Journal of Cyber Security

Tech Science Press

DOI: 10.32604/jcs.2024.053954

ARTICLE

Data-Efficient Image Transformers for Robust Malware Family Classification

Boadu Nkrumah[1,*], Michal Asante[1], Gaddafi Adbdul-Salam[1] and Wofa K. Adu-Gyamfi[2]

[1]Department of Computer Science, Kwame Nkrumah University of Science and Technology, Kumasi, AK-315-1475, Ghana

[2]Department of Mathematics, Kwame Nkrumah University of Science and Technology, Kumasi, AK-315-1475, Ghana

*Corresponding Author: Boadu Nkrumah. Email: bnkrumah7@st.knust.edu.gh

Received: 14 May 2024 Accepted: 21 August 2024 Published: 17 December 2024

ABSTRACT

The changing nature of malware poses a cybersecurity threat, resulting in significant financial losses each year. However, traditional antivirus tools for detecting malware based on signatures are ineffective against disguised variations as they have low levels of accuracy. This study introduces Data Efficient Image Transformer-Malware Classifier (DeiT-MC), a system for classifying malware that utilizes Data-Efficient Image Transformers. DeiT-MC treats malware samples as visual data and integrates a newly developed Hybrid GridBay Optimizer (HGBO) for hyperparameter optimization and better model performance under varying malware scenarios. With HGBO, DeiT-MC outperforms the state-of-the-art techniques with a strong accuracy rate of 94% on the MaleViS and 92% on MalNet-Image Tiny datasets. Therefore, this work presents DeiT-MC as a promising and robust solution for classifying malware families using image analysis techniques and visualization approaches.

KEYWORDS

Malware classification; machine learning; deep learning; DeiT; vision transformers; MaleVis dataset; malnet-image tiny dataset; visualization techniques; transfer learning

1 Introduction

With the increasing complexity of malware, including code encryption and packers techniques, automated detection systems are in demand. The number outlines the seriousness: 5.5 billion attacks are reported in SonicWall's 2023 Cyber Threat Report [1]. These new threats evade traditional methods and pose significant challenges for security professionals reliant on time-intensive reverse engineering practices [2]. Aside from this, the very modern communication infrastructure allows malware's evolution to grow faster and adapt quickly. However, current solutions face challenges in detecting closely related malware families, identifying obfuscated malware, solving class imbalances, and dealing with highly resource-intensive disassembly processes [3]. With the cost of cybercrime costing a predicted $10.5 trillion by 2025 [4], investment in research and collaboration is needed to develop more effective detection techniques. Classifying malware into families allows computer security experts to determine common traits that improve attribution, mitigation, and understanding of these threats.

Traditional methods for malware detection, such as signature-based and heuristic analysis, are easily bypassed by the changing threat landscape. New, sophisticated malware versions can easily bypass signature-based detection, while heuristics must be tuned to fight emerging threats [5]. While machine learning approaches are explored for better detection, time-consuming feature selection processes often hinder them. Recent developments concern visualization-based approaches in which detection automation is done by converting malware samples into images. Despite the computational and scalability challenges usually occurring in visualization techniques, this approach has shown substantial promise. For example, Seneviratne et al. successfully used grayscale and RGB representations in transformer-based detection, showing the potential of such methods [6].

Deep learning models have tremendously contributed to the domain of malware detection, and among these models, Convolutional Neural Networks (CNNs) efficiently process huge datasets. However, it is expensive computationally to train CNNs from scratch with large datasets [7]. As a remedy, transfer learning has emerged as a strong solution, capitalizing on pre-trained models on enormous image datasets. This massively saves the time required for training and makes the model generalize to new malware variants. Recent studies by Van Dao et al. show how transfer learning strengthens the generalizing ability of CNNs concerning new and unseen malware variants for enhancing accuracy and robustness in detection [8].

Transformer models, which have been successful in Natural Language Processing (NLP), are being applied as Vision Transformers (ViTs) in computer vision. The architectures of ViTs—the ability to capture long-range dependencies through self-attention—differ from CNNs and have led to state-of-the-art performances in tasks like image classification, semantic segmentation, and video action recognition. Despite the remarkable strides made in various domains, the potential of DeiT in malware detection remains largely untapped. Our study proposes the Data-Efficient Image Transformer Malware Classifier (DeiT-MC) to close the gap. This new method effectively leverages DeiT's capabilities to achieve efficient and accurate malware family classification. DeiT-MC eliminates labor-intensive disassembly processes, a common burden of traditional malware analysis. Most importantly, it incorporates a hybrid GridBay optimizer that mitigates common obstacles in malware detection, such as class imbalance, obfuscation techniques, and variant diversity, to increase the robustness of the malware classification. The key contributions of this study are as follows:

1. It introduces a new approach that combines image-based visualization with ViT techniques, specifically DeiT-MC, for classifying malware variants, enjoying the benefits of transfer learning to improve classification accuracy.

2. It demonstrates multiclass classification implemented on imbalanced datasets of diverse nature, including two well-established datasets of malware, showing the high efficiency and resilience to advanced malware evolution and anti-malware evasion tactics.

3. It presents a rapid and efficient deep learning-based malware classification system that operates directly on raw binary images, eliminating the need for reverse engineering, code disassembly, or binary execution. The proposed method significantly reduces feature spaces, simplifying the classification process.

4. DeiT-MC presents a new hybrid GridBay Optimizer, seamlessly integrated into fine-tuning. This unique optimization technique enhances model performance, rendering DeiT-MC more versatile and resilient in addressing a variety of malware scenarios. The novelty lies in applying this hybrid optimization to the DeiT model, adapted for malware classification–a first in this domain.

The paper's structure includes Section 2, which examines relevant works; Section 3, which explains the proposed DeiT-based malware detection system; and Section 4, which presents experimental results and comparisons, followed by performance analysis and extensive discussions. In Section 5, the study concludes the research and Section 6 discusses future works.

2 Related Work

Researchers have applied transformer-based models in natural language processing to Android malware detection. A study by Oak et al. [9] introduced a Bert-based model that achieved impressive results, with 96% accuracy on highly imbalanced datasets. However, challenges remain in improving the interpretability of the model and ensuring broader generalizability. Additionally, relying solely on assembly code for feature extraction has limitations, such as limited semantic understanding and susceptibility to obfuscation techniques and code structure changes.

In a similar work, Hu et al. [10] presented an alternative hierarchical transformer model using assembly code to classify the IoT malware family. The authors trained their model using IoT software samples, including the Mirai, Gafgyt, and benign samples. The model achieved a 94.67% family classification accuracy rate. However, the utilization of obfuscation methods on IoT malware is rare due to the constrained computational resources of IoT devices. Despite the authors successfully unpacking the malware and obtaining favorable results, the model could remain susceptible to code obfuscation.

In contrast, Lakshmanna et al. [11] proposed a lightweight energy consumption ensemble-based botnet detection model for IoT/6G networks. Their model leverages Machine Learning (ML) techniques within an ensemble-based detection framework to identify and safeguard IoT network infrastructure from botnet attacks. Zhou et al.'s approach emphasizes energy efficiency and sustainability, achieving a remarkable 100% accuracy rate in detecting malicious botnets within the network.

Building upon the ongoing research, Li et al. [12] introduced the Interpretable Malware Detector (I-MAD), a transformer-based framework for malware detection using assembly code. The I-MAD framework features a Galaxy Transformer network component that enhances performance by interpreting assembly code at multiple levels. The model achieves a high detection accuracy of 97.7% and provides interpretations for its results, which can benefit malware analysts. However, the interpretability of the model poses challenges in white-box scenarios, where attackers could exploit the interpretations to create adversarial samples more easily than with uninterpretable models.

Seneviratne et al. [6] presented SHERLOCK, a transformer-based computer vision model, as a self-supervised learning strategy for Android malware categorization. They tested the model on a large dataset of 1.2 million Android apps containing 47 malware-type families. According to the test results, the model has an accuracy of 97%. While innovative, its reliance on synthetic imagery poses resource-intensive challenges in computational requirements and time investment for data generation and model training.

Deng et al. [13] introduced TransMalDE, a new hierarchical architecture for detecting IoT malware. The proposed transformer-based detection model captures developing malware patterns, and their strategy entails relocating computation-intensive jobs. Results from experiments show that TransMalDE performs better than current state-of-the-art systems on a variety of benchmark datasets. While TransMalDE showcases significant advancements in efficiency and accuracy, the study does not explicitly address potential security and privacy concerns related to migrating computation to edge nodes.

