Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.079701

ARTICLE

MalDetect-IoT: Enhanced IoT Malware Variant Detection with a Deep Stacked Ensemble Approach

Muhammad Shaheer[1], Feng Zeng[1,*], Aqsa Yasmeen[2], Mudasir Ahmad Wani[3,*], Kashish Ara Shakil[4] and Muhammad Asim[5]

[1]School of Computer Science and Engineering, Central South University, Changsha, 410000, Hunan, China
[2]Department of Computer Science and Engineering, Govt. College University, Faisalabad, Pakistan
[3]College of Computer and Information Sciences, Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh, 13318, Saudi Arabia
[4]Department of Computer Sciences, College of Computer and Information Sciences, Princess Nourah bint Abdulrahman University, P.O. Box 84428, Riyadh, 11671, Saudi Arabia
[5]EIAS Data Science Lab, College of Computer and Information Sciences, Prince Sultan University, Riyadh, 11586, Saudi Arabia

*Corresponding Authors: Feng Zeng. Email: fengzeng@csu.edu.cn; Mudasir Ahmad Wani. Email: MAWani@imamu.edu.sa

Received: 26 January 2026; Accepted: 03 April 2026; Published: 08 May 2026

ABSTRACT: Malware remains a persistent and evolving threat to digital security, highlighting the need for advanced and resilient detection frameworks capable of mitigating increasingly sophisticated and evasive cyberattacks. Although deep learning ensembles have been explored, many existing approaches fail to balance computational efficiency with the diverse feature extraction capabilities needed for complex variants. To address this gap, this study proposes a novel stacking ensemble framework, MalDetect-IoT, which specifically eliminates the requirement for manual feature engineering and domain specific preprocessing traditionally required in malware classification. By fine-tuning two pre-trained models MobileNetV3 for its lightweight efficiency and Xception for its depthwise distinct convolutions within a stacked architecture, the ensemble achieves superior reliability and predictive accuracy while remaining suitable for resource limited Internet of Things (IoT) environments. The proposed approach leverages complementary features to identify nuanced structural characteristics in malware binaries transformed into images, achieving domain knowledge independence. The proposed approach was evaluated on two benchmark datasets, achieving accuracies of 98.75% on the Malimg dataset (9335 images) and 98.55% on the MaleVis dataset (14,226 images). Statistical validation via McNemar's test and a Cohen's kappa coefficient of 0.983 confirm that the framework consistently surpasses state of the art methodologies in effectively identifying malware instances.

KEYWORDS: Malware detection; Internet of Things (IoT); deep learning; stacking ensemble; transfer learning; image-based malware classification

1 Introduction

The Internet of Things (IoT) has fundamentally transformed the way humans interact with technology by enabling the seamless connectivity of billions of electronic devices. These interconnected systems span smart living environments and industrial automation sectors, facilitating efficient data exchange and intelligent decision-making [1]. Although the global network was forecasted to reach 43 billion devices by 2023, this growth has paradoxically expanded the international cyberattack surface. Such pervasive connectivity necessitates more robust frameworks to mitigate emerging threats [2]. IoT environments face

distinct vulnerabilities from malware, defined as unauthorized software intended to breach digital integrity. The 2016 Mirai attack, which compromised 2.5 million devices, exposed these fundamental security gaps, while current data indicates that IoT hardware is now targeted by an average of 5200 attacks per month [3,4]. Furthermore, with over 10,000 dangerous variants uncovered annually, the rapid evolution of threats has rendered traditional signature based detection methods rigid and ineffective for modern decentralized networks [5].

Modern cybersecurity requires the implementation of advanced defense paradigms to protect interconnected systems and users from catastrophic data breaches. In this context, machine learning (ML) and deep learning (DL) have become pivotal tools for improving the classification and categorization of malicious software. Pai et al. [6] introduced a systematic framework for IoT malware detection using the IoT23 dataset. Their study evaluated various ensemble strategies and automated frameworks, ultimately proposing SNIPE, a sparse neural network approach. These advancements provide analysts and researchers with effective tools for malware prevention, detection, and mitigation. Researchers have demonstrated exceptional success in detecting and classifying malware [7].

In recent studies, a deep boosted CNN integrated with an ensemble learning strategy was proposed for IoT malware detection. This approach combines multiple deep models to improve classification accuracy and robustness, showcasing the effectiveness of ensemble-based techniques in handling complex malware patterns [8]. Dong and Kotenko [9] proposed an image-analysis-based malware detection method for IoT environments that combines variational autoencoders with generative adversarial networks. This method extracts discriminative features, addressing the limitations of traditional signature-based and shallow feature-based detection approaches in heterogeneous IoT networks. Furthermore, efficient malware detection frameworks are crucial for protecting IoT networks. Recently, Javed et al. proposed EffiMalNetv3, a lightweight framework utilizing contrastive learning and attention-guided deep learning techniques for efficient malware detection in resource-constrained environments [10]. While DL and image-based malware representation have proven effective for classification, a significant challenge remains in achieving high detection accuracy while respecting the resource limitations of IoT edge devices. High-performance models are often computationally expensive, requiring industrial-scale hardware that is unavailable at the IoT node level. However, lightweight models frequently lack the structural depth needed to capture the complex and evolving features of zero-day malware, leading to higher misclassification rates in heterogeneous environments.

A critical review of current literature reveals a persistent trade-off between classification precision and computational efficiency. Many state-of-the-art ensembles suffer from architectural redundancy, where stacking identical deep models increases the computational overhead without yielding diverse feature perspectives. Furthermore, existing methodologies frequently rely on labor-intensive, domain-specific feature engineering, which lacks the flexibility to adapt to the dynamic nature of polymorphic malware.

The primary objective of this research is to bridge this gap by developing MalDetect-IoT, an advanced deep stacked ensemble framework. Unlike single model approaches that fail to capture the full variety of malware samples, our method utilizes architectural heterogeneity through transfer learning [11]. Integrating MobileNetV3 and Xception creates complementary learning paths that synthesize localized feature capture with deep structural insights. This synergistic approach enhances generalization and robustly classifies malware, all while ensuring the framework remains computationally lightweight for IoT environments.

This study provides the following main contributions to the field of IoT security:

- We propose a deep stacked ensemble that integrates two heterogeneous deep learning models. This approach captures a wider range of feature representations than standalone architectures, thereby enhancing classification robustness across diverse threat scenarios.

