methods have been proposed, such as an access activity model by Lanzi et al. [8], which captures generalized interactions of benign applications with operating system resources, resulting in a low false positive rate. However, dynamic analysis techniques face challenges in handling execution-driven datasets, security precautions during experimentation, and dynamic anti-analysis defenses used by modern malware to evade detection.

Static Analysis: On the other hand, static approaches perform analysis without executing the program. The research literature demonstrates a wide variety of static analysis methods, with SAFE [11] and SAVE [10] being influential heuristic static malware detection approaches. These works proposed using different patterns to detect malicious content in executable files. Since then, numerous techniques have emerged based on different malware attributes, such as the header or body of the Portable Executable (PE) file, with analysis conducted on bytecode or by disassembling the code to extract opcodes and other relevant information. The main challenge in static analysis is coping with packing and obfuscation. Recently, generic approaches for the automatic de-obfuscation of obfuscated programs have been proposed. Additionally, static techniques have been employed to assess if a detected malware is like a previously-seen variant without performing costly unpacking.

B. Machine Learning Models

To improve on the deficiencies of non-machine learning approaches and capitalize on the shared behavior patterns among malware variants, anti-malware entities have formulated advanced categorization techniques utilizing data mining and machine learning methodologies. These approaches utilize diverse feature extraction techniques to construct intelligent systems for detecting malware, often using SVM-based classifiers, Naïve Bayes classifiers, or multiple classifiers [9].

For example, Nataraj et al. [7] proposes a tactic to depict malware through grayscale images is employed, and the GIST method is utilized to calculate texture characteristics. Subsequently, these features are subjected to classification through a k-nearest neighbor algorithm. However, these shallow learning techniques suffer from scalability issues with the growing number of malware samples and require manual feature engineering. To overcome these challenges, the current research focuses on developing deep learning architectures that are more robust and applicable to various malware samples.

While some techniques target superior performance on specific datasets, like the Microsoft Malware Dataset [12], we aim to construct a more versatile framework applicable to any type of malware sample. For instance, Drew et al. [13], [14] incorporated techniques based on gene classification for detecting malware on the Microsoft dataset.

An additional study outlined in [16] suggests the utilization of a Convolutional Neural Network (CNN) for the classification of malware. The researcher conducted experiments with three distinct architectures, augmenting the base model with an additional block comprising a

convolutional layer succeeded by a Max-pooling layer in each iteration. Nevertheless, their model maintains a relatively shallow structure. In contrast, our investigation focuses on exploring more intricate CNN architectures to enhance the classification of malware.

III. DATA AVAILABILITY AND PREPARATION

For the purpose of demonstrating the effectiveness of DNNs on malware binaries, the dataset chosen was MaleVis [6]. The MaleVis [6] dataset contains 14,226 malware images spanning across 26 classes which also includes 1 cleanware class. From the dataset, 10 malware classes were sampled and a total of 1400 images were further sampled from these classes overall for the purpose of training.

For testing and validation purposes, a total of 550 images were sampled spanning across the 10 classes. The images in the MaleVis [6] dataset was obtained by extracting the binary images from the malware files in 3 channel RGB format. The images are then resized into square sized resolutions of 224x224 and 300x300.

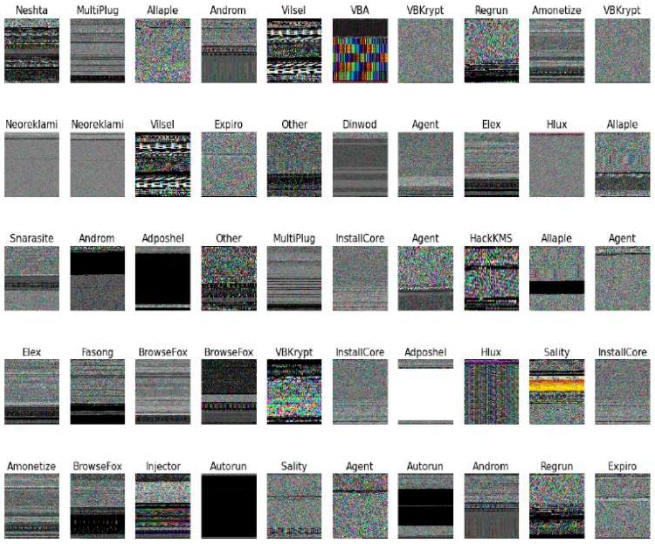


Fig -3: Images pertaining to the classes of the MaleVis dataset [6]