2.1 Transformer-Based Limitations & Discussions

Vision Transformers have since revolutionized computer vision, yet their quadratic computational and memory complexity concerning image resolution stands in critical opposition to the needed scalability. These complex computational burdens arise from the large number of parameters needing to be trained and the increased computation due to the pairwise attention mechanism. Researchers are investigating various methods to scale up to larger datasets and higher-resolution images, such as sharing parameters, pruning, and knowledge distillation [14]. However, the vulnerability of the vision transformers to adversarial attacks raises questions about the reliability of the malware detection system. For example, Jakhotiya et al. [15] found a transformer-based malware detector vulnerable, with 25% misclassified due to adversarial attacks, which points to the need to address such vulnerabilities for system robustness. Another problem is existing studies, such as those conducted by Oak et al. [9], Hu et al. [10], and Li et al. [12], mostly depend on disassembling malware binaries, which is time-consuming and complex. Moreover, there is a shortage of vision transformers in literature beyond the Windows and Android malware, leaving gaping voids for Mac and Linux. This gap in research specifically concerns critical systems, such as avionics, where robust security is paramount.

2.2 Recent Developments in Hyper-Parameter Optimization

Recent advances in hyper-parameter optimization (HPO) techniques have resulted in impressive performance improvements in machine learning (ML) models. Random and Grid searches are the most common, while Bayesian Optimization lately has shown very promising performance. Each technique offers strengths and limitations, as several studies have shown.

Grid search is a method of performing an exhaustive exploration based on a predefined set of hyper-parameters. Mantovani et al. [16] applied Grid Search to decision trees run over 102 datasets and observed better performance than the model's default settings. However, Grid Search can be very computationally expensive and time-consuming since it is an exhaustive method, which becomes increasingly so with growing hyper-parameters.

It is empirically shown that Random Search outperforms Grid Search in tuning neural networks, especially for high-dimensional spaces. Bergstra et al. [17] performed experiments that showed that random search efficiently finds better configurations of hyper-parameters with fewer iterations than grid search. This efficiency is very important in working with complex models and large hyper-parameter spaces.

Optimization of hyper-parameters using Bayesian Optimization has emerged as a strong alternative using probabilistic models. For example, Algorain et al. [18] demonstrated the applicability of finely tuning hyper-parameters to tasks such as malware detection and showed significant accuracy improvements over the default settings. Bayesian Optimization focuses its search on promising regions of the space of the hyperparameters and is particularly suitable when solving exploration-exploitation trade-offs.

The uniqueness of our work is in distinctly integrating Bayesian optimization with Grid search, particularly tailored for the DeiT model in malware classification; that is, a hybrid method we refer to as the Hybrid GridBay Optimizer (HGBO) that can harness their strengths—Bayesian optimization's global search efficiency and the breadth of grid search in the quest through hyper-parameter configurations.

3 Proposed Methodology

Our methodology strongly emphasizes the optimization strategies for fine-tuning machine learning models in the performance evaluation. Hyperparameter optimization (HPO) of deep learning models, including DeiT, requires careful tuning for the best performance. HPO represents a major challenge because of the huge size of the intricate search space of potential parameter values [19]. More specifically, DeiT-based architectures, with their attention-based aggregation and distillation techniques, require many hyperparameters: patch size, number and dimension of attention heads, dropout rate, learning rate, and distillation temperature, among others. While the traditional methods used for this purpose, like Grid Search, are efficient, they could be computationally expensive over large spaces of parameters. On the other hand, Bayesian optimization offers an exploratory approach, but certain scenarios might be limited [19]. To address these issues, we present the **Hybrid GridBay Optimizer (HGBO),** which combines the merits of these two approaches. In our proposed approach, HGBO combines Grid Search's structured search capabilities and the probabilistic modeling aspect of Bayesian Optimization to balance them. The final effect is a multifunctional and efficient way of tweaking hyperparameters by optimizing machine learning models. Algorithm 1 summarizes the approach.

Algorithm 1: Hybrid GridBay optimizer (HGBO)

Input: Hyperparameter space H, initial grid G, Bayesian Optimization B

Output: Optimal hyperparameters o_{ptimal}

1. **Grid Search Phase:**
   - Perform an exhaustive search over the grid: G $\subset$ H
   - Evaluate the model performance P or each set of hyperparameters: $P(\theta g)$ for $\theta g \in G$.
2. **Update Bayesian Method:**
   - Update the Bayesian Optimization model using the results from Grid Search: $BayesianModel \leftarrow$ UpdateBayesianModel(H, G, P).
3. **Bayesian Optimization Phase:**
   - Select the next set of hyperparameters based on the probabilistic predictions from the Bayesian model: $\theta_{next} \leftarrow SelectNextHyperparameters(BayesianModel)$
   - Evaluate the model performance for the selected hyperparameters: $P(\theta_{next})$
4. **Iterative Process:**
   - Iterate steps 3 and 4 until *convergence*: H $\leftarrow$ *IterativeUpdate* (H, *BayesianModel*) .
5. **Convergence Criteria:**
   - Define convergence criteria, such as maximum iteration count or a threshold for enhancement: *ConvergenceCriteria* (H) .
6. **Optimal Hyperparameters:**
   - Identify the set of hyperparameters that yielded the best model performance during the optimization process: $\theta_{optimal} = argmin\theta_g f\left(\theta_g\right)$.

The proposed DeiT-MC framework covers data preparation, model training, and model evaluation. Fig. 1 illustrates the framework. The following sections present the three phases in detail.

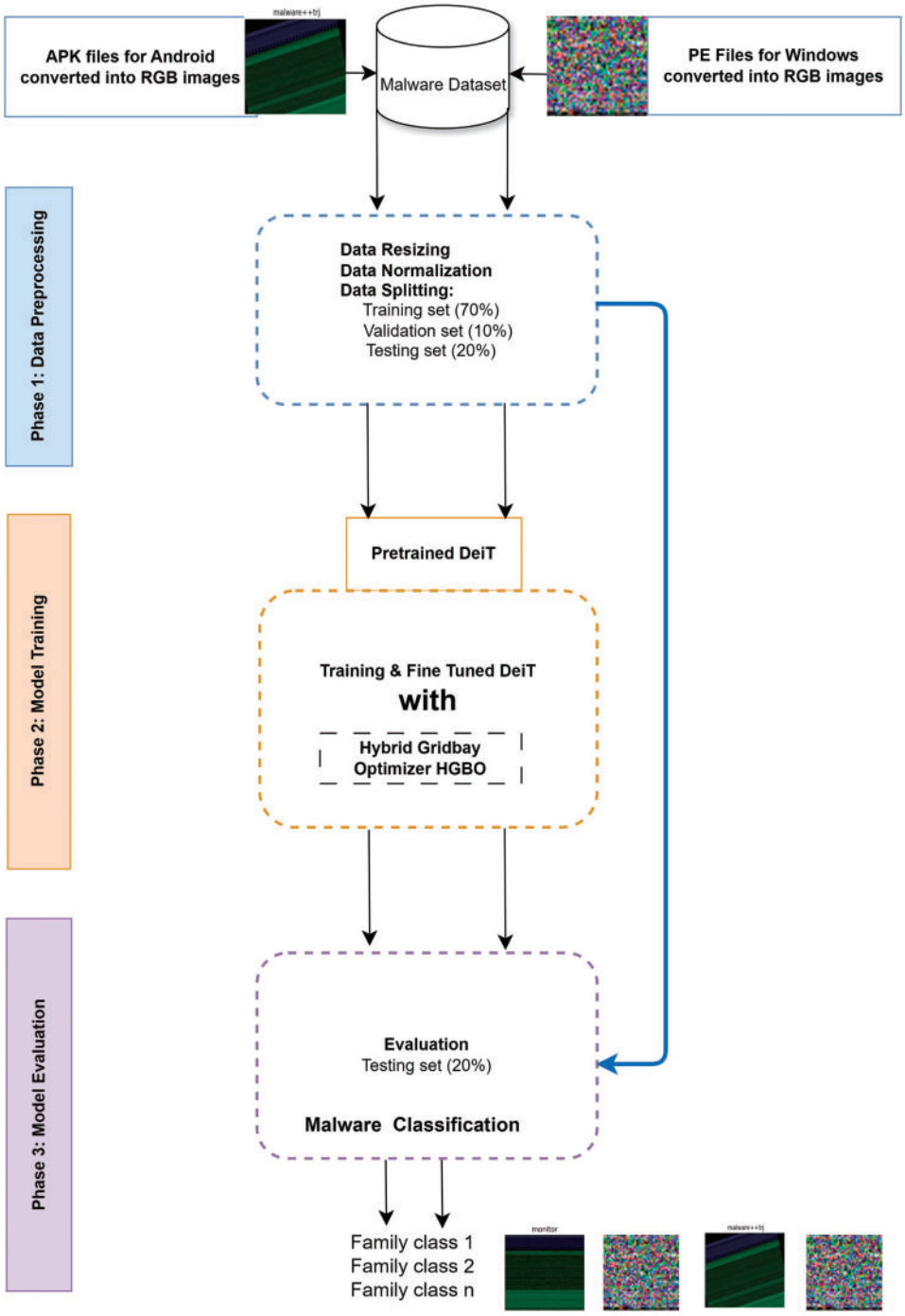

Figure 1: The DeiT-MC framework

Data Preparation: The dataset used in this study consists of Windows PE and Android Dex files converted into RGB images with a resolution of 224 × 224 pixels, standardizing them for compatibility with the DeiT model. This standardized image size is a common practice in image-based machine learning, validated for capturing essential features. However, we acknowledge that the fixed size of

224 × 224 pixels, based on the first 50 KB of the files, may introduce biases due to the header information, potentially causing overfitting. Furthermore, modern malware can be significantly larger, sometimes reaching hundreds of MBs, making using only the first 50 KB of such malware files potentially erroneous. To mitigate these limitations, we utilized the entire malware file content to minimize overfitting to potentially biased file header information and ensure a comprehensive representation of malware characteristics. Despite this, our study has not explored the effect of scaling down images from larger files (a form of lossy compression). This is a crucial consideration for future research to understand how image scaling impacts the model's performance and implications for real-world malware classification.