- Our methodology eliminates the requirement for domain specific feature engineering. By converting raw malware to images, the system offers an automated, adaptable pipeline for dynamic threat detection.
- We optimized the base models MobileNetV3 and Xception to eliminate redundant parameters. This innovation maintains high tier accuracy while drastically lowering the computational overhead, making it viable for resource constrained IoT environments.
- The framework achieved exceptional accuracies of 98.75% on the MALIMG dataset and 98.55% on the MaleVis dataset, outperforming current benchmarks and individual constituent models.

The remainder of this paper is organized as follows. Section 2 presents an overview of the related work, Section 3 describes the proposed methodology, Section 4 presents the results and discussion, and Section 5 concludes the paper.

2 Related Work

The term "malware detection" refers to the method that determines the authenticity of an executable file is malicious, and then classifying the file into the relevant family of malware. The purpose of carrying out this approach is to discover software that has malicious intent. Through the automation of the detecting method and the encouragement of cross-disciplinary versatility, ML and DL techniques are rapidly improving intrusion detection systems (IDS). To this end, continuous enhancements in detection capabilities are being developed, with significant advancements in the field being realized through the robust application of ML techniques. Features engineering, model training, dataset development, and performance evaluation are some of the processes that are involved in the design of these advanced techniques. The design of these techniques further involves data synthesis and performance benchmarking. This section surveys existing literature on the application of ML and DL to malware detection and evaluates the resulting research outcomes. By examining various algorithms, this review assesses their capability to classify a wide range of malware types while synthesizing key innovations and recent breakthroughs.

Smmarwar et al. [12] provided a comprehensive review of malware detection frameworks, systematically categorizing the literature into feature selection (FS) techniques, ML-based approaches and DL-based architectures. Their study identifies several critical research gaps, most notably the challenges of class imbalance, zero-day detection, and obfuscated malware. Furthermore, they emphasize the necessity for cross-platform generalization and lightweight feature selection schemes, ultimately identifying advanced deep learning ensembles and reinforcement learning as the most promising trajectories for future security systems. Kumar et al. [13] leveraged the malware dataset to evaluate a deep learning architecture across 9458 grayscale images and benign datasets, achieved an accuracy of 98% across 25 different classes, indicating the potential of image-based deep learning methods for robust malware detection. Schultz et al. [14] have created a static malware detection method that finds out features from programs, such as strings, executables and byte n-grams. The approach uses a classifier called Multinomial Naïve Bayes (MNB) to do the classification, which gives it an accuracy of 97.11%. The increasing number of malware attacks on IoT devices shows how limited traditional methods are that depend heavily on signature databases and the skills of malware researchers. Deng et al. [15] utilized a three-channel malware visualization technique based on assembly instructions and Markov transition probabilities, combined with a lightweight CNN for classification, demonstrating superior performance compared to traditional grayscale and byte-based visualization methods. D'Angelo et al. [16] proposed a data mining-based classification approach using static analysis of executable files. Features such as byte sequences, string information, and Portable Executable (PE) file characteristics are extracted and used to train machine learning classifiers, including RIPPER and Naïve Bayes for malware detection. Naeem et al. [17] presented a proficient malware classification of images system framework intended for IoT architecture. This system basically converts the malware files into grayscale

images and then extracts both the global and local features from these images for final predictions. Qaisar et al. [18] propose AMALGAN, a generative adversarial network-based approach for Android malware detection that leverages image representations of malware for classification, fine-tuning both generator and discriminator as auxiliary classifiers. The model was trained on an image-based malware dataset and achieved strong performance with 95.24% accuracy outperforming several state-of-the-art methods. The framework proposed by Vishwakarma and Kesswani [19] introduce a deep neural network-based intrusion detection system (DIDS) for IoT environments to enable real-time attack detection. The model was trained using NetFlow-based benchmark datasets, including NF-BoT-IoT, NF-ToN-IoT, NF-CSE-CIC-IDS2018, and NF-UNSW-NB15, which capture network traffic behavior in IoT systems and this model achieved high performance with accuracy reaching up to 99.21% in binary classification and approximately 97.48% in multi-class classification. Saleh [20] proposed a 2D-CNN architecture with data augmentation techniques for IoT malware detection, automatically extracting features directly from malware images in the Malimg dataset without manual feature engineering. Zooming and flipping augmentations were applied to reduce overfitting, enabling the model to train efficiently in only 10 epochs. The proposed single-CNN model achieved 98.86% accuracy, outperforming DenseNet, VGG16, and VBDN on the Malimg dataset.

In recent studies, Smmarwar et al. [21] proposed an AI-powered Android malware detection (AIMD) framework that extracts static, dynamic, and memory-based features from APK files, applied DeepWalk graph embedding and Red Deer Algorithm for feature optimization and classifies using an ensemble of SVM, Decision Tree, Random Forest, and Extra Trees with bagging, boosting, stacking, and soft voting. Evaluated on CICInvesAndMal2019 and CICMalMem2022 datasets, the framework achieved 98.78% and 99.99% accuracy, respectively, outperformed against obfuscated and zero-day malware. Kumar et al. [22] proposed IMCNN, a deep CNN for malware detection using pre-trained models like VGG16, VGG19, InceptionV3 and ResNet50 for feature extraction. The method was tested on real-world malware datasets, achieved accuracy of 92.11% on the real-world dataset. Kasongo and Sun [23] developed a DL-based IDS model using a feature selection approach with the Extra-Trees classifier to rank features on the UNSW-NB15 and AWID datasets. The top-ranked features were used in a feed-forward deep neural network (FFDNN) for attack classification. In both the multiclass and binary classification and performed better as compared to others like SVMs, decision trees and random forests.

The authors [24] proposed an image-based malware detection approach that converts executable binaries into grayscale images using n-gram features in the DCT domain. A combination of shallow and deep learning models including ResNet, was evaluated along with an ensemble feature fusion strategy. The method was tested on the MaleX dataset (≈1 million samples) and achieved 96% accuracy, demonstrating strong generalization and competitive performance. Awan et al. [25] introduced a CNN-based malware classification model with spatial attention, removing the need for manual feature engineering. Evaluated on the Malimg dataset, the model achieved 97.68% accuracy and demonstrated the effectiveness of attention mechanisms in image-based malware detection. The approach focuses on image-based features and ignore dynamic behavior of malware, which is important for detecting advanced or evolving threats. Ben Jabra et al. [26] introduced a custom CNN architecture for malware detection, which they compared against seven established pre-trained models using the Malimg dataset. By leveraging L1,L2 regularization and dropout, their model attained an accuracy of 98.26%. This custom approach proved more effective than the transfer learning alternatives for this specific task. Javed et al. [27] proposed MSMC-MobileNet, a modified MobileNetv3 integrating SE, ASPP and FPP modules for multi-scale IoT malware detection using byteplot images. With dropout regularization and data augmentation, the model achieved 98.79% accuracy and a perfect AUC of 1.00 on combined Malimg and MaleVis datasets.