Table -1: Classes Sampled for the Purpose of Training

Class ID	Family	Malware Category	Sample Size
1	Adposhel	Adware	140
2	Agent	Trojan	140
3	Allaple	Worm	140
4	Amonetize	Adware	140
5	Androm	Backdoor	140
6	Autorun	Worm	140
7	BrowseFox	Adware	140
8	Dinwod	Trojan	140
9	Hex	Trojan	140
10	Expiro	Virus	140

IV. IMPLEMENTTION METHODOLOGY

The experimental setup involved training the models for 10 epochs on a system with RTX 3050 as the GPU. For increasing the effectiveness of the models, pre-trained imagenet weights were imported and applied before initiating the training process. A learning rate of 10^{-4} was used while also being configured to be adaptive in nature during the training process with the minimum allowed learning rate being 10^{-7}.

We run extensive tests to gauge the precision and effectiveness of our method to evaluate its performance. We do this by comparing the deep neural network's classification accuracy against unseen samples after training it on a broad array of malware samples. One of the key metrics used in the evaluation of resource efficiency is computational latency. This computational latency was measured as the time taken to classify the set of 550 test images. Other metrics such as accuracy, recall and F1 score were also taken into consideration while testing the model and are covered below. The details pertaining to the models utilized are explored as

A. ResNet V2

Deep convolutional neural networks (CNNs) present issues, therefore ResNetV2 is an extension of ResNet created to address those challenges. By introducing "bottleneck" blocks that compress feature maps, it can retain efficiency while lowering computational complexity. To reduce deterioration and hasten convergence during training, "pre-activation" modules place batch normalization and ReLU activation before convolutions. ResNetV2 performs better than its predecessor, especially in more complex network topologies, displaying increased training effectiveness and precision. ResNetV2, a pioneering architecture in computer vision research, has been widely used for image classification, object recognition, and semantic segmentation applications. Its breakthroughs advance state-of-the-art in image recognition applications by solving gradient problems and optimizing learning functions in deep CNNs.

B. DenseNet201

DenseNet201 is a deep convolutional neural network architecture that extends the DenseNet concept by employing 201 layers. It utilizes dense blocks, where each layer receives feature maps from preceding layers, facilitating feature reuse, and mitigating the vanishing gradient problem. This densely connected structure fosters efficient information flow and parameter sharing, resulting in improved memory utilization and better gradient propagation during training. With its substantial depth, DenseNet201 excels in learning complex patterns and representations from data. Its exceptional performance on benchmark datasets has solidified DenseNet201 as a leading architecture in the domain of machine intelligence for visual recognition tasks.

C. InceptionNetV3

The An improved convolutional neural network architecture called InceptionNetV3, sometimes known as Inception V3, was created for image identification applications. It provides numerous parallel convolutional

layers of various filter sizes to effectively capture features at various scales and resolutions, building on the strengths of its forerunners, InceptionNet and Inception V2. In order to icorporate both fine-grained and global characteristics, the "Inception module" concurrently uses 1x1, 3x3, and 5x5 convolutions. Meanwhile, "Factorized 7x7" convolutions lessen computational complexity without sacrificing the receptive field.

With the use of batch normalization and auxiliary classifiers, it also improves convergence and addresses the vanishing gradient issue. Global average pooling minimizes the number of parameters and avoids overfitting. InceptionNetV3 has been widely used for research and practical applications because of its exceptional performance in picture classification, object identification, and visual recognition tasks.

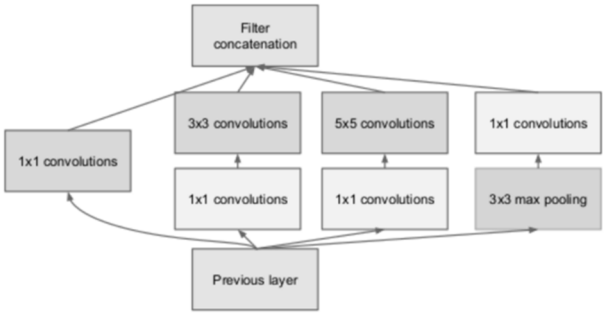


Fig -4: MobileNet Architecture

D. Xception

With The deep convolutional neural network architecture known as Xception, short for "Extreme Inception", was unveiled by Google and was motivated by the Inception idea. It uses "depth wise separable convolutions," which combine depth wise and pointwise convolutions, to replace conventional standard convolutions while maintaining accuracy.

