Malware Classification using Image Analysis

Prof. Syam Gopi
Dept. of Computer Science
Amal Jyothi College of Engineering (Autonomous)
Kottayam, India
syamgopi@amaljyothi.ac.in

Evelyn Susan Jacob
Dept. of Computer Science
Amal Jyothi College of Engineering (Autonomous)
Kottayam, India
evelynsusanjacob2025@cs.ajce.in

Joel John
Dept. of Computer Science
Amal Jyothi College of Engineering (Autonomous)
Kottayam, India
joeljohn2025@cs.ajce.in

Raynell Rajeev
Dept. of Computer Science
Amal Jyothi College of Engineering (Autonomous)
Kottayam, India
raynellrajeev2025@cs.ajce.in

Steve Alex
Dept. of Computer Science
Amal Jyothi College of Engineering (Autonomous)
Kottayam, India
stevealex2025@cs.ajce.in

***Abstract*—Malware detection and classification have evolved significantly with the integration of pattern recognition and image classification techniques. A pioneering study by Nataraj et al. (2011) [1] demonstrated that malware binaries could be visualized as grayscale images, revealing structural and textural similarities among malware families. Inspired by this approach, this research explores the effectiveness of deep learning-based architectures, specifically the hybrid CoatNet model, in improving malware classification accuracy. Using the MalImg dataset, we investigate the performance of CoatNet in identifying and categorizing various malware families, comparing its results with traditional image-processing-based classification methods. Our findings indicate that deep learning techniques offer superior accuracy and robustness in detecting malicious software without requiring code disassembly or execution. As malware threats continue to proliferate, advanced AI-driven approaches provide a scalable and efficient solution for real-time malware detection and cybersecurity enhancement.**

***Index Terms*—Malware detection, image classification, deep learning, CoatNet, cybersecurity, MalImg dataset, pattern recognition, artificial intelligence, malware analysis.**

I. Introduction

A. Background:

Malware continues to be a significant cybersecurity threat, affecting individuals, organizations, and critical infrastructures worldwide. Traditional detection techniques, such as signature-based and heuristic methods, often struggle against evolving malware variants, necessitating more robust approaches. In 2011, Nataraj et al. introduced an image-based malware classification technique, visualizing malware binaries as grayscale images to identify structural similarities among malware families. This method eliminated the need for disassembly or execution, offering a promising alternative to traditional detection methods. However, as malware complexity increases, more advanced classification techniques are required. Recent advancements in deep learning have brought significant improvements to image classification, with hybrid neural networks like CoAtNet demonstrating exceptional performance. CoAtNet, developed by Google Research, combines the strengths of Convolutional Neural Networks (CNNs) and Transformer-based attention models. [2] While CNNs excel in capturing local patterns, Transformers offer superior long-range dependencies. CoAtNet effectively integrates both approaches by unifying depthwise convolution and self-attention through relative attention mechanisms. Its architecture, which strategically stacks convolution and attention layers, has been shown to improve generalization, capacity, and efficiency. Given the success of deep learning in image recognition, applying CoAtNet to malware classification presents an opportunity to enhance detection accuracy and robustness.

B. Problem Statement:

Although image-based malware classification has shown promise, existing methods often rely on conventional image processing and machine learning techniques, which may not fully exploit deep learning's potential. Many approaches struggle with scalability, generalization to new malware families, and handling the growing complexity of malicious software. CNNs, while effective, may lack the model capacity needed for high-dimensional pattern recognition, whereas Transformers, despite their larger capacity, sometimes suffer from poor generalization. CoAtNet addresses these limitations by merging CNNs and Transformer models, offering a hybrid architecture optimized for both feature extraction and long-range dependencies [5]. However, its application in malware classification remains unexplored. This study aims to bridge this gap by evaluating CoAtNet's effectiveness on the MalImg dataset and comparing its performance with existing malware classification techniques.

C. Objectives

This research aims to:

- Explore the potential of deep learning-based image classification for malware detection.

- Implement and analyze the performance of CoAtNet for malware family classification using the MalImg dataset.
- Compare CoAtNet's accuracy, efficiency, and scalability with traditional CNN-based malware classification models.
- Assess whether CoAtNet's hybrid architecture improves generalization and robustness in detecting diverse malware families.
- Provide insights into leveraging advanced deep learning techniques for real-world cybersecurity applications.