Unlike techniques based on the analysis of assembly code, our method works directly on raw byte sequences. While raw byte sequences can also be obfuscated with some techniques, DeiT-MC uses strong data augmentation techniques such as random byte flipping and shuffling to make it resilient. Aside from this, model regularization techniques like dropout prevent overfitting and enhance the model's ability to generalize against unknown obfuscation patterns. Finally, DeiT-MC uses transformers to model long-range dependencies within raw byte sequences, which helps identify further patterns obfuscated by structural changes.

Normalization techniques were applied to the pre-converted images to reduce inherent variations and enhance generalization. Data augmentation strategies, including random rotations, flips, and color adjustments, were employed to enrich the dataset's diversity and robustness, aiding the model in better generalization across various training scenarios. Recognizing potential biases in the pre-converted dataset, advanced data-cleaning techniques and thorough labeling reviews were utilized to ensure that the dataset accurately represents the diversity of malware types and minimizes any skewness in class distribution. The preprocessed data was divided into stratified training, validation, and testing sets in a ratio of 70%, 10%, and 20%, respectively. This ensures a balanced distribution of samples across subsets, providing an unbiased evaluation framework. This comprehensive data preparation pipeline lays the foundation for strong and reliable performance in the subsequent stages of analysis and modeling.

1. **Model Training:** Training begins with loading a pre-existing, DeiT-based model and then adjusts the final layers to fit the specific number of malware classes within the dataset. The Hybrid GridBay Optimizer is integrated and used to improve this process of fine-tuning further. HGBO integrates grid search with Bayesian optimization methods into one solution that dynamically adapts learning rates and effectively travels through weight spaces to optimize a model's training efficiencies and adaptability. A hyperparameter space consists of important parameters such as learning rate and momentum, and tuning them fine-tunes the performance of a model. Leveraging a combination of grid search and Bayesian optimization aided by HGBO, we attain a balance between exhaustive exploration and efficient navigation of the hyperparameter space. This iterative optimization strategy is critical in dramatically improving DeiT-MC's performance over classifying malware instances. The efficient control of the hyperparameter search process by HGBO over 100 iterations includes a 30% allocation for grid search and 70% for Bayesian optimization. Notably, the defined grid search parameters encompass learning rates from 1e-5 to 1e-3 and momenta values from 0.9 to 0.99. We also implement a weight decay of 0.03 to avoid overfitting and leverage class weights for variable-sized samples. Class weights and data augmentation are employed to mitigate the effect of class imbalance, making DeiT-MC more resilient and improving its performance across diverse scenarios. The class weights, being inversely proportional to the class frequencies, guided the

model to look more at the underrepresented classes as dictated by the formula:

$$w_c = \frac{N}{num\_classes \times class\_frequency_c} \tag{1}$$

where w_c represents the weight for class c, N is the total number of samples, and *class_frequency* denotes the frequency of class c in the dataset.

The model is initialized with the weights from ImageNet; a learning rate scheduler with a reduction factor of 0.1 and patience of 5 epochs is then applied. Early stopping based on the validation F1-score condition prevents the model from further overfitting, reducing the learning rate through gradual decay up to a minimum of 10^{-6}. This kind of training regime—strong but coupled with the power of HGBO—considerably strengthens the model's ability to classify malware. DeiT-MC optimizes feature space dimensionality throughout the training phase by utilizing DeiT. The multi-head self-attention mechanism inside the DeiT transformer blocks emphasizes details in images that matter for a feature set, capturing the essence of the features while lowering the input space dimensions. This approach explains the effectiveness of DeiT-MC in carrying out diverse malware classification, reducing training and inference computations.

2. **Model Evaluation:** The trained DeiT-MC model then makes predictions on the unseen or test data. Its performance in detecting malware samples is evaluated using metrics like accuracy, precision, recall, F1 score, and confusion matrix to get a deep understanding without hampering the integrity of the evaluation.

3.1 DeiT Architecture

DeiT, a variant of transformers developed for efficient computer vision tasks by Touvron and colleagues, adopts a teacher-student strategy. This strategy uses knowledge distillation and tokens to help knowledge flow from a bigger, pre-trained teacher model to a smaller student model based on the attention mechanism. The basic architecture of DeiT-base is shown in Fig. 2. This architecture captures more global dependencies in the images, which are captured using much fewer parameters to arrive at the performance of CNNs. It is an encoder with stacked self-attention layers that build hierarchical feature representations. DeiT divides the images into 16×16 patches and sends them into an embedding layer to obtain the patch embedding, augmented by class and distillation tokens. The position embeddings (PE) are added to all the tokens. After adding, the sequence is fed into a series of encoders, wherein the self-attention mechanism follows each encoder and then the feed-forward neural network layer. The attention mechanism of DeiT uses dot product similarity normalized with the softmax function to compute the attention score. DeiT has residual links in the encoder layers and uses Multi-Layer Perceptron (MLP) blocks to enhance and boost the model's capacity. Our study uses the DeiT-base architecture with about 85 M parameters and 12 transformer encoder layers, where each transformer encoder layer contains 12 attention heads, amounting to a total of 144 attention heads.

3.2 Experimental Setup

We exploited Google's computing power using Google Colab Pro+ to run our experiments. Our study is hugely based on the powerful NVIDIA A100-SXM4 GPU, with 89 GB of RAM offered by the Colab platform. We attached our datasets to the Google Colab environment using Google Drive to allow easy handling. We implemented the pre-trained DeiT using the Python package TIMM and used the PyTorch library, an open-source deep learning framework, in Python 3.8.