Kumari et al. [28] proposed a deep learning-based approach for multi-class malware classification using CNNs on grayscale image representations of malware binaries. Their method leveraged transfer learning to facilitate model training and learn hierarchical features for classification. Three different CNN models were developed and achieved a validation accuracy of approximately 97%. The authors [29] proposed an image-based malware detection approach using a deep CNN trained on the Malimg dataset, which contains 9339 samples across 25 malware families. The malware binaries were converted into 64 × 64 grayscale images to reduce computational complexity while preserving important features. Their model achieved an accuracy of 96%. Building upon the limitations identified in existing studies, this research introduces a DL-based framework for malware classification. Unlike prior approaches that rely on single-model architectures, the proposed method employs a novel deep stacked ensemble model that integrates multiple complementary learning paradigms. This architectural advancement enables more effective capture of both local and global malware characteristics, thereby enhancing detection accuracy and robustness. Consequently, the proposed framework offers a more reliable and scalable solution for identifying malicious software in complex and evolving environments. Table 1 outlines the key contributions and performance metrics of the related works discussed in this section.

Table 1: Comparison of existing malware detection methods.

Model/Method	Dataset	Accuracy (%)	Reference
DL Network	25 Malware Families (9458 Grayscale Images) and Benign Dataset	98.00	Kumar et al. [13]
Static MNB Classifier	Program Features (Strings, byte n-grams)	97.11	Schultz et al. [14]
MCTVD	Microsoft Malware Classification Dataset (Kaggle, 2015)	99.44	Deng et al. [15]
Static Feature Extraction and Data Mining Classifiers	7.3K Malware Applications	99.00	D'Angelo et al. [16]
Malware Classification of Images System (grayscale)	9342 Windows malware samples	97.40	Naeem et al. [17]
AMALGAN (GAN-based)	Image-based malware dataset	95.24	Qaisar et al. [18]
DL-based Anomaly IDS for IoT	NetFlow-based benchmark datasets	97.48	Vishwakarma and Kesswani [19]
2D-CNN + Data Augmentation	Malimg dataset	98.86	Saleh [20]
AIMD (AI-powered Android malware detection)	CICInvesAndMal2019, CICMalMem2022	98.78/99.99	Smmarwar et al. [21]

(Continued)

Table 1 (continued)

Model/Method	Dataset	Accuracy (%)	Reference
IMCNN + Pre-trained CNNs (VGG16, VGG19, InceptionV3, ResNet50)	Real-world malware dataset	92.11	Kumar et al. [22]
FFDNN + Extra-Trees Feature Selection	UNSW-NB15, AWID	99.66	Kasongo and Sun [23]
Deep learning-based Model	MaleX dataset	96.00	Mohammed et al. [24]
SACNN (Spatial Attention CNN)	Malimg (9389 grayscale images, 25 families)	97.68	Awan et al. [25]
Custom CNN + 7 pre-trained models	Malimg dataset	98.26	Ben Jabra et al. [26]
MSMC-MobileNetv3 (SE + ASPP + FPP)	Malimg + MaleVis datasets	98.79	Javed et al. [27]
Deep CNN	Malimg datasets	96	Omran musa and Tahrir Younis [29]

3 Proposed Methodology

This section provides a detailed description of the MalDetect-IoT framework, outlining the transition from raw binary data to automated malware classification. The proposed methodology utilizes a hybrid stacked ensemble that integrates two distinct deep learning architectures, MobileNetV3 and Xception to capture complementary feature sets from malware images. By converting raw malware binaries into visual representations, the system achieves domain knowledge independence, effectively eliminating the requirement for manual feature engineering or labor intensive domain specific preprocessing. Subsequent sections delineate the dataset attributes, architectural selection logic and fine-tuning procedures, concluding with the mathematical derivation of the ensemble's decision-making process.

3.1 Datasets

To rigorously evaluate the proposed framework, two well established benchmark datasets from the malware visualization domain were employed, the Malimg dataset and the MaleVis dataset. Together, these datasets provide a diverse and representative collection of malware families, enabling a comprehensive assessment of the model's classification capability across varying visual patterns and binary structures. Table 2 provides a comprehensive breakdown of the dataset specifications used in this research.

Table 2: Summary of the datasets used for evaluation.

DataSet	No. of Images	No. of Classes
Malimg Dataset	9335	25
MaleVis Dataset	14,226	26

3.1.1 Malimg Dataset

The Malimg dataset [30] serves as a prominent benchmark within the domain of visual malware analysis, facilitating the evaluation of diverse image-based detection frameworks. This benchmark includes 9335 samples spanning 25 malware families, featuring widely studied classes such as Adialer C, Autorun K, Lolyda AT, and Yuner A, among others. Each image is a direct binary to image conversion of a malware executable in which the raw bytes of the file are mapped to pixel intensity values and then reshaped into a two dimensional matrix. This approach preserves the structural layout of the binary, making patterns such as repetitive code blocks, data sections, and encrypted payloads visually distinguishable across different malware families. The inherent visual diversity of the 25 classes provides a challenging and realistic classification task, making Malimg an ideal benchmark for evaluating image-based malware detection systems. Fig. 1 displays a dataset diversity chart.