Xception speeds up training and inference times by improving feature learning and parameter efficiency, making it the best choice for computer vision workloads, especially in contexts with limited resources like mobile devices and edge computing. Xception has established itself as a leading deep learning model and a popular option for image recognition applications thanks to its outstanding performance.

E. MobileNet Small

A variation of the MobileNet architecture called MobileNet Small is designed for quick and effective deep learning on devices with limited resources. It significantly decreases the model size and computational complexity by using depth wise separable convolutions, ensuring excellent performance on mobile devices and embedded systems.

In spite of its effectiveness, MobileNet Small retains respectable accuracy in jobs like object detection and image categorization. It is an ideal option for on-device AI applications because of this design decision, which enables real-time processing and reduces computational and energy expenses. MobileNet Small, which is widely used in edge computing applications, demonstrates its usefulness in enhancing deep learning for mobile and embedded devices.

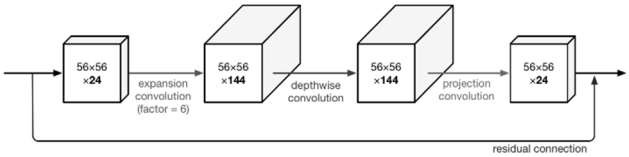


Fig -5: MobileNet Small Architecture

F. MobileNet Large

It is a lightweight deep learning architecture designed for efficient image classification on mobile devices. It also utilizes depth wise separable convolutions, a width multiplier, and a resolution multiplier to reduce computational complexity and model size. Despite its efficiency-focused design, MobileNet Large maintains competitive accuracy and is well-suited for real-time applications on resource-constrained devices, making it a significant advancement in the domain of computer vision.

In summary, MobileNet Small sacrifices some accuracy for even greater efficiency and compactness, making it ideal for scenarios where minimizing model size and computational requirements are critical, while MobileNet Large strikes a balance between efficiency and accuracy, making it more suitable for general-purpose mobile vision applications on devices with moderate resources.

V. RESULTS AND DISCUSSIONS

Table -2: Results Obtained from Training Various DNNs

Model	Compute Latency	Accuracy	Recall	F1 Score
ResNetV2	8.062	86.54	86.27	86.78
DenseNet201	10.87	94.54	94.43	94.42
InceptionNetV3	8.33	91.81	91.53	91.64
Xception	8.11	93.63	93.68	93.64
MobileNet-Small	3.51	85.63	84.52	81.77
MobileNet-Large	6.17	88.01	87.92	87.87

The above table summarizes the results obtained from testing various DNNs on the MaleVis [6] dataset. The compute latency depicts the time taken in seconds to classify 550 test images sampled from the dataset. It was observed that DenseNet201 achieved the highest accuracy in comparison to the other models during the test run, although a tradeoff between the computational latency and accuracy can be significantly noticed.

DenseNet201 showed the highest latency to compute along with an increased model accuracy. MobileNet-small on the other hand showed an accuracy on par with that of ResNetV2 with an exceptional computational latency of just **3.51 seconds**.

This proves that with effective fine tuning of the model, it could be deployed viably in real-world scenarios as well.

MobileNet-Large showed exceptional results achieving an accuracy higher than that of its smaller counterpart version, but with a slight tradeoff with the computational latency.

Furthermore, the above set of results can be utilized for choosing the right model for deployment in resource constrained scenarios as per the requirement and the availability of computational power in edge devices.

VI. CONCLUSION

In this survey article, we have explored the application of deep neural networks (DNNs) for malware classification. Malware detection and classification are critical tasks in today's cybersecurity landscape due to the ever-evolving nature of malicious threats. Traditional non-machine learning methods such as static and dynamic analysis have been widely used but are facing challenges in coping with the increasing complexity and diversity of malware.

The machine learning methods section focused on DNN architectures, namely ResNet, DenseNet, InceptionNet, Xception, MobileNet Small, and MobileNet Large. These DNNs have demonstrated promising results in various computer vision tasks and have shown potential for tackling malware classification as well.