II. Literature Review

A. Datasets

This study utilizes the MalImg dataset, a widely used benchmark dataset for malware classification through image analysis [3]. The dataset consists of grayscale image representations of malware binaries, converted from raw executable files. It contains samples from 25 different malware families, providing a diverse and challenging classification task. The dataset is split into training and validation sets, with the training set containing the majority of the samples for model learning [4]. For preprocessing and experimentation, we use an ImageDataGenerator to load images from their respective directories and apply real-time augmentation techniques. The dataset is structured into:

- Training Set: Used for model learning.
- Validation Set: Used to evaluate model performance and generalization.
- Test Set: A subset of the data is set aside for final evaluation.

A visualization of the dataset shows the distribution of different malware families, confirming class imbalances that need to be addressed in the training process.

B. Types of Malware

The MalImg dataset consists of grayscale image representations of malware binaries, categorized into 25 different malware families. Each malware type exhibits unique structural patterns when visualized as an image, allowing deep learning models to classify them based on texture and layout similarities. The primary types of malware in this dataset include:

1) Backdoors: Backdoors are malware programs that allow attackers to gain unauthorized remote access to a compromised system. These malware families in the MalImg dataset often exhibit dense, repetitive patterns in their visual representations, reflecting their structured codebase.
2) Trojans: Trojans disguise themselves as legitimate software to trick users into execution. Unlike worms, they do not spread independently but rely on user interaction. Malware families categorized as Trojans in the dataset often have irregular, fragmented structures, indicating obfuscation techniques used to evade detection.
3) Worms: Worms are self-replicating malware that spread across networks without user intervention. The worm families in the MalImg dataset tend to have highly repetitive patterns in their image representations, mirroring their ability to replicate themselves across systems.
4) Rootkits: Rootkits are designed to hide the presence of malware on infected systems, often embedding themselves deep into system processes. The rootkit samples in the dataset show scattered, obfuscated patterns, making them more challenging to detect through traditional methods.
5) Spyware: Spyware collects sensitive information from an infected system, such as login credentials or browsing history, and transmits it to an attacker. Spyware families in the MalImg dataset often exhibit more compact and dense image structures, reflecting their focus on stealth and data collection.

Each of these malware families has distinct visual characteristics when represented as images, making them suitable for classification using deep learning-based approaches like CoAtNet. By analyzing these structural patterns, deep learning models can effectively differentiate between various malware types, offering an advanced approach to malware detection and classification.

III. Malware Preprocessing Methods

To effectively classify malware using deep learning, raw executable binaries in the MalImg dataset must be preprocessed into structured image representations. This section outlines the technical pipeline used for preprocessing, ensuring the data is well-formatted for training deep learning models like CoAtNet.

A. Binary to Image Conversion

Each malware sample in the MalImg dataset is originally a Windows Portable Executable (PE) binary. To transform these binaries into images: Each byte (0–255) in the binary file is mapped directly to a grayscale pixel of the same intensity. The binary sequence is reshaped into a 2D array, with dimensions determined heuristically based on file size to maintain aspect ratio. The resulting images capture the structural and textural patterns of the malware, which serve as discriminative features for classification.

B. Image Resizing and Normalization

To standardize input dimensions across all malware samples: Each grayscale image is resized to 64×64 pixels using bilinear interpolation. This ensures uniformity across varying malware sample sizes and optimizes model performance. Pixel values are normalized to the range [0,1] by dividing by 255, preventing dominance of large intensity values and aiding gradient-based optimization during training.

```
x_train, x_test, y_train, y_test=
train_test_split(imgs/255., labels,
test_size=0.3)
```

Fig. 1. MalImg dataset.

C. One-Hot Encoding of Labels

Since the dataset consists of 25 malware families, the classification task is multi-class. To prepare the labels: Each malware sample is assigned a one-hot encoded vector of size 25. This encoding represents each class as a binary vector, enabling categorical cross-entropy loss during training.

D. Data Augmentation

To enhance model generalization and mitigate dataset imbalance, real-time data augmentation is applied:

- Random Rotation: Helps the model generalize spatial variations in malware structures.
- Zooming & Scaling: Ensures robustness against malware samples of varying sizes.
- Horizontal & Vertical Flipping: Compensates for orientation differences in binary-to-image transformations.