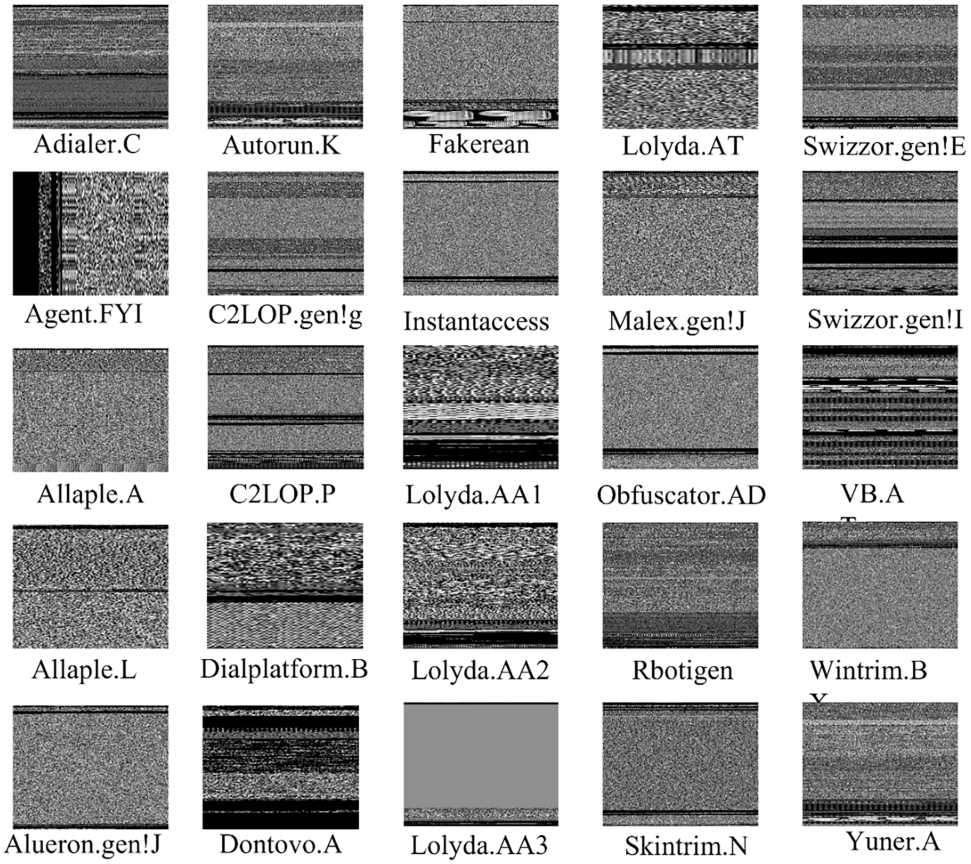

Figure 1: Random samples from the Malimg dataset.

3.1.2 MaleVis Dataset

The MaleVis [31] dataset extends the malware visualization paradigm by providing a more extensive and class diverse collection. It contains 14,226 images organized across 26 malware categories, including families such as Allaple, BrowseFox, HackKMS, Injector, Sality and VBKrypt. Unlike Malimg, the MaleVis samples encompass both grayscale and color mapped visual representations, which introduces additional complexity and textural variation for the classifier to handle. This richer dataset serves as a rigorous stress test for the proposed ensemble, validating its generalizability beyond a single image modality. The combination of Malimg and MaleVis, spanning a total of 51 distinct malware classes and 23,561 images, ensures that the

proposed framework is evaluated under varied and realistic conditions. To better the visual representation, systematic random selection of results from the dataset has been consolidated in Fig. 2. Each of the classes contains images that are representative of a particular visual style. The distinct visual patterns across various malware families facilitate the rapid discrimination of samples, which is a critical feature for effective class separation. Consequently, this dataset remains an invaluable resource for evaluating the efficacy of automated detection systems.

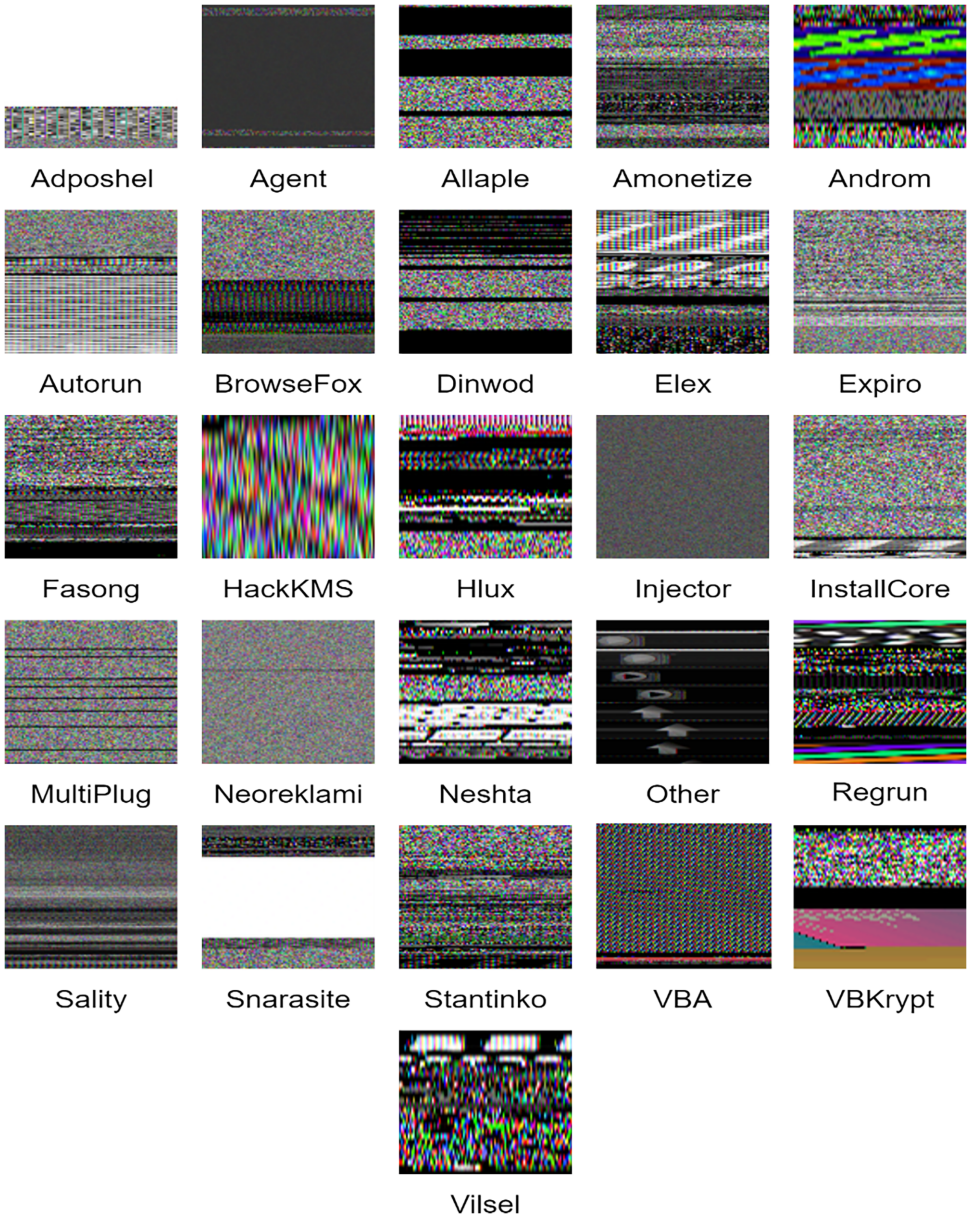

Figure 2: Random samples from MaleVis dataset.

