Dynamic Malware Classification of Windows PE Files using CNNs and Greyscale Images Derived from Runtime API Call Argument Conversion

Md Shahnawaz
Department of CSE
NIT Rourkela,
Odisha, India
0009-0003-2936-246X

Bishwajit Prasad Gond
Department of CSE
NIT Rourkela,
Odisha, India
0000-0003-3640-0463

Durga Prasad Mohapatra
Department of CSE
NIT Rourkela,
Odisha, India
0000-0002-4824-7091

***Abstract*—Malware detection and classification remains a topic of concern for cybersecurity, since it is becoming common for attackers to use advanced obfuscation on their malware to stay undetected. Conventional static analysis is not effective against polymorphic and metamorphic malware as these change their appearance without modifying their behavior, thus defying the analysis by code structure alone. This makes it important to use dynamic detection that monitors malware behavior at runtime. In this paper, we present a dynamic malware categorization framework that extracts API argument calls at the runtime execution of Windows Portable Executable (PE) files. Extracting and encoding the dynamic features of API names, argument return values, and other relative features, we convert raw behavioral data to temporal patterns. To enhance feature portrayal, the generated patterns are subsequently converted into grayscale pictures using a magma colormap. These improved photos are used to teach a Convolutional Neural Network (CNN) model discriminative features, which allows for reliable and accurate malware classification. Results from experiments indicate that our method, with an average accuracy of 98.36% is effective in classifying different classes of malware and benign by integrating dynamic analysis and deep learning. It not only achieves high classification accuracy but also demonstrates significant resilience against typical evasion strategies.**

***Index Terms*—API call, Dynamic Analysis, Magma Colormap, Malware Classification, CNN, PE Files, API Call Arguments.**

I. INTRODUCTION

As there is an intensifying digitalization of services and infrastructure, the danger of cybercrime is rising in amplitude and sophistication. Organizations are facing an upswing of advanced cyberattacks fueled by geopolitics, vulnerabilities in the supply chain, and new technologies, according to the World Economic Forum's *Global Cybersecurity Outlook 2025*. Industry reports stress there is an imperative for intelligent, responsive security mechanisms due to changing threat vectors [1].

Malware detection has become increasingly harder in recent years with advancements in evasion and code obfuscation at an accelerated rate. Polymorphic and metamorphic malware examples modify their code at run time so that traditional static code inspection, as well as signature-based detection, is evaded by them [2]. To bypass these shortcomings, dynamic analysis methods have become popular. By observing run-time activities, particularly API call sequences and their arguments at run-time upon running Windows Portable Executable (PE) files, analysts can identify malicious intent with better reliability [3].

One of the latest developments in malware analysis is to convert such sequences of API calls to grayscale or RGB images to produce graphical representations that retain temporal as well as behavioral patterns. These images can further be classified by deep learning models, especially by Convolutional Neural Networks (CNNs), which are exceptionally efficient in extracting spatial features from images [4], [5]. CNNs have manifestly excelled in evasive threat detection based on their capability to learn hierarchically from raw inputs.

Along with standard CNN models, certain works explored advanced architectures and hybrid methods. For instance, Musaev et al. blended an optimized CNN with fine-tuning MobileNetV2 to achieve maximum performance on the Maling dataset, highlighting that methods based on ensembling perform well [6]. They proposed MVC-RSN, a CNN approach that aims to improve malware classification under adversarial environments [7]. A lightweight CNN architecture designed for effective malware detection in resource-constrained IoT environments was also presented by Yuan et al. [8].

Further expanding the application of deep learning for malware detection, Sweety et al. proposed a multi-view CNN model for both classification and signature generation, pointing to the strength of synergistically integrating behavioral views [9]. Divya presented a system named Mal Class, which employs deep CNNs for malware image classification with promising accuracy over various malware families [10]. Darwish and Roy compared federated learning, traditional machine learning, and deep learning in a study that showed that deep models such as CNNs outranked traditional means under all circumstances, especially with distributed or in IoT-based scenarios [11].

In this study, we propose a CNN-based visual malware classification system that leverages dynamic behavioral data.

By capturing runtime API call sequences and transforming them into images, we aim to extract discriminative features for accurate malware classification. The scope of our work includes preprocessing API logs, designing a CNN architecture tailored for malware patterns, and benchmarking the model's efficiency on real-world data. Our objective is to propose an effective, scalable, and adaptable malware detection framework suitable for deployment in evolving cyber threat landscapes.

The objective of this paper is to explore the effectiveness of using dynamically generated API call-based images to classify malware samples using a CNN architecture. By leveraging the power of visual data and deep learning, we aim at enhancing the robustness and accuracy of malware classification systems. The approach is further validated using a comprehensive experimental setup and evaluation metrics, contributing to the growing domain of visual malware classification research.

The remaining sections of the paper are formatted as follows: Section II outlines the basic concepts related to Malware Analysis and CNN. Section III reviews the existing related work. Section IV details the methodology used in this research. Section V describes the experimental setup employed. Section VI presents the results and their analysis. In Section VII, we compare our approach with state-of-the-art techniques. Finally, Section VIII concludes with future research directions.

II. BASIC CONCEPTS

A. Malware Analysis

Malware analysis is the practice of assessing malware to determine its intent, functional attributes, and potential risk. Security professionals can identify the various strains of malware, come up with appropriate countermeasures, and reinforce the defenses against cyber threats by studying malware carefully. There are two general categories into which the analysis techniques can be classified.