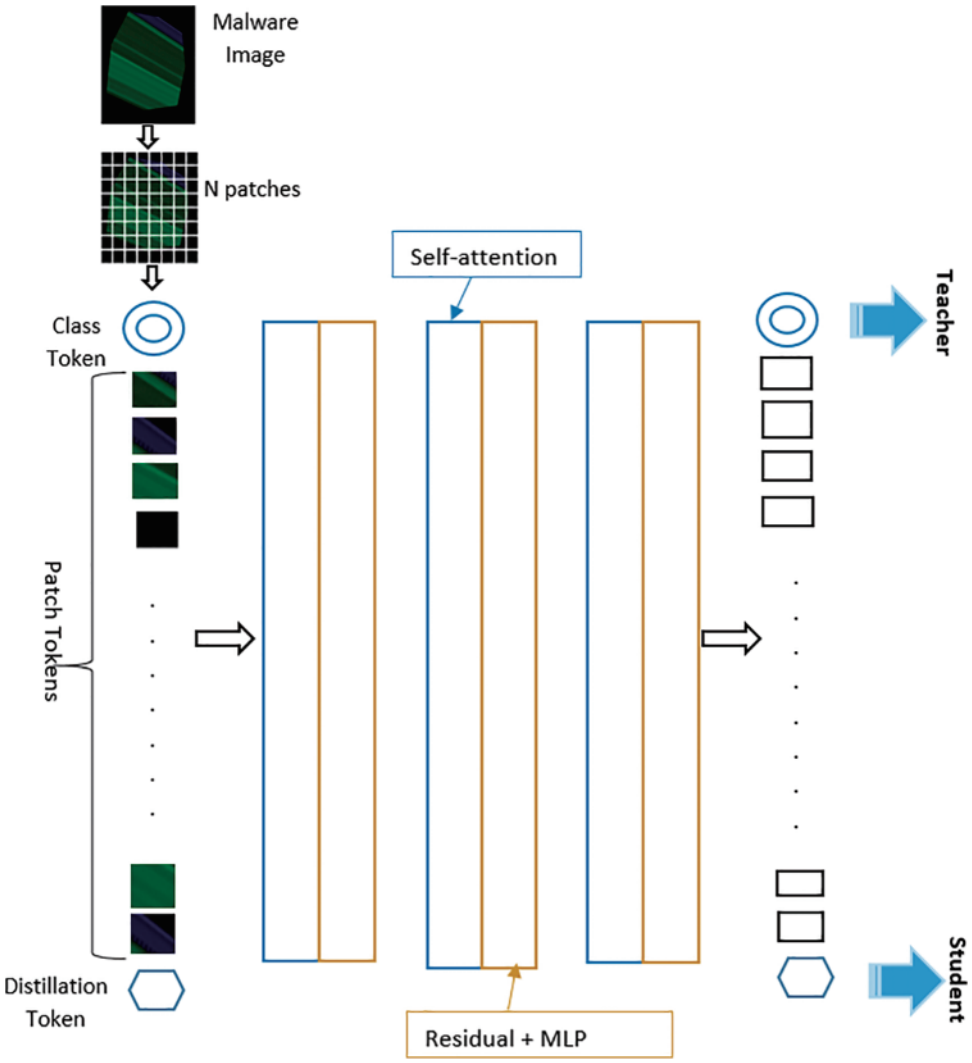

Figure 2: DeiT architecture

3.3 Datasets

The MaleVis dataset [20] introduced in 2019 and presented in Fig. 3, was provided by **C**omodo Inc. It contains 14,226 RGB images in both resolutions: 224 × 224 and 300 × 300 pixels. Further, it is divided into 26 categories, including one for benign class and 25 for malware types. We use images with a resolution of 224 × 224 pixels. It shows a balanced class distribution for malware-type images; it contains around 500 images per class. The 'Normal' class has 1832 samples, creating an imbalanced dataset. We have divided the dataset into 70% training, 10% validation, and 20% testing subsets while keeping the class ratios the same. This careful partitioning strategy retains the representational distributions of malware types and benign samples in subsets to provide unbiased model evaluation.

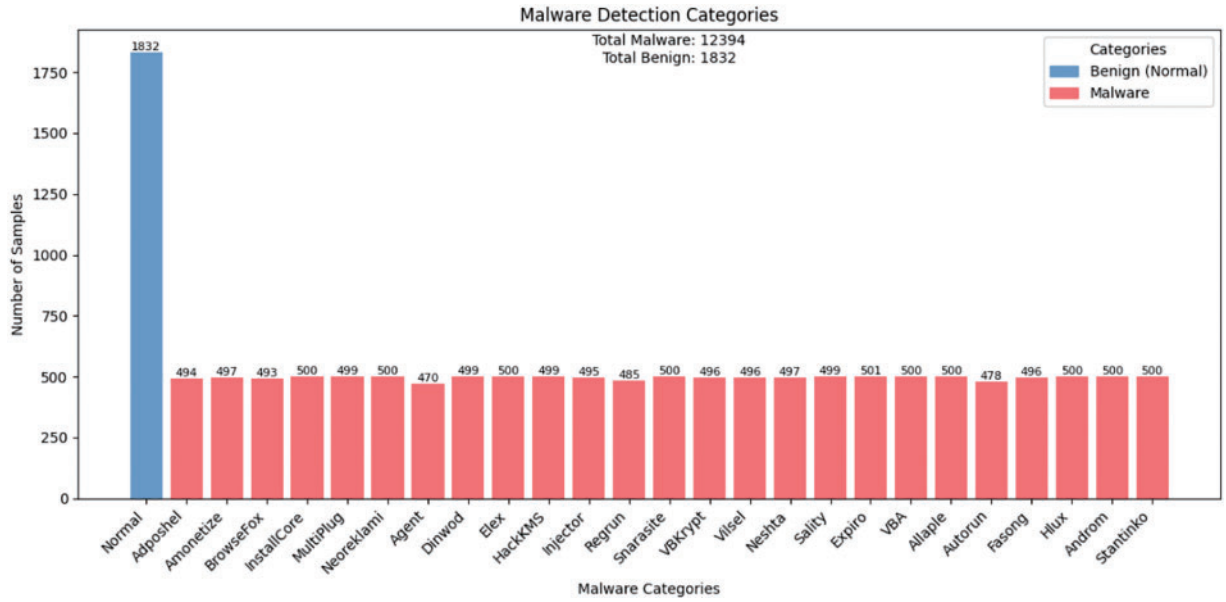

Figure 3: Distribution of malware samples on MaleVis dataset

The **MalNet-Image Tiny dataset** [21] was introduced in 2022. It comprises 61,201 training images, 8,742 validation images, and 17,486 test images, categorized into 43 malware types. The dataset is constructed from real-world malware samples sourced from VirusTotal, ensuring it reflects actual conditions. It includes many malware families, providing a comprehensive basis for evaluating malware classification techniques. Maintaining a stratified ratio of 70-10-20 for training, validation, and testing, this distribution ensures robust model evaluation. We selected 15 malware types for our experiments, considering resource constraints such as disk space, GPU capacity, and memory availability. As observed in Fig. 4, the dataset exhibits a high degree of class imbalance. Fig. 3 displays samples of malware families from the MaleVis dataset, whereas Fig. 4 displays malware samples from the highly imbalanced MalNet-Image Tiny dataset.

4 Results and Discussion

4.1 Evaluation Metrics

The metrics for evaluating our model include Accuracy, Precision, Recall, and F1-score. Accuracy is the general metric that calculates overall correct classification. Precision is a measure of the ability of the model to identify true positives. Recall focuses on correctly identifying true positives, and the F1-score is the balanced measure of the two. Eqs. (2)–(5) give these measures.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

$$Recall = \frac{TP}{TP + FN} \tag{4}$$

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{5}$$

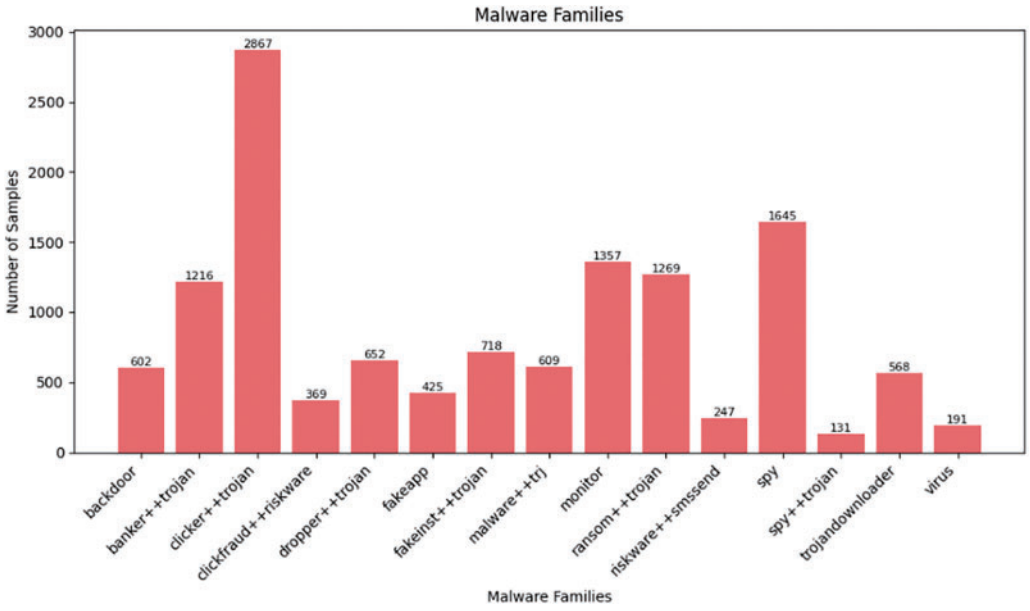

Figure 4: Distribution of malware samples on MalNet-Image tiny dataset

Additionally, we used weighted averages, micro averages, and the confusion matrix to evaluate the model's prediction comprehensively. Weighted averages are used to adjust the class imbalance property of our dataset; that is, weights for classes with larger sample sizes or more significance are higher when calculating metrics. The micro-averages objectively measure the model's performance, considering all classes as one. The confusion matrix elaborates on the count of true positives, true negatives, false positives, and false negatives, and this can be used to bring out patterns or trends in the way the model does misclassifications. Receiver Operating Characteristic-Area Under the Curve (ROC-AUC) measures a model's capability to differentiate between classes, considering the trade-off between true positives.

4.2 Ablation Study

This section evaluates how different hyperparameter optimization methods impact the DeiT model using the MaleViS and MalNet-Image Tiny datasets. The methods assessed include the Baseline DeiT Model, Grid Search, Bayesian Optimization, Random Search, and HGBO (combined approach).

4.3 Experimental Setup

Experiments are conducted using the DeiT model and each optimization method across both datasets. The evaluation metrics used include accuracy (Acc), precision (Prec), recall (Rec), F1-score (F1), the number of model parameters (Params), and prediction time (Pred Time). Experiments are conducted using DeiT, and each optimization method is applied across both datasets.

Table 1 outlines the parameters tested, the selected settings, and the rationale behind each choice. Key parameters include learning rate, batch size, number of epochs, optimizer, regularization techniques, and data augmentation strategies.