3.1.3 Data Preprocessing Pipeline

To prepare both datasets for model training, a standardized preprocessing pipeline was implemented, encompassing the following sequential stages:

1. **Image Resizing:** All images were uniformly resized to 224×224 pixels. This standardization ensures dimensional consistency across the dataset, which is a prerequisite for batch processing in CNN architectures pretrained on ImageNet.
2. **Pixel Normalization:** Pixel values were normalized to the range $[0,1]$ by dividing by 255. This transformation aligns the input distribution with the statistical properties, on which the base models were pretrained, thereby accelerating gradient convergence during fine-tuning.
3. **Dataset Shuffling:** Samples were randomly shuffled prior to training to prevent sequential ordering from introducing positional bias into the learned representations, thereby improving generalization.
4. **Train-Test Split:** The preprocessed data was partitioned into an 80% training set and a 20% test set. This split ensures sufficient training data while maintaining a statistically meaningful holdout set for unbiased performance evaluation.

3.2 Deep CNN Architectures

The backbone of the MalDetect-IoT framework relies on two architecturally distinct pre-trained CNN models, MobileNetV3 [32] and Xception [33]. Rather than an arbitrary selection, these architectures were chosen for their proven efficacy and complementary strengths in extracting discriminative features from malware-image representations. When converted to images, malware binaries exhibit a dual-layered visual structure: localized, fine-grained byte-level noise patterns encoding signature-like features, and large-scale, global structural textures reflecting the overall layout of code sections and data segments. Consequently, no single architecture can capture both spatial dimensions optimally. MobileNetV3 excels at efficiently capturing localized spatial features, while Xception is particularly adept at learning global, high level structural representations. Their integration within a stacked ensemble therefore provides a more complete description of the malware image than any single model approach.

3.2.1 MobileNetV3

MobileNetV3 [32] is an efficient CNN architecture Optimized for mobile devices and embedded vision applications. With the increasing demand for neural networks specifically engineered for platforms characterized by limited processing power, restricted memory capacity, and tight energy budgets, MobileNetV3 addresses these challenges through several architectural innovations. Central to its efficiency are depthwise separable convolutions, which split traditional convolution into two steps, a depthwise convolution that processes each input channel separately, followed by a pointwise convolution that combines the results. This approach significantly reduces computational cost while maintaining performance.

MobileNetV3 also employs lightweight building blocks optimized for minimal computation and memory usage while effectively extracting pivotal characteristics. Its development builds on the advances of MobileNetV1, which introduced depthwise separable convolutions, and MobileNetV2, which added inverted residuals and linear bottlenecks. Inverted residuals expand feature maps before convolution and then reduce them, while linear bottlenecks reduce computational complexity by avoiding non-linear transformations at narrow layers. Enhancing these concepts, MobileNetV3 integrates squeeze and excitation modules and the h-swish activation function. Squeeze and excitation adaptively weights feature channels, boosting representational power and accuracy, while h-swish provides efficient non-linear activation, reducing computation and storage needs. To accommodate varying hardware requirements, MobileNetV3 provides a Large variant for high-performance benchmarks and a Small variant optimized for the low-memory and low-latency demands of image-based malware classification.

3.2.2 Xception

Xception [33] is a state-of-the-art CNN architecture renowned for its superior performance and efficiency. Its unique design allows it to maintain high-throughput accuracy, making it a preferred choice for demanding visual analysis applications. A distinguishing feature of Xception is its extensive use of depthwise separable convolutions. In the field of CNNs, the convolutions are essential for feature extraction from images. To optimize feature extraction, Xception utilizes a two-step depthwise separable convolution strategy that significantly diminishes complexity relative to conventional methods. By first applying independent filters to each channel and subsequently integrating the results via pointwise convolutions, the architecture captures comprehensive spatial data with high efficiency. This dual-stage approach is particularly effective for extracting hidden malicious patterns while remaining within the strict processing constraints of IoT environments.

The distinctive architecture of Xception across the mere use of depthwise separated convolutions. It is designed to facilitate the model acquisition of highly discriminative characteristics. In computer vision, discriminative characteristics include those which can proficiently differentiate between various objects, textures, or patterns within an image. The architecture of Xception has been carefully developed that recognize and highlight these qualities throughout this learning procedure. Xception has demonstrated significant efficacy in malware classification. Malware classification is a diverse endeavour that needs an analysis of several attributes of possibly dangerous software for determining its harmfulness. The proficiency of Xception in learning distinct characteristics and reliability provide it an optimal choice for this application. It can examine the patterns and behaviours of malware programs, analogous to its analysis of images for visual patterns, and classify them with accuracy. In this field, it has attained cutting-edge results, overcoming other alternative methods. Moreover, it can achieve this while reducing the required processing resources. This indicates that security systems can utilize Xception to swiftly and precisely detect malware without significantly taxing the computational infrastructure, hence improving the entire security and effectiveness of the whole system. The distinctive architecture of Xception and functionalities render it a crucial tool for both computer vision and malware categorization endeavors.

3.3 Fine-Tuning Strategy

The MalDetect-IoT framework adopts a transfer learning strategy, initializing both base models with ImageNet weights to exploit their pre-learned hierarchical representations. While the early layers are frozen to retain fundamental visual primitives (e.g., edges and gradients), the terminal layers are unfrozen for task-specific optimization. The specialized classification head appended to each architecture comprises the following layers:

- **Global Average Pooling (GAP):** Spatially aggregates the final convolutional feature maps into a fixed length vector. GAP retains channel wise discriminative information while eliminating large, parameter heavy fully connected layers, acting as a powerful regularizer against overfitting.
- **Dense Layer (256 neurons, ReLU):** Introduces non-linear transformations to enable the model to learn complex, non linear combinations of the pooled features specific to malware image patterns.
- **Batch Normalization:** Normalizes mini batch activations to stabilize and accelerate training by reducing internal covariate shift, also acting as an implicit regularizer.
- **Dropout (p = 0.3):** Randomly deactivates 30% of neurons per training step, preventing feature co-adaptation and reducing overfitting particularly beneficial given the relatively limited size of the malware datasets compared to ImageNet.