1) Static analysis: Static analysis refers to the process of examining a malware sample without running it. The method analyzes the internal structure of the code, such as embedded strings, file headers, and binary patterns. This aims to distill useful information (e.g., function calls, file dependencies, well-known signatures) that is helpful for enabling early detection.

2) Dynamic analysis: In contrast, dynamic analysis monitors malware behavior when it is operating in a controlled or sandboxed environment. Analysts can use this to watch how the operating system is affected by the malicious code, what it alters, and what external connections it tries to make. Such a real-time glimpse provides valuable insights into the malware's goals and potential destruction.

B. Sandboxing

Sandboxing refers to a security method whereby one runs suspected files or software in an isolated environment to identify their behavior without putting the actual system in danger. This approach revolutionizes the detection of zero-day exploits and the analysis of yet unrecognized malware variants. The sandbox emulates a real OS so that malware can behave naturally, but it is also isolated from the main machine so it cannot do any damage.

1) Cuckoo Sandbox Analysis Process: An open-source automated malware analysis tool called Cuckoo Sandbox enables dynamic executable analysis. When a file is executed inside the sandbox, Cuckoo collects information about its behavior and outputs a comprehensive behavioral report in JSON format. It contains information about API calls (invoked by the executable) with their order and frequency, arguments, process-level activity, network traffic, and regime changes. This detail is key for determining the malware's objective, spread methodology, and the damage that could be done to the target computer.

C. Convolutional Neural Networks (CNN)

This is followed by processing and analyzing visual data, which is called convolutional neural networks (CNN). They learn hierarchical features from images automatically, utilizing completely linked layers, pooling, and convolutions. Mathematically, the convolution operation, which is the basic building block of CNNs, can be described as:

$$\text{output}(i,j) = \sum_m \sum_n \text{input}(i+m, j+n) \cdot \text{kernel}(m,n) \quad (1)$$

In this equation:

- $\text{output}(i,j)$ is the output at pixel position (i,j).
- $\text{input}(i+m, j+n)$ corresponds to the pixel values in the input image at the specific position that is m and n units away respectively from the reference point given by (i,j).
- $\text{kernel}(m,n)$ is the filter (or kernel) applied at position (m,n) over the input image.

More specifically, a kernel is a matrix of arbitrary size that is used to slide over the image and take dot products as it moves around. CNNs are particularly powerful in fields like image classification and object detection because they can automatically recognize spatial hierarchies, minimizing the need for manual feature extraction for efficient learning from images.

The mapping of features to their images is IMAGE[EXEMPLAR][DENSITY][LIGHT][ANGLE], which indicates that each feature is represented as an image in image space.

Experimental Conversion of Features into Image: Data Conversion

The process included feature encoding to the matrix or image-like structure so that convolutional networks could be applied to the data. A very common function that maps a feature x to another one could be summed up with the following equation:

$$I(x,y) = f(\mathbf{X}) \quad (2)$$

In this equation:

- $I(x,y)$: pixel solution on the coordinates (x,y) of the converted image.

- $\mathbf{X}$: feature vector derived from the original high-dimensional data.
- $f(\mathbf{X})$ represents a function that manipulates the feature vector to give it an image-like appearance (through reshaping or normalizing it into a 2D grid).

Because CNNs find underlying spatial patterns in images, features can also be generalized using this data transformation and applied to any type of features which will allow CNN to learn the basic spatial patterns and structure relationships that are characteristic of the features, thus greatly aiding any prediction or analysis task.

III. RELATED WORK

Deep learning, particularly with Convolutional Neural Networks (CNNs), has drastically enhanced malware classification, primarily due to their exceptional ability to identify visual trends in the data. Traditional techniques like signature and heuristic-based methods can hardly detect obfuscated and polymorphic malware. Consequently, researchers are now increasingly targeting dynamic, image-based, and hybrid detection methods.

Sasikala and Shanmuganathan [5] proposed an image-based approach for malware classification based on a specially trained CNN. The approach converts malware binaries into both grayscale and RGB photo format imprints, which gets past the network and truly recognizes design. This method reached 10% level of accuracy and was resilient to many obfuscation techniques.

Sweety et al. [9] recently proposed a multi-view CNN-based system for dynamic malware detection and signature generation. By analyzing malware behavior from multiple perspectives, their method was able to accurately detect threats and generate precise signatures, highlighting the power of behavior-driven visual analysis.

Divya [10] presented *Mal Class*, a CNN-based architecture for automated classification of malware images. Using both monochrome and RGB image representations, the model achieved promising results on the Malimg dataset, showcasing CNNs' capacity for learning discriminative visual patterns even under adversarial variance.

Musaev et al. [6] proposed an ensemble method that integrates a custom CNN with a fine-tuned MobileNetV2. Their Optimized Epoch Selection strategy selects the best-performing model checkpoints, resulting in improved generalization and setting a new accuracy benchmark of 99.05% on the Malimg dataset.

Wu et al. [7] developed MVC-RSN, a malware classification method with variant identification capability. It utilizes ResNet-based architectures to improve generalization and robustness in detecting evolving malware variants, particularly in adversarial settings.

To address the limitations of computational overhead, Yuan et al. [8] designed a lightweight CNN (LCNN) optimized for IoT malware classification. Their approach utilized multidimensional Markov images derived from raw binaries and demonstrated over 99.35% accuracy on resource-constrained platforms.