The ImageDataGenerator in TensorFlow is used to apply these transformations dynamically during training:

```
    datagen = ImageDataGenerator(
    rotation_range=20,
    zoom_range=0.2,
    horizontal_flip=True,
    vertical_flip=True
)
    train_batches =
    datagen.flow_from_directory(
    directory=train_root_path,
    target_size=(64,64), batch_size=32)
```

E. Dataset Splitting

The dataset is split into:

1) Training Set (70%) – Used for model learning.
2) Validation Set (30%) – Used to tune hyperparameters and monitor generalization.

The train_test_split function from Scikit-Learn is used for partitioning:

```
    from sklearn.model_selection
    import train_test_split
    x_train, x_test, y_train, y_test =
    train_test_split(imgs, labels,
    test_size=0.3, random_state=42)
```

F. Dataset Visualization

To inspect the preprocessed images and label distribution: A subset of 50 malware images is plotted using Matplotlib to visualize structural variations. A bar chart is generated to analyze class distribution, highlighting any imbalance across malware families. python

```
    plt.figure(figsize=(10,6))
    plt.xticks(rotation='vertical')
    plt.bar(classes,(sum(labels)/
    labels.shape[0])*100)
    plt.show()
```

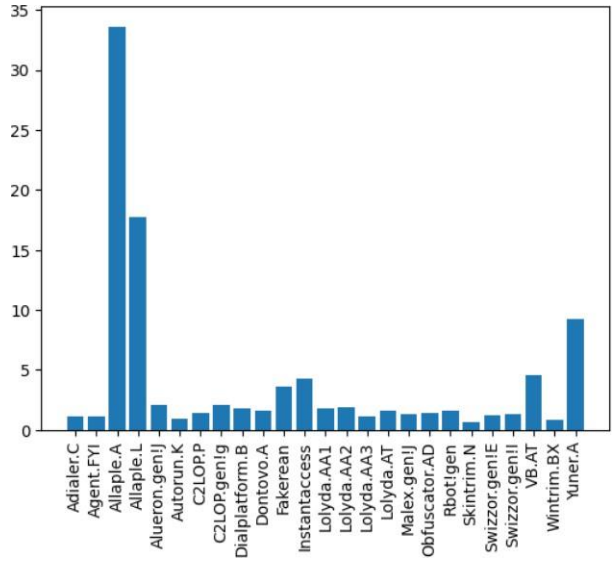

Fig. 2. Dataset visualization.

IV. Methodology

The methodology employed for malware detection in this study is grounded in leveraging deep learning and image classification techniques. This approach relies on the CoAt-Net architecture, a hybrid neural network that combines the strengths of Convolutional Neural Networks (CNNs) and Self-Attention Transformers. The fundamental concept centers on representing malware binaries as grayscale images, enabling the classification task to be reframed as an image recognition problem. This method circumvents the need for disassembly or dynamic analysis, thereby providing a more efficient and scalable solution for malware detection.The process begins with data acquisition and preprocessing.

The MalImg dataset, which comprises grayscale images of malware binaries from 25 different families, forms the basis of this study. These images are resized to 64×64 pixels to ensure uniformity in processing. Labels are one-hot encoded

to facilitate multi-class classification, allowing the system to distinguish malware into its respective families effectively. This preprocessing step ensures that the dataset is optimized for input into the CoAtNet model, which is the centerpiece of the detection architecture.

The CoAtNet model capitalizes on its unique architecture that integrates depthwise convolution layers and self-attention layers. Depthwise convolution layers are instrumental in capturing local spatial relationships within the malware patterns, while self-attention layers enable the model to identify global dependencies critical for accurate classification. The hybrid architecture of CoAtNet stacks these convolutional and transformer layers in a structured manner, leveraging their respective strengths to achieve improved generalization and classification efficiency.

During model training, several optimization techniques are employed. The AdamW optimizer, with a learning rate of 1e-3 and weight decay of 0.0001, is utilized to minimize overfitting. The Categorical Cross-Entropy Loss function, augmented with label smoothing of 0.1, further enhances generalization. Training occurs over 40 epochs with a batch size of 128, and the dataset is split into 90% for training and 10% for validation. Additionally, learning rate adjustments are implemented through the ReduceLROnPlateau callback, which dynamically reduces the learning rate if validation accuracy stagnates for four consecutive epochs. Early stopping with a patience of eight epochs is also applied to avoid unnecessary training and overfitting.