- **Output Dense Layer (Softmax):** A final dense layer with K neurons and Softmax activation generates a normalized probability distribution over all malware classes. The argmax of this distribution constitutes classification decision of the base model, subsequently passed to the meta learner.

3.4 Proposed MalDetect-IoT Ensemble Architecture

A MalDetect-IoT is a hybrid stacked ensemble approach that we propose to address the drawbacks of individual models in the classification of malware images. Single models might not succeed in addressing the complex structure of malware datasets, but ensemble approaches capitalize on the advantages of several models to improve classification precision. The principal benefit of stack ensemble is its capacity to utilize the varied capabilities of distinct base models, thus enhancing overall forecast robustness and accuracy [34]. The core innovation of MalDetect-IoT lies in its stacked ensemble architecture, which integrates the fine-tuned MobileNetV3 and Xception base models through a structured meta learning framework. As illustrated in Fig. 3, the overall pipeline is organized into three principal stages.

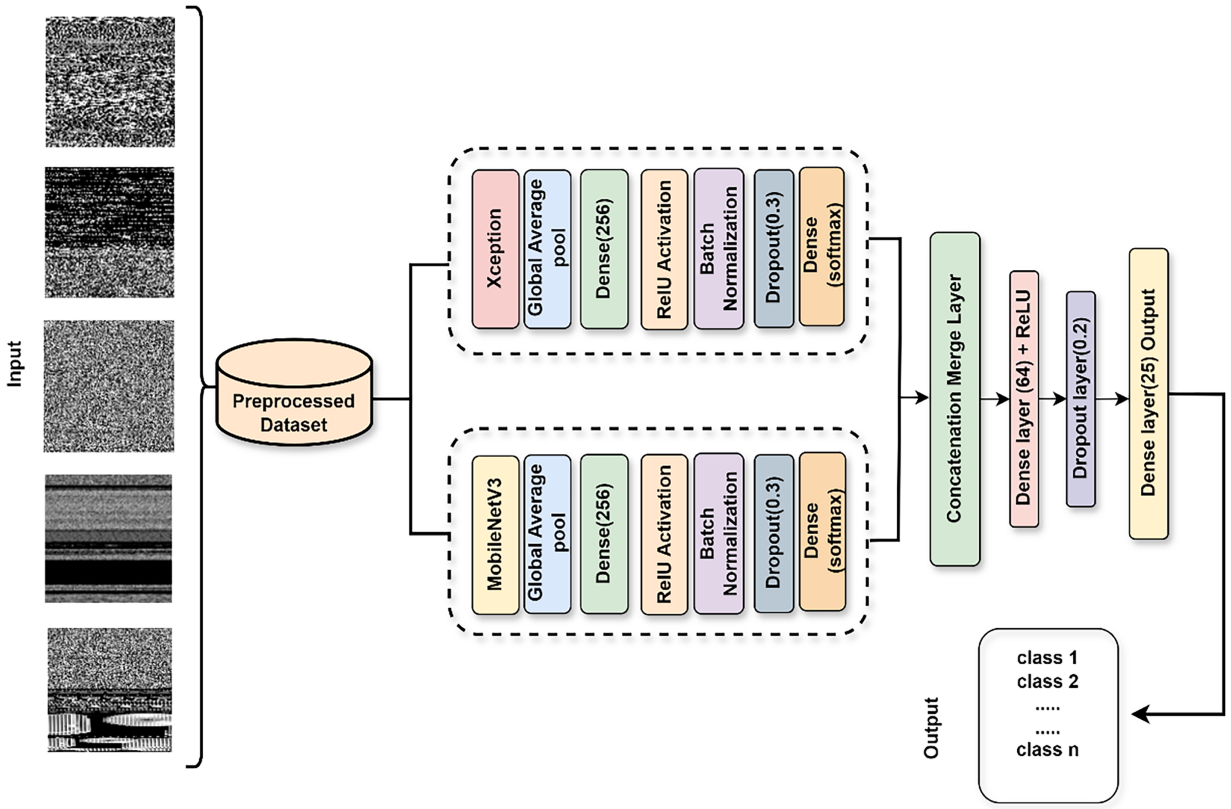

Figure 3: Architecture of the proposed MalDetect-IoT ensemble method for malware classification.

The interpretability of the proposed ensemble is rooted in its structured hierarchical decision making process. Unlike single model architectures, the transparency of this framework is achieved through a concatenation merge strategy, where the discrete output probabilities from MobileNetV3 and Xception are fused to serve as a high level feature set for the meta learner. This allows the meta learner to act as an informed arbitrator, weighing the confidence of each specialized base model across all potential malware families. The final Softmax activation layer provides a probabilistic explanation for each classification, generating a distribution that quantifies the certainty of the model. This flow ensures that the final decision is a visible

synthesis of weighted evidence from two distinct architectural perspectives, enhancing the overall reliability and trust in the MalDetect-IoT system.

3.4.1 Stage 1: Parallel Base Model Processing

A preprocessed input malware image $\mathbf{x} \in \mathbb{R}^{224\times224\times3}$, the image is simultaneously passed through both fine-tuned base models. MobileNetV3, parameterized by optimized weights $\boldsymbol{\theta}_1$, generates a class probability vector:

$$\mathbf{P}_1 = f_1(\mathbf{x},\, \boldsymbol{\theta}_1) \in \mathbb{R}^K \tag{1}$$

Concurrently, Xception, parameterized by weights $\boldsymbol{\theta}_2$, independently produces:

$$\mathbf{P}_2 = f_2(\mathbf{x},\, \boldsymbol{\theta}_2) \in \mathbb{R}^K \tag{2}$$

where K denotes the number of malware classes. The ImageNet-pretrained weights of both base models are frozen during meta-learner training to preserve the integrity of the feature representations acquired during fine-tuning.

3.4.2 Stage 2: Concatenation Merge Layer

The probability vectors $\mathbf{P}_1$ and $\mathbf{P}_2$ are concatenated along the feature dimension to form a combined meta-feature vector $\mathbf{H}$:

$$\mathbf{H} = Concat\,(\,\mathbf{P}_1, \mathbf{P}_2\,) \in \mathbb{R}^{2K} \tag{3}$$

This concatenation strategy is deliberately chosen over averaging or voting as it retains the comprehensive informational integrity of the predictive outputs from each individual architecture. The meta-learner receives a $2K$-dimensional input encoding the complete confidence distribution across all classes from both architectural perspectives, enabling it to resolve ambiguous cases where one model is confident and the other is uncertain, or where both are confident but disagree.