Darwish and Roy [11] performed a comparative analysis of federated learning, deep learning, and traditional methods for IoT malware detection. Their findings indicated that deep CNNs consistently outperformed other models in distributed environments, underscoring the value of deep representations.

Javed and Amjad [3] explored the integration of dynamic behavioral analysis and SIEM systems with deep learning. Their approach processed API sequences extracted via Cuckoo Sandbox [17] and achieved significant accuracy improvements through few-shot learning, showcasing the benefit of incorporating contextual behavioral logs.

Gond et al. [13] proposed a deep learning framework for malware classification that integrates Natural Language Processing (NLP) techniques. By using n-grams of API call sequences, they successfully capture malware behavior patterns, achieving enhanced classification accuracy and robustness in contrast to conventional techniques.

Rajneekant et al. [14] conducted a comparative analysis of machine learning models based on API sequences for malware classification. Their results show that XGBoost outperforms other models, achieving an accuracy of 98.87%, demonstrating the effectiveness of incorporating API call sequences and arguments for precise malware detection.

Kishore et al. [15] proposed a hybrid analysis model for classifying malicious applications using computationally efficient machine learning techniques. Their method, which combines static and dynamic analysis, achieves high Matthews Correlation Coefficients (MCC), including 89% to classify malware and 81% to classify malware families, addressing class imbalance effectively.

Our work expands these core studies by creating a flexible malware classification system that turns API call sequences into grayscale images. We use a deep CNN model trained on these images to effectively pull out features and identify malware families. This blended method enhances both structural and behavioral detection while staying strong against evasion tactics.

IV. PROPOSED FRAMEWORK

Figure 1 illustrates our proposed work, which uses a multi-step system to identify malware by analyzing its behavior through image-based deep learning. The process involves preparing malware behavior logs, converting features into visual images, and training a Convolutional Neural Network (CNN) to classify them. The three key stages are: Preprocessing, Feature Transformation, and Model Training & Evaluation.

Phase 1: Analysis Phase

In Analysis phase, We carry out the following activities: After obtaining the PE files from VirusShare[1] and Virustotal[2] for all eight multiclass like Adware, Backdoor, Benign, Downloader,

[1] https://virusshare.com/

[2] https://www.virustotal.com/

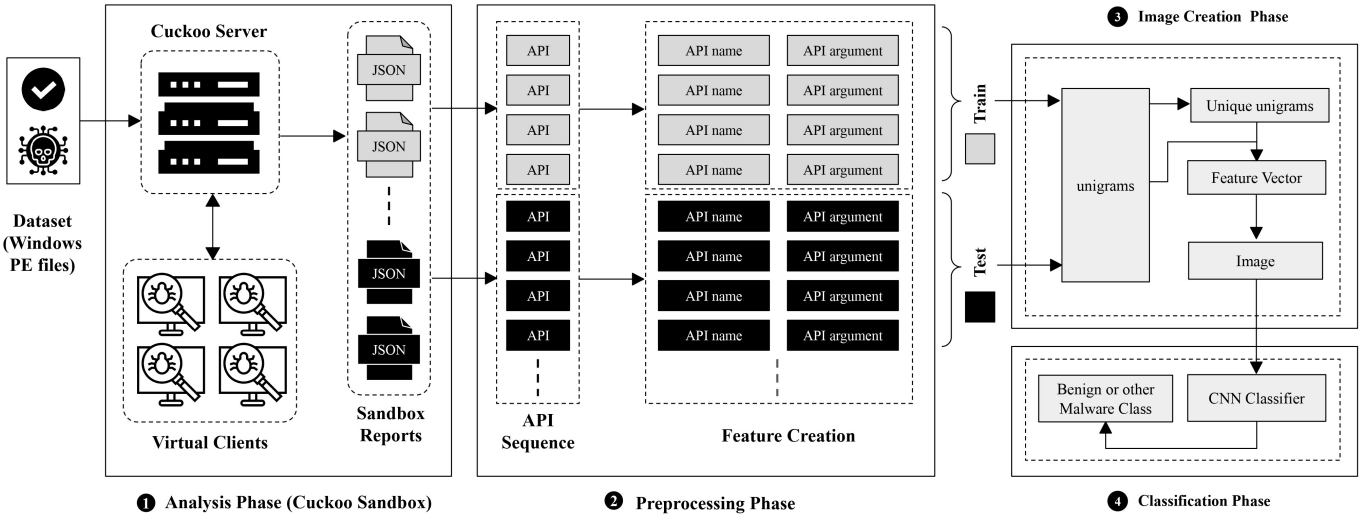

Fig. 1: Proposed CNN-based Architecture for Malware Analysis

Spyware, Trojan, Virus, and Worm we performed behavioral analysis using Cuckoo Sandbox, resulting in a behavioral report in JSON format. We then split the file into four parts: API category, API name, API argument, and API return. From this, We selected the API name and argument to create unigrams, where the API name is the first part and the API call argumant is added using underscores.

Phase 2: Preprocessing Phase

The following actions are taken during the preprocessing stage:

1) **API Dataset & Cuckoo Sandbox Analysis:** The data used in this study was collected from VirusShare, a well-known malware sample repository. Each sample was executed in a controlled environment using Cuckoo Sandbox, which generated behavioral reports in JSON format, capturing runtime interactions of PE (Portable Executable) files.

2) **API Feature Extraction:** From each behavioral JSON file, we extracted API-level elements such as `APICategory`, `APIName`, `APIArgument`, and `APIreturn`. These fields provide insight into how the malware interacts with the system.