Table 1: Experimental setup parameters

Parameter	Settings tested	Selected setting	Rationale
Learning rate	1×10^{-5}, 1×10^{-4}, 1×10^{-3}	1×10^{-4}	Best trade-off between convergence speed and stability
Batch size	16, 32, 64	32	Balance between computational efficiency and model performance
Number of epochs	50	50	Sufficient for convergence in preliminary experiments
Optimizer	Adam	Adam	Adaptive learning rate properties improve convergence times
Regularization techniques	Dropout rate: 0.3, 0.5, 0.7	Dropout rate: 0.5	Prevents overfitting and enhances model generalization
Weight decay	0.03	0.03	Regularization to prevent overfitting
Data augmentation	Random cropping, flipping, rotation	Applied	Enhances model generalization capabilities
Class weights	Inversely proportional to class frequencies	Applied	Mitigates the effect of class imbalance
Model architecture	DeiT with minor modifications	Applied	Adapted for malware classification tasks
Baseline DeiT parameters	Default DeiT parameters	Applied	Used as a reference point for performance comparison
Grid search parameters	Learning rates: 1×10^{-5} to 1×10^{-3}; Momenta: 0.9 to 0.99	N/A	Explored to find the optimal configurations
Bayesian optimization parameters	Learning rate: 1×10^{-5} to 1×10^{-3}; Dropout rate: 0.3–0.7	N/A	Fine-tuning to achieve optimal performance
Random search parameters	Randomly selected within predefined ranges	N/A	Ensures diverse exploration of the hyperparameter space

(Continued)

Table 1 (continued)

Parameter	Settings tested	Selected setting	Rationale
Optimisation strategy	30% Grid Search, 70% Bayesian Optimization	Applied	Balanced exploration and exploitation of hyperparameter space

4.4 Results

The experimental results for the MaleVis and MalNet-Image Tiny datasets are presented in Tables 2 and 3, respectively.

Table 2: MaleViS dataset

Method	Acc (%)	Prec (%)	Rec (%)	F1 (%)	Params (M)	Pred time (s)
Baseline DeiT	88.5	88	88.7	88.35	85	17
Grid search	92	91.8	92.2	92	85	16.8
Bayesian optimization	92.5	92.2	92.6	92.4	85	16.5
Random search	92.2	92	92.4	92.2	85	16.7
HGBO (Combined approach)	**94**	**94**	**94**	**94**	**85**	**16**

Table 3: MalNet-Image Tiny dataset

Method	Acc (%)	Prec (%)	Rec (%)	F1 (%)	Params (M)	Pred time (s)
Baseline DeiT	88	87.5	88.2	87.85	85	16.5
Grid search	90	89.7	90.1	89.9	85	16.3
Bayesian optimization	91.5	91.2	91.6	91.4	85	16
Random search	90.5	90.2	90.7	90.45	85	16.2
HGBO (Combined approach)	**92**	**92**	**92**	**92**	**85**	**15.62**

4.5 Key Findings

Our ablation study revealed several critical insights. HGBO, the combined approach, achieved the highest weighted accuracy of 94.00% on the MaleViS dataset and 92.00% on the MalNet-Image Tiny dataset, which represents a significant improvement over the Baseline DeiT model by +5.50% and +4.00%, respectively. Additionally, HGBO consistently reduced the predictive time to 16.00 s for MaleViS and 15.62 s for MalNet-Image Tiny, which outperformed all the other methods in every

scenario. Equally, all methods had 85 million parameters, thus showing that the optimization process targeted the hyperparameters and not the complexity of the model. HGBO performed consistently better than Grid Search, Bayesian Optimization, and Random Search, underpinning why advanced model optimization strategies are necessary to stretch model performance.

It has been shown that hyperparameter optimization exerts a very considerable impact on models trained from scratch. Applying HGBO at the initial stages of training can thus be assumed to create an extended effect. This systematic approach will optimize critical hyperparameters immediately and improve the model's performance and efficiency. While our current study did not involve training from scratch—due to the resource constraint—and focused on transfer learning, this might be explored in future work. Studies such as [18] have demonstrated that hyperparameter optimization greatly impacts models trained from scratch; it is easy to envision that HGBO would bring even larger gains when applied at the early stages of model training. This would enable HGBO to systematically optimize key hyper-parameters initially, further improving both quality and efficiency for models.

4.6 Performance of DeiT-MC Family Malware Classification

As demonstrated in Table 4, DeiT-MC shows strong cross-platform malware classification, achieving weighted average scores of 94% for accuracy, precision, recall, and F1-score on the MaleViS dataset. This trend continues in the MalNet-Image Tiny dataset with strong weighted average scores of 92% across the metrics measured, as shown in Table 5. However, further optimization is needed for some classes, such as 'ransom++trojan,' given the accuracy rate of 0.79. DeiT-MC overcame the imbalanced dataset problem and obfuscated malware without disassembly—a major advantage in the real world where timely analysis is essential. DeiT-MC used class weights and data augmentation to tackle class imbalance in the datasets. This approach enhanced its performance in light of imbalanced data.

Table 4: Classwise performance on MaleViS dataset

Classification report: Family	Precision	Recall	F1-score	Support
Adposhel	1.00	1.00	1.00	74
Agent	0.81	0.93	0.87	71
Allaple	0.97	0.99	0.98	72
Amonetize	1.00	0.96	0.98	74
Androm	0.93	0.92	0.93	75
Autorun	0.92	0.89	0.90	74
BrowserFox	0.97	0.99	0.98	74
Dinwood	0.99	1.00	0.99	75
Elex	0.99	1.00	0.99	74
Expiro	0.86	0.99	0.92	74
Fasong	1.00	1.00	1.00	75
HackKMS	0.99	1.00	0.99	75
Hlux	1.00	1.00	1.00	75
Injector	0.96	0.85	0.90	75
InstallCore	0.99	0.99	0.99	75
MultiPlug	0.93	0.95	0.94	75

(Continued)

Table 4 (continued)

Classification report: Family	Precision	Recall	F1-score	Support
Neoreklami	1.00	0.99	0.99	75
Nesta	0.67	0.55	0.60	74
Other	0.85	0.84	0.84	276
Regrun	0.99	1.00	0.99	73
Sality	0.72	0.74	0.73	75
Snarasite	1.00	1.00	1.00	75
Stantinko	1.00	1.00	1.00	75
VBA	1.00	1.00	1.00	75
VBKrypt	0.97	1.00	0.99	74
Visel	1.00	0.99	0.99	74
Accuracy			**0.94**	**2135**
macro avg	**0.94**	**0.92**	**0.93**	**2135**
weighted avg	**0.94**	**0.94**	**0.94**	**2135**

Table 5: Classwise performance on MalNet-Image Tiny

Classification report: Family	Precision	Recall	F1-score	Support
clicker++trojan	0.72	0.85	0.78	60
malware++trj	0.83	0.87	0.85	221
fakeapp	0.99	0.99	0.99	287
spy	1.00	1.00	1.00	37
virus	0.98	0.95	0.97	119
dropper++trojan	0.95	0.95	0.95	42
trojandownloader	0.87	0.92	0.89	72
clickfraud++riskware	0.87	0.98	0.92	72
fakeinst++trojan	0.97	0.98	0.97	136
riskware++smssend	0.95	0.91	0.93	231
monitor	0.81	0.68	0.74	25
backdoor	0.93	0.93	0.93	164
ransom++trojan	0.71	0.50	0.59	24
banker++trojan	0.92	0.77	0.84	57
spy++trojan	0.75	0.63	0.69	19
accuracy			**0.92**	**1555**
macro avg	**0.88**	**0.86**	**0.87**	**1555**
weighted avg	**0.92**	**0.92**	**0.92**	**1555**

4.7 Performance Assessment through Confusion Matrices

Figs. 5 and 6 show confusion matrices that help analyze the malware classification results of DeiT-MC. The confusion matrices provide a complete overview of how well DeiT-MC has classified instances from each malware family. The diagonal components indicate the correct data points for the predicted label. In contrast, the off-diagonal components indicate the number of data points for which the predicted label is incorrect.