3.4.3 Stage 3: Meta-Learner Network

The meta-learner is a compact neural network trained on the concatenated meta-features $\mathbf{H}$. Its architecture is:

$$\text{Dense}(64,\ \text{ReLU}) \;\rightarrow\; \text{Dropout}(0.2) \;\rightarrow\; \text{Dense}(K,\ \text{Softmax})$$

Formally, the ensemble's final prediction is:

$$\hat{Y} = \mathcal{L}(\mathbf{H},\, \boldsymbol{\theta}_L) = \text{Softmax}(\mathbf{W} \cdot \sigma(\mathbf{U} \cdot \mathbf{H} + \mathbf{b}_1) + \mathbf{b}_2) \tag{4}$$

where $\mathbf{U}$ and $\mathbf{W}$ are the learnable weight matrices of the hidden and output layers, $\mathbf{b}_1$ and $\mathbf{b}_2$ are the bias vectors, $\sigma(\cdot)$ denotes the ReLU activation, and $\boldsymbol{\theta}_L = \{\mathbf{U}, \mathbf{W}, \mathbf{b}_1, \mathbf{b}_2\}$.

The meta-learner is optimized using the Adam optimizer with a learning rate of 0.0001 an order of magnitude smaller than the base models' rate of 0.001. This conservative rate is critical because large weight updates at the meta-learning stage would disrupt the high level semantic representations and degrade

ensemble performance. The model parameters are optimized by minimizing the categorical cross-entropy loss function, which is defined as follows:

$$\mathcal{L} = -\sum_{i=1}^{N} \mathbf{y}_i \cdot \log(\hat{\mathbf{y}}_i) \tag{5}$$

where $\mathbf{y}_i$ is the one-hot ground-truth label vector and $\hat{\mathbf{y}}_i$ is the predicted probability distribution for the i-th sample over all N training instances.

3.5 Automated Feature Extraction and Domain Knowledge Independence

A key characteristic of the MalDetect-IoT framework is the complete elimination of manual feature engineering. Traditional malware detection systems require domain experts to manually design and extract features, such as API call sequences, opcode histograms, imported library lists, or network traffic signatures rendering such systems brittle in the face of polymorphic and metamorphic malware variants. These sophisticated threats employ obfuscation techniques to dynamically alter their code signatures, successfully circumventing traditional analysis. By contrast, our framework automates the extraction process, identifying latent structural patterns that remain resilient against such evasive mutations. In the MalDetect-IoT framework, feature extraction is fully automated through the hierarchical convolutional architecture of the base models, specifically designed to capture the dual layered visual structure of malware binaries. The initial layers of MobileNetV3 and Xception function as generic visual detectors, identifying low level pixel gradients and local intensity variations that correspond to raw byte level noise.

As data propagates through the network, these features are abstracted into mid-level textures and ultimately into high-level structural descriptors of the binary layout. MobileNetV3 utilizes inverted residual blocks to efficiently weight localized spatial features, while Xception modules decouple cross channel and spatial correlations to capture long range structural dependencies and global textures, such as the arrangement of code sections or encrypted data blocks. The pointwise convolutions then fuse these per-channel representations into consolidated feature maps that represent the nuanced characteristics of the malware. Finally, the stacking strategy allows the meta-learner to synthesize these two complementary feature sets, enabling it to discern complex inter class patterns such as subtle variations between polymorphic variants that would be extremely difficult to define through manual, human defined procedures. This approach achieves complete domain-knowledge independence, ensuring the system remains inherently adaptable to emerging malware families without the need for specialized human expertise in cybersecurity.

The proposed framework is grounded in the assumption that binary to image conversion effectively captures structural complexity of malware through distinct visual patterns. By leveraging the feature transferability of pre-trained Xception and MobileNetV3 models, the methodology achieves domain knowledge independence, eliminating the need for manual feature engineering. Algorithm 1 outlines the overall computational workflow of the proposed MalDetect-IoT framework, including data preprocessing, feature extraction, and meta-learning.

Algorithm 1: MalDetect-IoT: stacked ensemble malware detection framework

Require: Malware image dataset $D = \{(x_i, y_i)\}_{i=1}^{N}$
Ensure: Trained ensemble classifier $\mathcal{E}$
1: **Step 1: Data Preprocessing**
2: **for** each sample $x_i \in D$ **do**
3: Resize x_i to 224×224

(Continued)

Algorithm 1 (continued)

Normalize pixel values to $[0,1]$
end for
Shuffle dataset and split into training/testing sets
Step 2: Base Model Initialization
Initialize MobileNetV3 with pretrained ImageNet weights θ_1
Initialize Xception with pretrained ImageNet weights θ_2
Train base models f_1 and f_2 on training dataset
Step 3: Feature Extraction and Base Predictions
for each input image x_i **do**
    $P_1^{(i)} = f_1(x_i, \theta_1)$
    $P_2^{(i)} = f_2(x_i, \theta_2)$
end for
Step 4: Stacking (Concatenation Layer)
for each sample i **do**
    $H_i = \text{concat}(P_1^{(i)}, P_2^{(i)})$
end for
Step 5: Meta-Learner Training
Initialize meta-learner $\mathcal{L}$:
    Dense(64, ReLU) → Dropout(0.2) → Dense(K, Softmax)
Train $\mathcal{L}$ on H_i using Adam optimizer (learning rate = 0.0001)
Minimize categorical cross-entropy loss
Step 6: Ensemble Prediction
for each test sample x **do**
    Compute $P_1 = f_1(x, \theta_1)$ and $P_2 = f_2(x, \theta_2)$
    Form $H = Concat(P_1, P_2)$
    $\hat{y} = \mathcal{L}(H)$
end for
return Final trained ensemble model $\mathcal{E} = \{f_1, f_2, \mathcal{L}\}$

3.6 Design Justification and Novelty

The novelty of the proposed MalDetect-IoT framework transcends the standard application of stacking by addressing the architectural redundancy typically found in homogeneous ensembles. While conventional methods often combine structurally similar models, yielding marginal improvements, this work implements a heterogeneous structural synergy by pairing MobileNetV3 and Xception. This specific combination is novel because it establishes a dual perspective learning path, MobileNetV3 utilizes inverted residuals for efficient localized feature capture, while Xception employs depthwise separable convolutions for global structural patterns. Furthermore, the training strategy introduces a disciplined dual learning rate protocol, where the meta-learner is restricted to a learning rate of 0.0001 an order of magnitude lower than the base models to preserve the integrity of high-level abstract features without disrupting the underlying knowledge gain. This ensures a lean computational footprint suitable for IoT edge deployment while achieving state-of-the-art accuracy.