3) **CSV File Construction:** For every malware sample, a single row was created in a CSV file. The first column contains the malware hash (unique identifier), the second column specifies the malware family label (e.g., backdoor), and the remaining columns store numeric values representing frequency or intensity of specific API calls.

This CSV file was then used as the structured input for further processing.

Phase 3: Feature Transformation Phase

This phase focuses on converting the structured CSV data into images[3] suitable for training with CNN.

1) **Normalization and Reshaping:** All numeric API values in each row were normalized into a range of 0–255. The resulting vectors were reshaped into square matrices (e.g., 128×128), generating grayscale image representations of API usage patterns for each malware.

2) **Image Enhancement Techniques:** To highlight important features and patterns in the image, the following techniques were applied:

- **Gaussian Blur** – To reduce noise and smooth variations.
- **CLAHE (Contrast Limited Adaptive Histogram Equalization)** – For improving local contrast.
- **Sobel Edge Detection** – To emphasize edge transitions and structural boundaries.

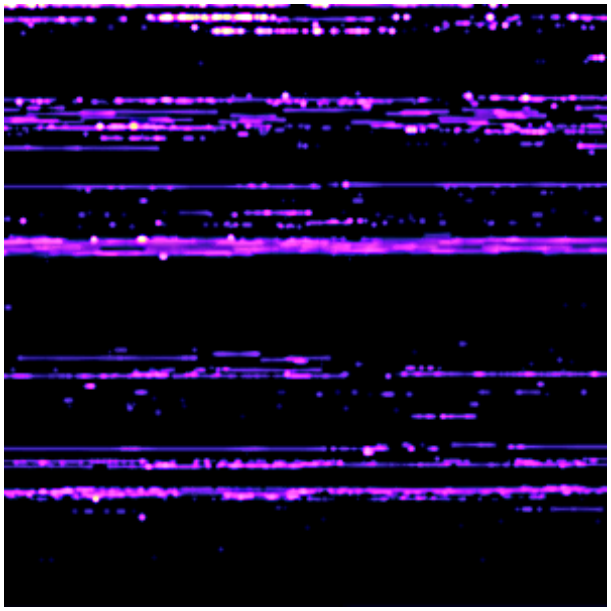

Fig. 2: A sample image after applying Magma Colormap

3) **Color Mapping and Saving:** A magma colormap was applied to the grayscale images, adding color richness for enhanced feature representation as shown in Figure 2. Finally,

[3]Code and Dataset

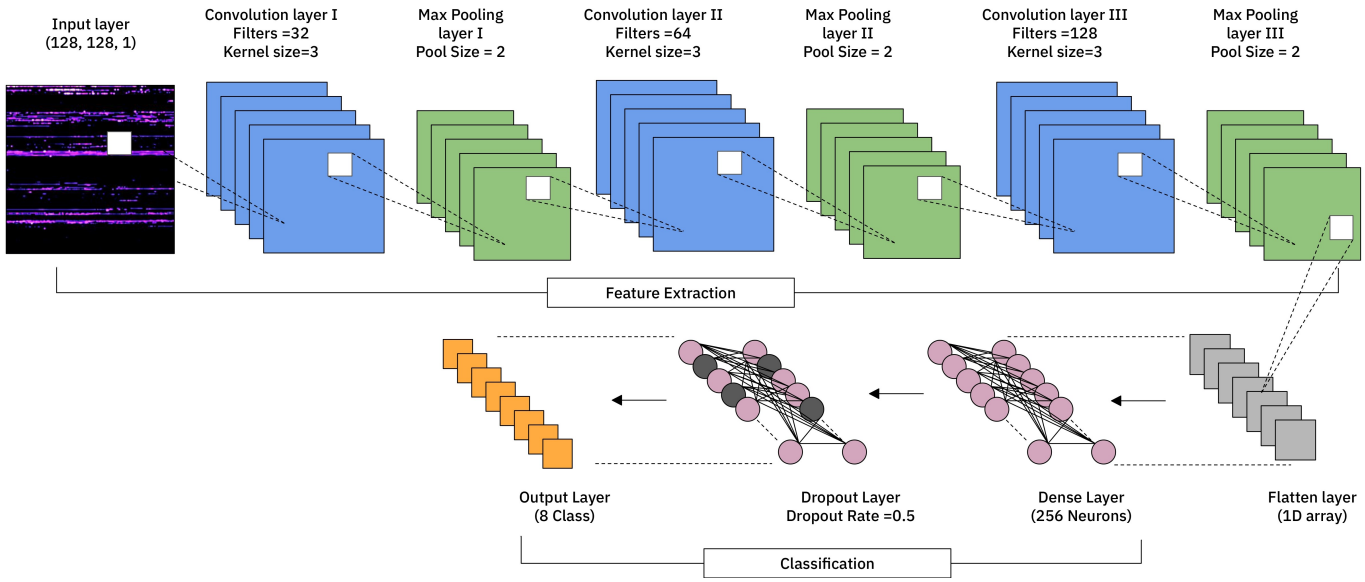

Fig. 3: CNN Model Architecture

contrast and sharpness improvements were applied before saving the images to disk for training.

Phase 4: Model Training and Evaluation Phase

This phase includes the design, training, and evaluation of a CNN model tailored for malware classification using grayscale images generated from API call patterns.