From the performance evaluation, it is evident that DNN architectures can effectively detect and classify malware binaries with high accuracy and improved generalization. DenseNet201 showed the best performance among the models evaluated with an accuracy of 94.5. The ability to handle large-scale datasets and learn intricate patterns allows DNNs to discern even the most sophisticated malware variants. Moreover, transfer learning techniques can be leveraged to adapt pre-trained models on related tasks, reducing the data requirements and training time.

Regarding the applicability in edge devices, the compact nature of some DNNs like MobileNet Small and MobileNet Large allows for efficient deployment on resource-constrained devices, such as IoT devices and smartphones. The ability to perform classification on the edge can enhance real-time threat detection and response, mitigating the need for constant cloud communication and reducing latency.

However, societal concerns also need to be addressed when using DNNs for malware classification. There are ethical and privacy considerations related to data collection, model fairness, and potential misuse of these technologies. It is crucial to adhere to robust privacy policies and ensure the transparency and accountability of the deployed models.

REFERENCES

[1] Bozkir, Ahmet Selman, et al. "Utilization and Comparision of Convolutional Neural Networks in Malware Recognition." *2019 27th Signal Processing and Communications Applications Conference (SIU)*, IEEE, 2019.

[2] Chen, Yuanfang, et al. "Deep Learning for Secure Mobile Edge Computing." *ArXiv [Cs.CR]*, 2017, http://arxiv.org/abs/1709.08025

[3] Kalash, Mahmoud, et al. "Malware Classification with Deep Convolutional Neural Networks." *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, IEEE, 2018, pp. 1–5.

[4] Khoda, Mahbub E., et al. "Malware Detection in Edge Devices with Fuzzy Oversampling and Dynamic Class Weighting." *Applied Soft Computing*, vol. 112, no. 107783, 2021, p. 107783, doi:10.1016/j.asoc.2021.107783.

[5] Ngo, Quoc-Dung, et al. "A Survey of IoT Malware and Detection Methods Based on Static Features." *ICT Express*, vol. 6, no. 4, 2020, pp. 280–286, doi:10.1016/j.icte.2020.04.005.

[6] Pascanu, Razvan, et al. "Malware Classification with Recurrent Networks." *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2015, pp. 1916–1920.

[7] L. Nataraj, S. Karthikeyan, G. Jacob, and B. Manjunath, "Malware images: visualization and automatic classification," in Proceedings of the 8th international symposium on visualization for cyber security. ACM, 2011, p. 4.

[8] A. Lanzi, D. Balzarotti, C. Kruegel, M. Christodorescu, and E. Kirda. Accessminer: Using system-centric models for malware protection. In Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS '10, pages 399–412, New York, NY, USA, 2010. ACM.

[9] Kalash, M., Rochan, M., Mohammed, N., Bruce, N.D., Wang, Y., & Iqbal, F. (2018). Malware Classification with Deep Convolutional Neural Networks. 2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS), 1-5.

[10] A. H. Sung, J. Xu, P. Chavez, and S. Mukkamala. Static analyzer of vicious executables (save). In Proceedings of the 20th Annual Computer Security Applications Conference, ACSAC '04, pages 326–334, Washington, DC, USA, 2004. IEEE Computer Society

[11] M. Christodorescu and S. Jha. Static analysis of executables to detect malicious patterns. In Proceedings of the 12th Conference on USENIX Security Symposium - Volume 12, SSYM'03, pages 12–12, Berkeley, CA, USA, 2003. USENIX Association.

[12] "Microsoft malware classification challenge (big 2015)," https://www.kaggle.com/c/malware-classification, 2017, accessed: 2017-01-30.

[13] J. Drew, T. Moore, and M. Hahsler, "Polymorphic malware detection using sequence classification methods," in Security and Privacy Workshops. IEEE, 2016, pp. 81–87.

[14] J. Drew, M. Hahsler, and T. Moore, "Polymorphic malware detection using sequence classification methods and ensembles," EURASIP Journal on Information Security, vol. 2017, no. 1, p. 2, 2017.

[15] M. Ahmadi, D. Ulyanov, S. Semenov, M. Trofimov, and G. Giacinto, "Novel feature extraction, selection and fusion for effective malware family classification," in Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy. ACM, 2016, pp. 183–194.

[16] Gibert Llaurado, "Convolutional neural networks for malware classification," Master's thesis, Universitat Politecnica de Catalunya, 2016