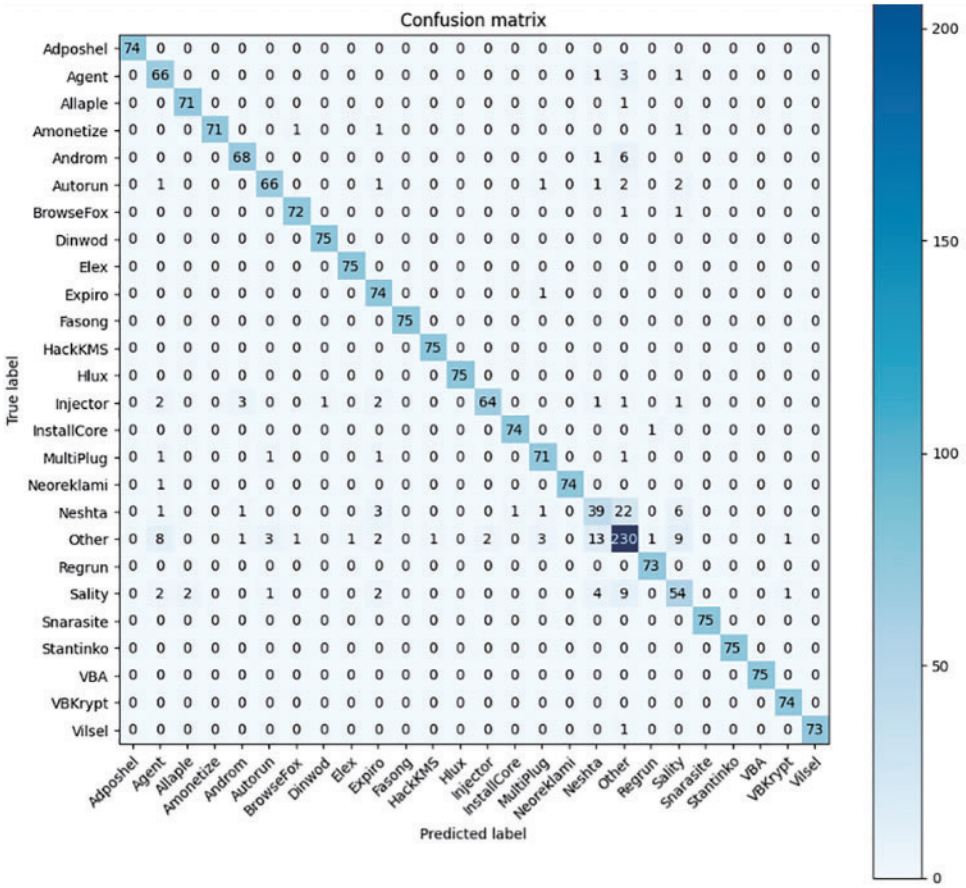

Figure 5: Confusion, matrix of the performance of DeiT-MC on the MaleVis dataset, accurately reflects performance by pinpointing the true class label

4.8 Performance Assessment through ROC Curves

To better understand DeiT-MC's performance, we analyze the Receiver Operating Characteristic (ROC) curve, displayed in Figs. 7 and 8. The ROC curve depicts the trade-off between the true positive rate (TPR) and the false positive rate (FPR) at various classification thresholds. A larger AUC signifies a better classification model.

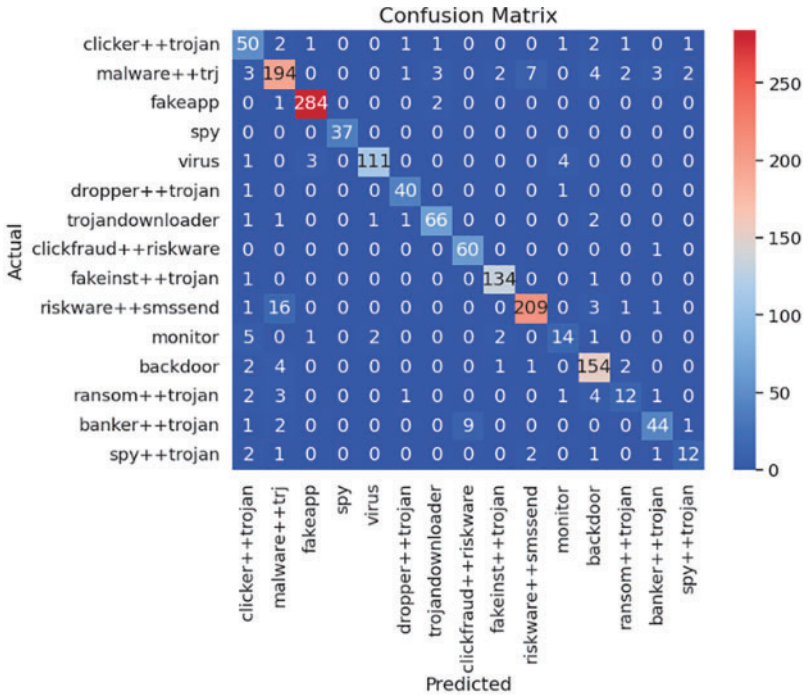

Figure 6: Confusion matrix of the performance of DeiT-MC on the MalNet-Image Tiny dataset accurately reflects performance by pinpointing the true class label

4.8.1 MalNet-Image Tiny Dataset

Fig. 7 displays the ROC curves of DeiT-MC on the MalNet-Image Tiny dataset. DeiT-MC demonstrates exceptional discriminatory capability, achieving an impressive micro-average AUC of 0.98 and a macro-average AUC of 0.97. These results indicate that DeiT-MC can effectively distinguish between different malware families, regardless of their prevalence in the dataset (micro-average) or balance across classes (macro-average). However, it's important to note that the ROC-AUC value for the 'ransom ++ trojan' class was comparatively lower at 0.79, indicating a less robust performance for this specific class. Moreover, DeiT-MC was compared with the work of Seneviratne et al. [6], revealing a notable improvement of 0.59% in micro-average AUC, surpassing the previously reported state-of-the-art performance. This highlights the robustness and effectiveness of DeiT-MC in accurately classifying diverse malware families.

4.8.2 MaleViS Dataset

Fig. 8 presents the ROC curves of DeiT-MC on the MaleViS dataset. The legend provides the AUC value for each class ROC curve. Notably, 11 out of the 26 malware classes attained an impressive AUC of 100%. For the remaining classes, the AUC values fall between 76% and 99%, indicating significant discriminatory power. These findings underscore the effectiveness of DeiT-MC in accurately classifying diverse malware families.

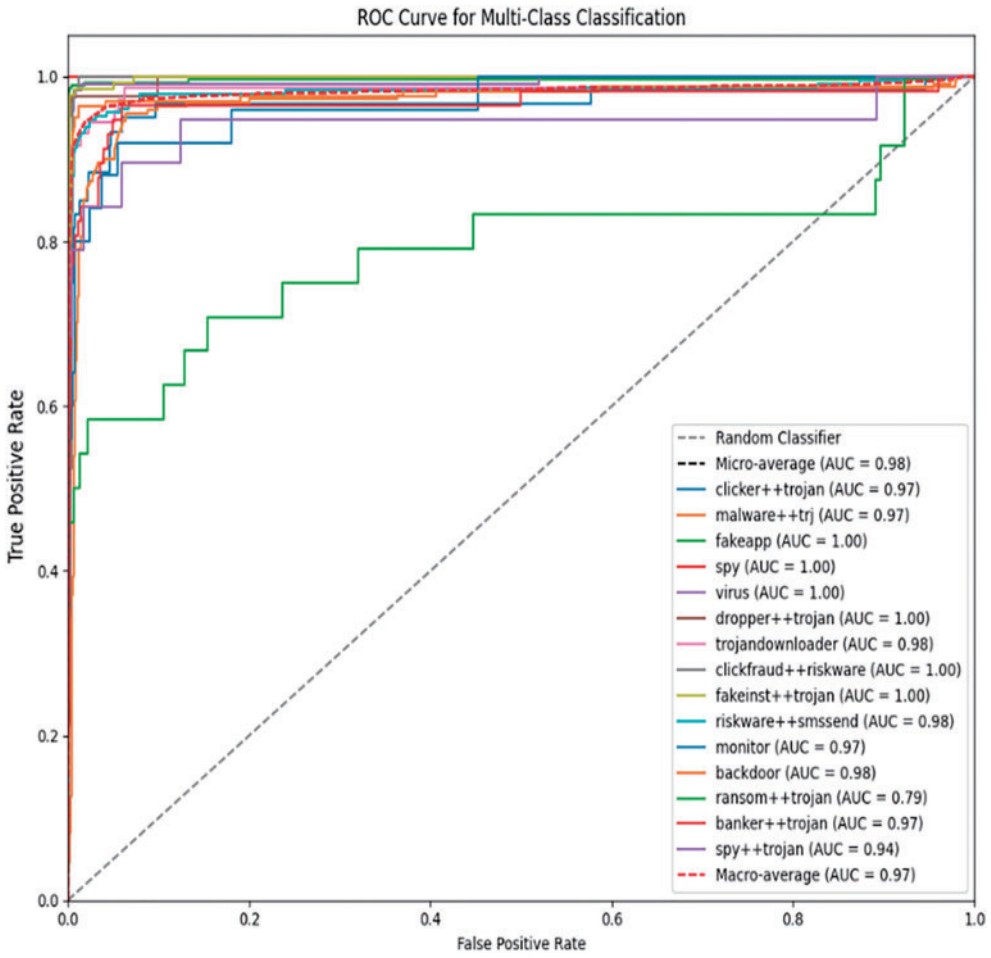

Figure 7: ROC curves of DeiT-MC on the MaleViS datasets. The legend provides the AUC value for each class ROC curve

4.9 Performance Assessment with Similar Works

This section evaluates the performance of the proposed DeiT-MC model in malware family classification tasks on the MaleViS dataset. We compare DeiT-MC's performance to other effective approaches in Table 6. DeiT-MC performs competitively on all metrics and garners a weighted average accuracy, precision, recall, and F1-score of 94.00%, demonstrating its competitiveness concerning state-of-the-art models. This competitive performance marks DeiT-MC as a viable solution for real-world malware classification tasks.