*1) **CNN Model Architecture**:* The architecture for our CNN model is defined as follows:

Input on layer: $(128, 128, 1)$
Convolutional Layer 1: $Filters = 32$, Kernel Size $= 3$, Activation $=$ ReLU
MaxPooling Layer 1: Pool Size $= 2$
Convolutional Layer 2: $Filters = 64$, Kernel Size $= 3$, Activation $=$ ReLU
MaxPooling Layer 2: Pool Size $= 2$
Convolutional Layer 3: $Filters = 128$, Kernel Size $= 3$, Activation $=$ ReLU
MaxPooling Layer 3: Pool Size $= 2$
Flatten Layer: Flattens the input to a 1D array
Dense Layer: Neurons $= 256$, Activation $=$ ReLU
Dropout Layer: Dropout Rate $= 0.5$
Output Layer: Neurons $= 8$, Activation $=$ Softmax

CNN Architecture: We designed a custom CNN from scratch as shown in Figure 3, comprising:

- Three convolutional layers with increasing filter sizes.
- Three Max pooling layers for downsampling after each convolution.
- A flattened layer followed by a dense layer with dropout for regularization.

The images were resized to 128×128 pixels and normalized before feeding into the network. To improve generalization, during training, data augmentation methods like flipping, zooming, and rotation were used.

*2) **Training and Output Generation:** :* The CNN model was trained for 100 epochs utilizing categorical cross-entropy loss and the Adam optimizer. The model artifacts, including trained weights, prediction results, evaluation plots, and training logs, were saved in a time-stamped results folder.

*3) **Performance Evaluation:** :* The trained model got evaluated using

- **Confusion Matrix:** Analyzing False Negatives, True Negatives, True Positives, and False Positives.
- **Classification Metrics:** Including F1-Score, Accuracy, Precision, and Recall.

These metrics collectively provided thorough explanation of how well the model could categorize different malware families.

V. EXPERIMENTAL SETUP

Our experimental setup was designed to evaluate the effectiveness of image-based malware classification using behavioral API features. It comprises the following components:

1) **Analysis Environment:**

- **Host System:** An Intel Xeon(R) Silver 4216 CPU with 128 GB RAM and 5TB HDD was used for executing the Cuckoo Sandbox, performing preprocessing, and training the deep learning model.

2) **Windows 10 Environment:**

- **OS:** Windows 10
- **Hardware:** A machine with a 5TB storage capacity, 128GB of RAM, an Intel i7 processor, and a 16GB NVIDIA GPU was utilized to collect and analyze the dynamic behavior of Cuckoo Sandbox reports.

3) **Development Environment:**

- **Programming Language:** Python Python 3.10.9 was used to implement the complete pipeline from data extraction to model evaluation.
- **IDE:** Spyder IDE facilitated efficient development, debugging, and visualization.

TABLE I: Datasets used

S.No	Types	Test Sample	Train Sample	Total Sample
1	Adware	316	1670	**1986**
2	Backdoor	228	446	**674**
3	Downloader	500	1999	**2499**
4	Spyware	167	779	**946**
5	Trojan	675	2893	**3568**
6	Virus	464	1928	**2392**
7	Worms	305	1052	**1357**
8	Benign	1857	6777	**8634**
	Total	**4512**	**17544**	**22056**

4) **Malware Samples:**

- A dataset consisting of 22,056 samples [18] was used, out of which 17,544 samples were utilized for training and 4,512 samples for testing the classification model as shown in Table I.

This setup facilitated controlled experimentation to evaluate the potential of converting API sequences into visual features for improving malware classification performance. All steps—from Cuckoo-based behavior capture to CNN-based classification—were performed within this environment.

VI. RESULT ANALYSIS

In this section, we evaluate the performance of the proposed malware classification model using a confusion matrix, detailed performance metrics, and a visual comparison of classification outcomes across various malware types.

A. Confusion Matrix Analysis

Figure 4 illustrates the confusion matrix, providing insight into the model's performance across eight malware types and benign samples. The model performs exceptionally well for the *benign* and *downloader* classes, achieving 1704 and 500 true positives, respectively, with almost negligible misclassifications. For the *trojan* class, the model correctly predicts 641 samples, but there are a few misclassifications into *benign* and *backdoor*. Classes like *adware* and *worm* show moderate confusion with other categories, highlighting areas where the model can benefit from further optimization or additional feature refinement.