Evaluation of the trained model is conducted using a separate validation set. Metrics such as accuracy, top-5 accuracy, precision, recall, and F1-score are used to assess performance comprehensively. A confusion matrix is generated to visualize misclassifications among the malware families, providing deeper insights into the model's strengths and weaknesses.

The methodology underscores the robustness and scalability of the CoAtNet-based malware detection framework, which significantly outperforms traditional approaches that rely on static or dynamic analysis.

The primary model utilized in this study is CoAtNet0, a compact yet powerful variant of the CoAtNet family tailored for image classification tasks. CoAtNet0's hybrid architecture effectively combines the feature extraction capabilities of CNNs with the global attention mechanisms of transformers. This combination results in superior generalization and classification performance compared to conventional deep learning models.

The architecture of CoAtNet0 includes depthwise convolution layers for local feature extraction and self-attention layers for global context modeling. By vertically stacking these layers in a principled manner, the model achieves an optimal balance between capacity and efficiency. The input shape of the model is set to 64×64×3, corresponding to the normalized grayscale images of malware binaries. A softmax activation function is employed for the final classification layer, ensuring probabilistic outputs for multi-class classification.

The evaluation metrics used include accuracy, top-5 accuracy, precision, recall, and F1-score. Additionally, a confusion matrix is generated to analyze misclassifications across malware families, offering valuable insights into the model's classification behavior. These metrics highlight the efficacy of CoAtNet0 in achieving high classification accuracy and robust generalization.

A. Malware Detection Using CNNs

Malware detection using Convolutional Neural Networks (CNNs) has proven to be an effective approach, particularly when malware binaries are transformed into image representations. CNNs excel at extracting spatial dependencies and hierarchical features, which are crucial for identifying patterns and structures in malware images. The architecture of CNNs typically includes convolutional layers for feature extraction, pooling layers for dimensionality reduction, and fully connected layers for final classification.

In the context of malware detection, CNNs analyze the grayscale images of malware binaries to discern subtle differences between malware families. These networks leverage their ability to learn complex patterns from data, enabling them to classify malware with high accuracy. However, while CNNs are highly effective at capturing local features, they often struggle with modeling long-range dependencies, which can limit their performance in more complex classification tasks.

B. Other Implementations

In addition to CNNs, several alternative deep learning architectures have been explored for malware detection. Vision Transformers (ViTs), for example, have gained attention for their ability to model global relationships within data. However, ViTs require large-scale datasets for effective training and often lack the local inductive biases inherent to CNNs, making them less efficient for smaller datasets like MalImg.

Hybrid transformer-CNN models, such as Swin Transformers and EfficientNet variations, have emerged as promising alternatives. These models integrate CNNs' local feature extraction capabilities with transformers' global attention mechanisms, offering a balanced approach to malware classification. They have demonstrated superior performance in fine-grained classification tasks, making them suitable for malware detection in diverse datasets.

Another avenue of exploration is Graph Neural Networks (GNNs), which model relationships between malware families to uncover hidden patterns in malware evolution. While GNNs offer unique insights into malware relationships, they require structured datasets with relational data, limiting their applicability in image-based malware classification tasks.

In conclusion, the CoAtNet-based malware detection system stands out as a robust and scalable framework that effectively combines the strengths of CNNs and transformers. Its ability to achieve high classification accuracy while maintaining computational efficiency underscores its potential as a state-of-the-art solution for malware detection. Future research could focus on

enhancing the robustness of transformer-based models or integrating adversarial defense mechanisms to counteract evasion techniques employed by sophisticated malware variants.

V. Results

The training process demonstrated a significant improvement in model performance over the epochs. Initially, during the first epoch, the model exhibited a relatively high loss of 2.2432, with an accuracy of 61.91%. However, rapid improvements were observed in subsequent epochs, with accuracy surpassing 90% as early as the second epoch. By the third epoch, the model achieved an accuracy of 96.86% and continued to improve steadily.

Validation performance closely mirrored the training performance, suggesting that the model generalized well to unseen data. The validation loss consistently decreased, reaching a stable point by epoch 13. Notably, the top-5 accuracy quickly approached 100%, indicating that even in cases where the primary prediction was incorrect, the correct class was nearly always among the top five predictions. This result highlights the robustness of the model in classifying malware accurately.

A notable aspect of training was the adjustment of the learning rate. At epoch 9, the ReduceLROnPlateau function reduced the learning rate from 0.0010 to 0.0004 due to a stagnation in validation loss improvement. A further reduction to 0.00016 occurred at epoch 13, demonstrating the model's stabilization and the necessity for fine-grained updates.