DeiT-MC has a competitive advantage over Malware-SMELL, as described by Barros et al. [22] in both efficiency and effectiveness. Although Malware-SMELL applies a modern zero-shot learning approach, it depends on sophisticated representations, such as latent and S-Space. The architecture of DeiT-MC is based on transformers; the process is direct since it works on raw input data to deliver high performance without using complex representations.

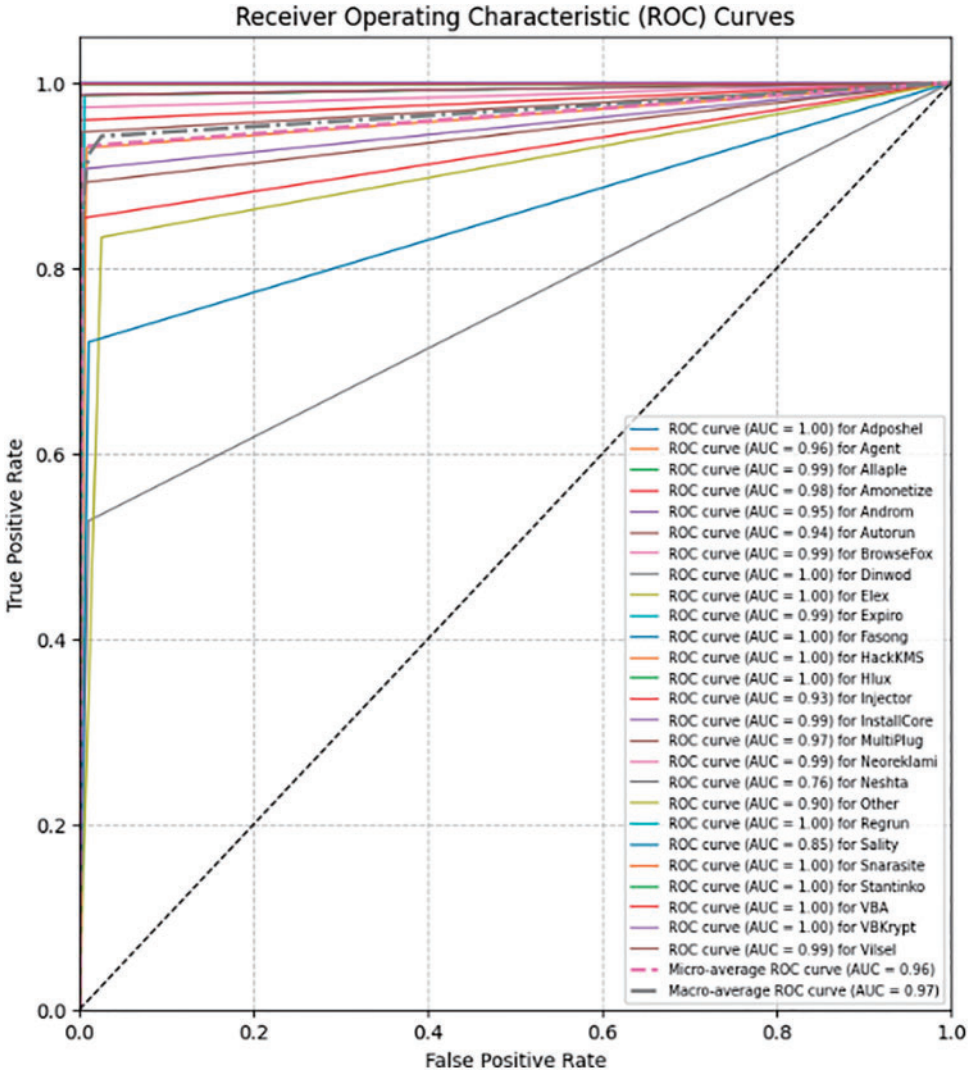

Figure 8: ROC curves of DeiT-MC on MaleViS datasets are displayed. The legend reports the AUC value for each class ROC curve

Table 6: Comparative analysis of models on MaleViS dataset

Study	Year	Method	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
Barros et al. [22]	2022	Zero-shot learning with k-Nearest Neighbors	93.70	93.16	93.06	93.42
Paik et al. [23]	2022	CNN-based encoder + LSTM-based decoder with an attention mechanism	93.49	92.9	93.49	92.82

(Continued)

Table 6 (continued)

Study	Year	Method	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
Nivaashini et al. [24]	2023	Ensemble Gaussian NB, decision trees, XGBoost, logistic regression, bagging, RF, SGD	90.07	91.00	90.00	90.00
DeiT-MC	**2024**	**DeiT**	**94.00**	**94.00**	**94.00**	**94.00**

Compared with EnDePMal by Paik et al. [23] which adopts a CNN-LSTM architecture, DeiT-MC is flexible and scalable due to its transformer-based architecture, thus capable of modeling long-range dependencies in the malware samples. These capabilities are then translated into superior performance, as shown by the higher accuracy, precision, and F1-score of DeiT-MC on the MaleViS dataset.

Finally, in contrast to Nivaashini et al. [24], DeiT-MC proposes a single, streamlined transformer-based architecture that works on raw input data, thereby avoiding the complications of engineered features and management inherent in most ensemble approaches combining multiple classifiers.

Table 7 illustrates the performance of DeiT-MC on the Malnet-Image Tiny dataset compared to other models. DeiT-MC has an accuracy of 92.00%, equated to its precision, recall, and F1-score metrics. In contrast, Naeem et al. [25] obtained an accuracy of 91.00% using ensembles of Inception V3, while Seneviratne et al. [6] attained an accuracy of 80.2% using Vision Transformer. This puts DeiT-MC at the front line for malware detection without using any complex feature extraction techniques.

Table 7: A comparative analysis of models on MalNet-Image Tiny dataset

Study	Year	Method	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
Naeem et al. [25]	2022	Ensembles inception V3	91.00	91.00	91.00	91.00
Seneviratne et al. [6]	2022	Vision transformer	80.2	86.7	89.00	87.8
DeiT-MC	2024	DeiT	**92.00**	**92.00**	**92.00**	**92.00**

Table 8 compares DeiT-MC and state-of-the-art models on key performance metrics: accuracy, F1-score, and whether disassembly is required. Disassembling machine code to human-readable instructions—deconstruction—can be computationally costly and complex. The table suggests that DeiT-MC offers 94.00% accuracy, slightly lower than the models of Hu et al. [10] and Deng et al. [13], which are 94.67% and 96.07%, respectively. Still, DeiT-MC has a competitive F1-score of 94.00% without disassembly; hence, it is more robust and efficient for execution in real-world tasks of malware detection. Moreover, transfer learning from the pre-trained DeiT models shows adaptability. DeiT-MC is an attractive solution, though a bit lower in accuracy, especially when disassembly is impractical.

Table 8: Performance comparison of DeiT-MC with state-of-the-art models

Author	Year	Method	Accuracy (%)	F1-score (%)	Does it require disassembly?
Hu et al. [10]	2020	Transformer with opcode	94.67	Not stated	Yes
Li et al. [12]	2021	Transformer, I-MAD (ST+) with assembly code	91.5	91.5	Yes
Deng et al. [13]	2023	Transformer with function call graph (FCG) *on* AZ dataset	96.07	95.20	Yes
DeiT-MC	**2024**	**DeiT Transformer with RGB images**	**94.00**	**94.00**	**No**

5 Conclusion

DeiT-MC has proved quite effective in malware classification on several benchmark datasets, with 94% accuracy on the Windows MaleViS dataset and 92% on the Android MalNet-Image Tiny dataset. This is attributed to the novel application of image-based visualization, transfer learning, and a hybrid GridBay Optimizer, effectively addressing class imbalance-related problems and complex malware families. Our ablation study underpins the critical role of each component involved in DeiT-MC. HGBO's incorporation significantly improves the tuning of hyperparameters while harnessing DeiT efficiently and effectively for classification. This quantifies the importance of tailored choices of parameters in maximizing model performance for an ML-based solution in any domain of malware detection applications or, for that matter, in any cybersecurity application.