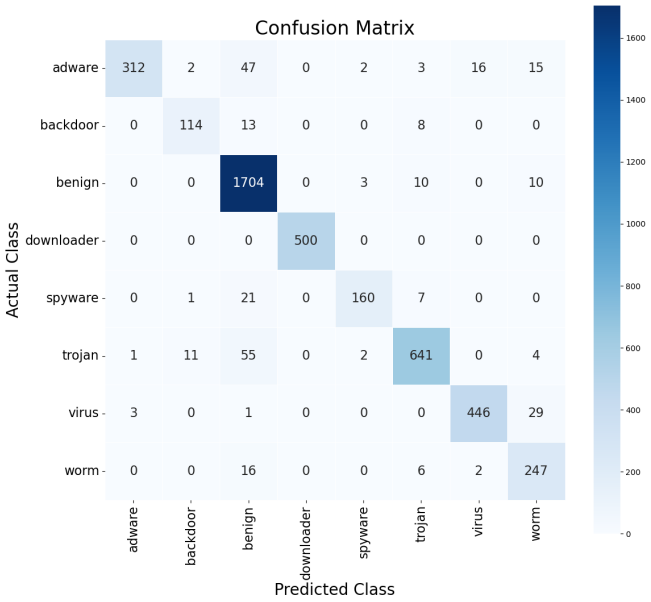

Fig. 4: Confusion Matrix of malware classification

TABLE II: Performance metrics for each malware type

No	Malware Type	Accuracy	F1 Score	Recall	Precision
1	Adware	98.00%	87.50%	78.60%	98.70%
2	Backdoor	99.20%	86.70%	84.40%	89.10%
3	Benign	96.00%	95.10%	98.70%	91.80%
4	Downloader	100.00%	100.00%	100.00%	100.00%
5	Spyware	99.20%	89.90%	84.70%	95.80%
6	Trojan	97.60%	92.30%	89.80%	95.00%
7	Virus	98.80%	94.60%	93.10%	96.10%
8	Worm	98.10%	85.80%	91.10%	81.00%

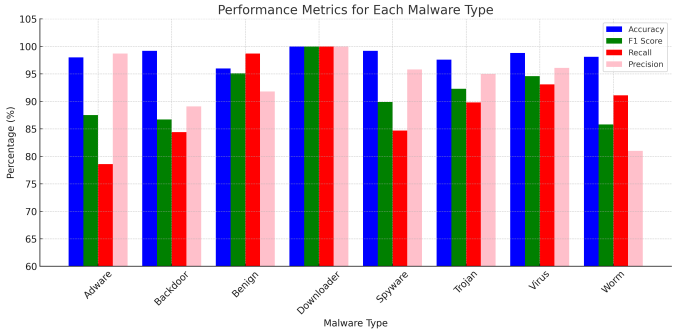

Fig. 5: Bar graph comparing Accuracy, F1 Score, Recall, and Precision for each malware type

B. Performance Metrics Overview

As shown in Table II, the model performs with high accuracy across all categories, with *downloader* achieving a perfect 100% in all metrics. *Benign* and *virus* categories also exhibit outstanding F1 scores and recall, indicating reliable detection with minimal false negatives.

Figure 5 provides a visual summary of performance, confirming the consistent and balanced results. It is evident that while *adware*, *spyware*, and *backdoor* show slightly lower recall, their precision remains high, suggesting correct positive detections when identified.

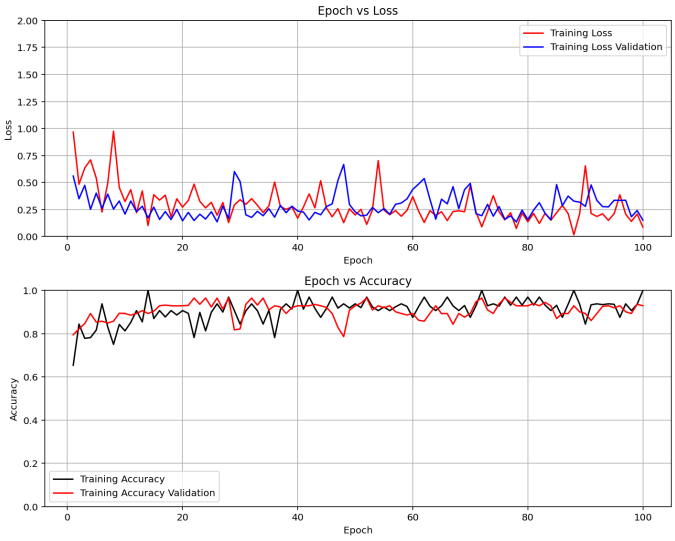

Fig. 6: Loss and Accuracy measures for proposed model

TABLE III: Evaluation metrics of malware classification

S. No.	Malware Type	TP	TN	FP	FN
1	Adware	312	4011	4	85
2	Backdoor	114	4263	14	21
3	Benign	1704	2532	153	23
4	Downloader	500	3912	0	0
5	Spyware	160	4216	7	29
6	Trojan	641	3664	34	73
7	Virus	446	3915	18	33
8	Worm	247	4083	58	24

Figure 6 shows, our model exhibits effective learning dynamics. The training and validation loss steadily decrease over epochs, indicating proper convergence. The accuracy plot reveals a high training accuracy and a consistently stable validation accuracy, suggesting that the model generalizes well. Minimal oscillations in validation accuracy imply controlled overfitting and good performance across unseen data.

C. True/False Positive-Negative Metrics

Table III summarizes the actual classification counts. Again, *downloader* is the top-performing class with perfect TP and TN and no FP or FN. Conversely, the *benign* class shows the highest false positive count (153), which may indicate that some benign samples resemble malware patterns.

In summary, the proposed CNN-based approach demonstrates strong classification performance, especially for key classes like *virus*, *downloader*, and *benign*. While minor recall dips are noted for a few malware types, the overall results confirm the model's effectiveness for real-world malware classification.

VII. Comparison of Our Work with Present State-of-the-Art Techniques

Our proposed approach focuses on classifying malware through dynamic analysis by converting API call sequences into grayscale images and leveraging a Convolutional Neural Network (CNN) for classification. While our dataset and image-generation method are unique, and hence, direct comparison is limited, we present a comparative discussion with existing deep learning-based techniques to highlight the effectiveness and distinctiveness of our method in Table IV.

Sweety et al. [9] developed a multi-view CNN framework that analyzes malware from several behavioral perspectives. Their approach effectively classified and generated malware signatures, leveraging dynamic execution features collected from sandbox environments. Although comprehensive in behavior analysis, their architecture is relatively more complex compared to our single-view grayscale image approach.