TABLE I
Classification Table

	class label	precision	recall	f1-score	support
0	Adialer.C	1.000	1.000	1.000	35
1	Agent.FYI	1.000	1.000	1.000	27
2	Allaple.A	0.998	1.000	0.999	857
3	Allaple.L	1.000	1.000	1.000	440
4	Alueron.gen!J	1.000	1.000	1.000	50
5	Autorun.K	1.000	1.000	1.000	29
6	C2LOP.P	0.800	0.933	0.862	30
7	C2LOP.gen!g	0.902	0.948	0.924	58
8	Dialplatform.B	1.000	0.955	0.977	44
9	Dontovo.A	1.000	1.000	1.000	48
10	Fakerean	1.000	0.989	0.994	91
11	Instantaccess	1.000	1.000	1.000	107
12	Lolyda.AA1	1.000	0.981	0.991	54
13	Lolyda.AA2	1.000	1.000	1.000	50
14	Lolyda.AA3	1.000	1.000	1.000	25
15	Lolyda.AT	1.000	0.970	0.985	33
16	Malex.gen!J	0.972	0.972	0.972	36
17	Obfuscator.AD	1.000	1.000	1.000	39
18	Rbot!gen	1.000	1.000	1.000	32
19	Skintrim.N	1.000	1.000	1.000	16
20	Swizzor.gen!E	0.576	0.919	0.708	37
21	Swizzor.gen!I	1.000	0.032	0.062	31
22	VB.AT	0.975	1.000	0.987	115
23	Wintrim.BX	1.000	1.000	1.000	16
24	Yuner.A	1.000	1.000	1.000	222

VI. Discussion

The high classification accuracy observed in both training and validation suggests that the deep learning model was highly effective at distinguishing between malware types. The final validation accuracy of 98.26% demonstrates that the model achieved near-perfect classification. However, some malware classes exhibited slightly lower precision and recall scores, indicating potential misclassification in certain instances.

For example, the class **Swizzor.gen!I** demonstrated a precision of 1.000 but a recall of only 0.032, leading to a very low F1-score. This indicates that the model struggled to correctly identify instances of this malware family, potentially due to a small sample size or high intra-class variance. Similarly, **Swizzor.gen!E** had a lower precision of 0.576 and recall of 0.919, suggesting a higher rate of false positives.

Most other classes, however, exhibited precision and recall values close to 1.000, demonstrating strong classification performance across the dataset. The macro-averaged precision (0.9689) and recall (0.9479) indicate that, overall, the model maintains a balanced performance across different malware families.

The reduction in learning rate as the model stabilized highlights the effectiveness of adaptive learning rate strategies in deep learning training. By reducing the step size, the model avoided overshooting optimal weight updates, allowing for fine-tuned improvements in later epochs.]

VII. Conclusion

This deep learning-based malware classification model achieved high levels of accuracy, with validation accuracy stabilizing around 98.26%. The model effectively differentiated between various malware families, with most classes achieving near-perfect classification scores. However, certain malware classes with lower precision and recall scores indicate areas where further improvements may be required. These could involve data augmentation, incorporating additional feature engineering techniques, or exploring different neural network architectures. Future work could focus on improving model generalization for underrepresented malware families and reducing misclassification rates. Additionally, testing the model on unseen real-world malware samples would provide further validation of its robustness and practical usability.

References

[1] Nataraj, Lakshmanan, et al. "Malware images: visualization and automatic classification." Proceedings of the 8th international symposium on visualization for cyber security. 2011.

[2] Dai, Zihang, et al. "Coatnet: Marrying convolution and attention for all data sizes." Advances in neural information processing systems 34 (2021): 3965-3977.

[3] Alzahrani, Abdullah IA, et al. "Detecting the presence of malware and identifying the type of cyber attack using deep learning and VGG-16 techniques." Electronics 11.22 (2022): 3665.

[4] Bruzzese, Roberto. "Building visual malware dataset using virusshare data and comparing machine learning baseline model to coatnet for malware classification." Proceedings of the 2024 16th International Conference on Machine Learning and Computing. 2024.

[5] Huang, Hong, et al. "A malicious code detection method based on stacked depthwise separable convolutions and attention mechanism." Sensors 23.16 (2023): 7084.