However, DeiT-MC comes at a huge computational and memory cost that hinders its deployment on resource-constrained edge devices like IoT and mobile platforms. Quantization-aware training could alleviate the limitations of DeiT-MC. For example, Zafrir et al. [14] further showed how quantization-aware training can be used in BERT to achieve a 4x memory reduction while maintaining as much as 99% of the accuracy of the full precision. This approach thus optimizes model parameters to operate efficiently with lower bit precisions so it can be deployed on resource-constrained devices, such as edge devices. Quantization-aware training can ensure that the model has high accuracy even with reduced computational requirements, demonstrating practical benefits in edge computing scenarios. Similar techniques applied to DeiT-MC could make the model much more efficient and practical for edge-device deployment.

Despite achieving high accuracy, one obvious deficiency of DeiT-MC is that it was not benchmarked for real-time applications, which are most needed in time-critical scenarios such as on-the-fly file scanning and network traffic analysis. Further evaluation with real-world malware samples from platforms like VirusTotal and VirusShare will be critical for validating its robustness in practical cybersecurity settings.

6 Future Work

Looking ahead, accessibility and performance enhancement of DeiT-MC are very important. In particular, cloud-based deployment mitigates this concern about the access of DeiT-MC by offloading its execution to remote server infrastructure, thus further reducing local hardware resources required for high-performance GPUs or TPUs. This approach makes the model accessible from a much wider range of devices without compromising computational power or efficiency. Moreover, employing incremental and continual learning strategies is paramount to fine-tuning DeiT-MC against new malware threats. Techniques like LIME and Grad-CAM will enhance model explainability and trustworthiness in terms of their applications to cybersecurity. Additionally, exploring advanced techniques like evolutionary algorithms (such as Genetic Algorithms and Particle Swarm Optimization) that tune hyperparameters will improve performance. DeiT-MC can grow into a more powerful and adaptive solution for modern cybersecurity challenges by addressing the current limitations and further researching these areas.

Acknowledgement: None.

Funding Statement: The authors did not receive any specific funding for this study.

Author Contributions: The authors confirm their contribution to the paper as follows: study conception and design: Boadu Nkrumah, Michal Asante, Gaddafi Adbdul-Salam; data collection: Boadu Nkrumah; analysis and interpretation of results: Boadu Nkrumah, Wofa K. Adu-Gyamfi; draft manuscript preparation: Boadu Nkrumah. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The data will be available upon request.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

[1] Sonicwall Inc., "SonicWall Cyber Threat Report 2023," 2023. Accessed: Nov. 27, 2023. [Online]. Available: https://www.sonicwall.com/2023-cyber-threat-report/

[2] J. Geng, J. Wang, Z. Fang, Y. Zhou, D. Wu and W. Ge, "A survey of strategy-driven evasion methods for PE malware: Transformation, concealment, and attack," *Comput. Secur.*, vol. 137, 2024, Art. no. 103595. doi: 10.1016/j.cose.2023.103595.

[3] A. A. Alhashmi *et al.*, "Hybrid malware variant detection model with extreme gradient boosting and artificial neural network classifiers," *Comput. Mater. Contin.*, vol. 76, no. 3, pp. 3483–3498, 2023. doi: 10.32604/cmc.2023.041038.

[4] S. Morgan, "Cybercrime to cost The world $10.5 trillion annually by 2025," 2023. Accessed: May 15, 2023. [Online]. Available: https://cybersecurityventures.com/-cybercrime-damages-6-trillion-by-2021

[5] M. Dener and S. Gulburun, "Clustering-aided supervised malware detection with specialized classifiers and early consensus," *Comput. Mater. Contin.*, vol. 75, no. 1, pp. 1235–1251, 2023. doi: 10.32604/cmc.2023.036357.

[6] S. Seneviratne, R. Shariffdeen, S. Rasnayaka, and N. Kasthuriarachchi, "Self-supervised vision transformers for malware detection," *IEEE Access*, vol. 4, p. 1, 2022. doi: 10.1109/ACCESS.2022.3206445.

[7] K. Shaukat, S. Luo, and V. Varadharajan, "A novel deep learning-based approach for malware detection," *Eng. Appl. Artif. Intell.*, vol. 122, 2023, Art. no. 106030. doi: 10.1016/j.engappai.2023.106030.

[8] T. Van Dao, H. Sato, and M. Kubo, "ZSL-SLCNN: Zero-shot learning with semantic label CNN for malware classification," in *2023 12th Int. Conf. Control. Autom. Inf. Sci.*, 2023, vol. 22, pp. 572–577. doi: 10.1109/ICCAIS59597.2023.10382321.

[9] R. Oak, M. Du, D. Yan, H. Takawale, and I. Amit, "Malware detection on highly imbalanced data through sequence modeling," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2019, pp. 37–48. doi: 10.1145/3338501.3357374.

[10] X. Hu, R. Sun, K. Xu, Y. Zhang, and P. Chang, "Exploit internal structural information for IoT malware detection based on hierarchical transformer model," in *Proc. 2020 IEEE 19th Int. Conf. Trust. Secur. Priv. Comput. Commun. Trust. 2020*, 2020, pp. 927–934. doi: 10.1109/TrustCom50675.2020.00124.

[11] K. Lakshmanna, P. Kumar, R. Maddikunta, and M. Iwendi, "A lightweight energy consumption ensemble-based botnet detection model for IoT/6G networks," *Sustain. Energy Technol. Assess.*, vol. 60, 2023, Art. no. 103454. doi: 10.1016/j.seta.2023.103454.

[12] M. Q. Li, B. C. M. Fung, P. Charland, and S. H. H. Ding, "I-MAD: Interpretable malware detector using Galaxy Transformer," *Comput. Secur.*, vol. 108, no. 4, 2021, Art no. 102371. doi: 10.1016/j.cose.2021.102371.

[13] X. Deng, Z. Wang, X. Pei, and K. Xue, "TransMalDE: An effective transformer based hierarchical framework for IoT malware detection," *IEEE Trans. Netw. Sci. Eng.*, vol. 11, no. 1, pp. 140–151, 2023. doi: 10.1109/TNSE.2023.3292855.

[14] Q. Fournier, G. M. Caron, and D. Aloise, "A practical survey on faster and lighter transformers," *ACM Comput. Surv.*, vol. 55, no. 14s, pp. 1–40, 2023. doi: 10.1145/3586074.

[15] Y. Jakhotiya, H. Patil, J. Rawlani, and S. B. Mane, "Adversarial attacks on transformers-based malware detectors," Nov. 2022, *arXiv:2210.00008v2*.

[16] R. G. Mantovani, "Hyper-parameter tuning of a decision tree induction algorithm," in *2016 5th Brazilian Conf. Intell. Syst.*, 2016, vol. 7, pp. 37–42. doi: 10.1109/BRACIS.2016.018.

[17] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, 2012.

[18] F. T. Algorain and J. A. Clark, "Bayesian hyper-parameter optimisation for malware detection," *Electronics*, vol. 11, no. 10, May 2022, Art. no. 1640. doi: 10.3390/electronics11101640.

[19] H. Alibrahim and S. A. Ludwig, "Hyperparameter optimization: Comparing genetic algorithm against grid search and bayesian optimization," *2021 IEEE Congr. Evol. Comput. (CEC)*, vol. 13, pp. 1551–1559, 2021. doi: 10.1109/CEC45853.2021.9504761.

[20] A. S. Bozkir, E. Tahillioglu, M. Aydos, and I. Kara, "Catch them alive: A malware detection approach through memory forensics, manifold learning and computer vision," *Comput. Secur.*, vol. 103, no. 1, 2021, Art. no. 102166. doi: 10.1016/j.cose.2020.102166.

[21] S. Freitas, R. Duggal, and D. H. Chau, "MalNet: A large-scale image database of malicious software," in *Int. Conf. Info. Knowl. Manag. Proc.*, 2022. doi: 10.1145/3511808.3557533.

[22] P. H. Barros, E. T. C. Chagas, L. B. Oliveira, F. Queiroz, and H. S. Ramos, "Computers & security malware-SMELL: A zero-shot learning strategy for detecting zero-day vulnerabilities," *Comput. Secur.*, vol. 120, no. 6, 2022, Art. no. 102785. doi: 10.1016/j.cose.2022.102785.

[23] J. -Y. Paik and R. Jin, "Malware family prediction with an awareness of label uncertainty," *Comput. J.*, vol. 67, no. 1, pp. 376–390, Jan. 2024. doi: 10.1093/comjnl/bxac181.

[24] M. Nivaashini, S. Aarthi, and R. S. Ramya, "MalNet: Detection of malwares using ensemble learning techniques," in *2023 7th Int. Conf. Electron. Commun. Aerosp. Technol. (ICECA)*, 2023, pp. 1469–1477. doi: 10.1109/ICECA58529.2023.10395740.

[25] H. Naeem, B. M. Alshammari, and F. Ullah, "Explainable artificial intelligence-based IoT device malware detection mechanism using image visualization and fine-tuned cnn-based transfer learning model," *Comput. Intel. Neurosci.*, Jul. 2022. doi: 10.1155/2022/7671967.