Divya [10] introduced *Mal Class*, a CNN-based deep learning method for classifying malware images. Their system used RGB and monochrome images converted from malware binaries. While the approach is similar in using image inputs, our work differs by utilizing behavioral API sequences rather than static binaries, thereby improving robustness against polymorphism.

Musaev et al. [6] proposed an ensemble of custom CNN and fine-tuned MobileNetV2 for image-based malware classification. Their technique, optimized with epoch selection, achieved 99.05% accuracy on the Malimg dataset. Our method, while simpler in architecture, is tested on a much larger and dynamically generated dataset, showcasing practical adaptability.

Wu et al. [7] developed MVC-RSN, a ResNet-based malware classification model with variant identification capabilities. Their work excels in generalization across adversarial malware, whereas our work emphasizes behavioral pattern capture through visual encoding of API sequences.

Yuan et al. [8] focused on lightweight CNNs for IoT malware detection. Their method achieved high accuracy with minimal resource usage, targeting constrained devices. Our method prioritizes robustness and behavioral coverage, suitable for enterprise-level malware detection systems.

Darwish and Roy [11] presented a comparative evaluation of federated learning, deep learning, and traditional models for

TABLE IV: Quantitative comparison of selected deep learning-based malware detection and classification techniques

No	Author	Image Type	Deep Learning Model	Dataset Used	Dataset Size	Detection	Cls^n	Acc
1	Musaev et al. [6]	Grayscale	Custom CNN + MobileNetV2	Malimg [12]	9339	✗	✓	99.05%
2	Wu et al. [7]	Binary	MVC-RSN (ResNet-based)	Custom	Not disclosed	✗	✓	98.6%
3	Yuan et al. [8]	Markov Images	Lightweight CNN	VirusShare [16]	Not disclosed	✓	✗	99.36%
4	Sweety et al. [9]	Feature Maps	Multi-View CNN	Custom	6000+	✓	✓	98%
5	Divya [10]	Grayscale / RGB	CNN	Malimg [12]	Not disclosed	✗	✓	98.99%
6	Darwish and Roy [11]	Not Image-Based	CNN / Federated Learning	NetworkX	6500+	✓	✓	95.27%
7	Proposed Work	Greyscale	CNN	VirusShare [16]	**22056**	✗	✓	98.36%

IoT malware detection. Their findings reinforce the strength of CNNs in distributed environments, supporting the foundational design of our image-based CNN model.

Our methodology stands apart due to its combination of runtime API call sequence analysis, n-gram extraction, grayscale image conversion, and CNN-based classification. We evaluated our model on a dataset of 22,056 samples spanning seven malware categories, capturing behavioral semantics through visual data. This hybrid approach makes our system robust, scalable, and suitable for deployment in dynamic cybersecurity environments.

VIII. CONCLUSION AND FUTURE WORKS

In this research, we proposed a new malware classification approach that transforms dynamically captured API call sequences into grayscale images and classifies them using a Convolutional Neural Network (CNN). By leveraging behavioral data and visual pattern recognition, our model effectively identified and classified seven major malware families across a dataset comprising 22,056 samples. The results demonstrated high classification performance, particularly for malware types such as Downloader, Backdoor, Spyware, Worms, Adware, Trojan, and Virus, with accuracy scores exceeding 97%. This image-based behavioral modeling provides a scalable and effective alternative to traditional static analysis methods, especially in handling obfuscation and evasive malware strategies.

Our study demonstrates the potential of integrating dynamic execution data with deep learning for robust malware detection. By converting API sequence patterns into visual representations, the model captures both structural and temporal features, enabling it to generalize across varied malware classes.

Below are some future directions to enhance this research:

- **LLM-Assisted API Interpretation:** Integrating Large Language Models (LLMs) to semantically analyze API arguments and contextual behaviors can enhance feature richness and interpretability, especially for novel or evolving threats.
- **Cross-Platform Compatibility:** Extending this framework to classify Android and Linux-based malware using system call equivalents could broaden its applicability across platforms.
- **Model Compression:** Exploring lightweight CNNs and pruning techniques will be important for deploying the model in resource-constrained environments like IoT gateways or mobile endpoints.
- **Advanced Similarity Learning:** Incorporating Siamese networks or triplet loss could provide a deeper understanding of intra-class and inter-class similarity in malware behaviors.
- **Temporal Sequencing:** Combining CNN with temporal models like LSTM or Transformer-based architectures may improve understanding of long-range behavioral dependencies in complex malware routines.

Through these future extensions, we aim to evolve our malware classification system into a highly accurate, adaptive, and lightweight solution capable of defending against next-generation cyber threats in real-time.

REFERENCES

[1] World Economic Forum. **Global Cybersecurity Outlook 2025.** 2025. [Online]. Available: https://reports.weforum.org/docs/WEF_Global_Cybersecurity_Outlook_2025.pdf [Accessed: 2025-04-13].

[2] Hebish, Mohamed Wael, and Mohamed Awni. *"CNN-Based Malware Family Classification and Evaluation."* In **2024 14th International Conference on Electrical Engineering (ICEENG)**, pp. 219–224. IEEE, 2024. [Online]. Available: https://doi.org/10.1109/ICEENG58856.2024.10566448

[3] Javed, Sheikh Muhammad Zeeshan, and Muhammad Faisal Amjad. *"Enhancing Malware Classification Through Dynamic Behavioral Analysis, SIEM Integration, and Deep Learning."* In **2024 International Seminar on Intelligent Technology and Its Applications (ISITIA)**, pp. 663–668. IEEE, 2024. [Online]. Available: https://doi.org/10.1109/ISITIA63062.2024.10667741

[4] Peng, Derek, Mohammed Husain, Abdullah Siddiqui, and Srijit Bhattacharya. *"Visual Malware Classification Using a CNN."* In **2024 IEEE MIT Undergraduate Research Technology Conference (URTC)**, pp. 1–5. IEEE, 2024. [Online]. Available: https://doi.org/10.1109/URTC65039.2024.10937559

[5] Sasikala, L., and C. Shanmuganathan. *"Effective Malware Classification using Fine-tuned CNN Architecture: An Image-Based Approach."* In **2024 2nd International Conference on Networking and Communications (ICNWC)**, pp. 1–6. IEEE, 2024. [Online]. Available: https://doi.org/10.1109/ICNWC60771.2024.10537444

[6] Musaev, Akobir, Abdulaziz Anorboev, and Jonghee M. Youn. *"Optimized Epoch Selection Ensemble: Integrating Custom CNN and Fine-Tuned MobileNetV2 for Malimg Dataset Classification."* **IEEE Access**,

vol. 13, pp. 45623–45633, 2025. [Online]. Available: https://doi.org/10.1109/ACCESS.2025.3547791

[7] Wu, Wei, Haipeng Peng, Haotian Zhu, and Lixiang Li. *"MVC-RSN: A Malware Classification Method With Variant Identification Ability."* **IEEE Internet of Things Journal**, vol. 11, no. 21, pp. 35654–35668, 2024. [Online]. Available: https://doi.org/10.1109/JIOT.2024.3436903

[8] Yuan, Baoguo, Junfeng Wang, Peng Wu, and Xianguo Qing. *"IoT Malware Classification Based on Lightweight Convolutional Neural Networks."* **IEEE Internet of Things Journal**, vol. 9, no. 5, pp. 3770–3783, 2022. [Online]. Available: https://doi.org/10.1109/JIOT.2021.3100063

[9] G, Sweety Prasanna Kiruba, Shubha G. Sanu, K. Saranya, Tatiraju V. Rajani Kanth, and S. Mahesh. *"Dynamic Malware Classification and Signature Generation Using Multi-View Convolutional Neural Networks."* **In 2024 International Conference on Integrated Intelligence and Communication Systems (ICIICS)**, pp. 1–7. IEEE, 2024. [Online]. Available: https://doi.org/10.1109/ICIICS63763.2024.10859526

[10] S, Divya. "*Mal Class: A Deep Learning Approach for Automatic Classification of Malware Images.*" In **2025 4th International Conference on Sentiment Analysis and Deep Learning (ICSADL)**, pp. 1329–1333. IEEE, 2025. [Online]. Available: https://doi.org/10.1109/ICSADL65848.2025.10933410

[11] Darwish, Rami, and Kaushik Roy. *"Comparative Analysis of Federated Learning, Deep Learning, and Traditional Machine Learning Techniques for IoT Malware Detection."* **In 2025 IEEE 4th International Conference on AI in Cybersecurity (ICAIC)**, pp. 1–10. IEEE, 2025. [Online]. Available: https://doi.org/10.1109/ICAIC63015.2025.10849203

[12] Nataraj, Lakshmanan, Sreejith Karthikeyan, Gregoire Jacob, and Bangalore S. Manjunath. *"Malware images: visualization and automatic classification."* In Proceedings of the **8th international symposium on visualization for cyber security**, pp. 1-7. 2011. [Online]. Available: https://paperswithcode.com/dataset/malimg [Accessed: 2025-04-13].

[13] Gond, Bishwajit Prasad, Atul Kumar Singh, and Durga Prasad Mohapatra. *"A Deep Learning Framework for Malware Classification using NLP Techniques."* In **2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT)**, pp. 1–8. IEEE, 2024. [Online]. Available: https://doi.org/10.1109/ICCCNT61001.2024.10725427

[14] Rajneekant, Pushkar Kishore, Bishwajit Prasad Gond, and Durga Prasad Mohapatra. *"Enhancing Malware Classification with Machine Learning: A Comparative Analysis of API Sequence-Based Techniques."* In **2024 IEEE International Conference on Smart Power Control and Renewable Energy (ICSPCRE)**, pp. 1–6. IEEE, 2024. [Online]. Available: https://doi.org/10.1109/ICSPCRE62303.2024.10675011

[15] Kishore, Pushkar, Swadhin Kumar Barisal, and Durga Prasad Mohapatra. *"Family Classification of Malicious Applications using Hybrid Analysis and Computationally Economical Machine Learning Techniques."* In **2022 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)**, pp. 442–449. IEEE, 2022. [Online]. Available: https://doi.org/10.1109/WI-IAT55865.2022.00072

[16] VirusShare. *VirusShare Malware Repository*. [Online]. Available: https://virusshare.com/ [Accessed: 2025-04-14].

[17] Cuckoo Sandbox. **Automated Malware Analysis**. [Online]. Available: https://cuckoosandbox.org/about.html [Accessed: 2025-04-14].

[18] Bishwajit Prasad Gond and Md Shahnawaz, *Malware Benign Image Classification Dataset*, 2025, **Kaggle**. Available at: https://doi.org/10.34740/KAGGLE/DSV/11412547