This specific architectural pairing allows the meta learner to resolve classification ambiguities that a single architecture approach or a homogeneous ensemble would be unable to distinguish, particularly in complex, evolving malware families that may share either local byte signatures or global structural layouts

but not both. Furthermore, the framework design deliberately addresses the unique constraints of IoT deployment and the use of MobileNetV3 as one of the two base models ensures that the ensemble retains a lean computational footprint, while the stacking strategy confines the added computational overhead to a compact meta learner network rather than requiring additional heavy base models. The result is a system that achieves state of the art detection accuracy while remaining viable for deployment in resource-limited IoT edge environments.

4 Experiments and Results

4.1 Experimental Setup and Hyperparameters

The proposed models were trained on a hardware platform equipped with an AMD Ryzen 5 processor and an NVIDIA MX450 GPU. While the CPU managed data preprocessing and general computational tasks, the dedicated GPU was utilized to accelerate the high-dimensional matrix operations inherent in deep learning. To ensure optimal convergence and stability, the hyperparameters were strategically selected for both the base learners and the meta-learner. The Adam optimizer was employed due to its adaptive learning rate capabilities, which leverage first and second-moment estimates of the gradients to navigate complex optimization landscapes effectively.

The base architectures were trained with a learning rate of 0.001, providing a balance between rapid pattern recognition and stable convergence. A batch size of 64 was selected to optimize memory utilization and ensure consistent gradient updates. The training duration was set to 20 epochs, which proved sufficient for the models to extract discriminative features while mitigating the risk of overfitting or unnecessary computational overhead. In contrast, a more conservative learning rate of 0.0001 was implemented for the meta-learner. This reduction was intentional, as the meta-learner is responsible for synthesizing high-level latent representations from the base models, a lower learning rate facilitates more granular and disciplined weight adjustments. This ensures that the meta-learner refines the integrated knowledge without disrupting the established feature associations. The complete hyperparameter suite is summarized in Table 3.

Table 3: Hyperparameter configuration of the MalDetect-IoT framework.

Hyperparameter	Value	Justification
Base Model Learning Rate	0.001	Balanced convergence speed vs. stability
Meta-Learner Learning Rate	0.0001	Preserves high-level abstract features
Batch Size	64	Optimizes memory usage and training stability
Base Model Epochs	20	Sufficient for pattern learning; avoids overfitting
Optimizer	Adam	Adaptive learning rates per parameter
Dropout (Base Models)	0.3	Prevents co-adaptation and overfitting
Dropout (Meta-Learner)	0.2	Light regularization at fusion stage
Input Image Size	224 × 224 px	Compatible with ImageNet pretrained weights

4.2 Model Training Loss and Accuracy

The experiments are performed individually on both the datasets. The training progress for the MalDetect-IoT model is illustrated in Fig. 4, showing the loss and accuracy curves. The training loss exhibits a sharp initial decline, successfully converging toward a value near 0.005, which indicates that the meta-learner effectively synthesized the features from the base models. Concurrently, the training accuracy demonstrates a steady upward trend, surpassing 99% as the training progressed. The smooth convergence of these curves,

despite the diversity of the 9335 Malimg images and 14,226 MaleVis images, confirms the stability of the optimization process and the effectiveness of the 0.0001 learning rate used for the meta-learner.

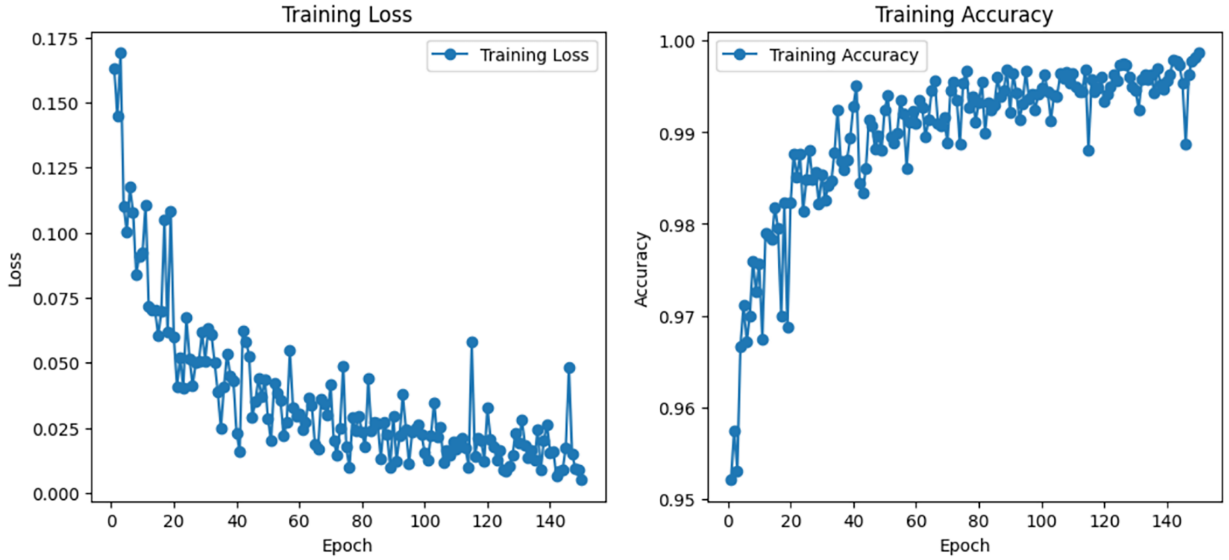

Figure 4: The training loss and accuracy diagram of the proposed model.

4.3 Evaluation Metrics

We employed standard evaluation metrics commonly used in model assessment, including accuracy, precision, recall, and F1-score. These metrics were utilized to perform comprehensive comparisons between different models, providing insights into their performance across various aspects of classification accuracy and effectiveness.

4.3.1 Accuracy

Accuracy is a metric that measures the overall correctness of a model by calculating the ratio of correctly predicted instances to the total number of instances. It gives a general sense of how well the model performs across all classes. The equation for accuracy is:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{6}$$

where TP is True Positive, TN is True Negative, FP is False Positive, and FN is False Negative. Accuracy is useful for balanced datasets but can be misleading for imbalanced datasets where the majority class dominates.

4.3.2 Precision

Precision, also known as positive predictive value, measures the proportion of true positive predictions among all positive predictions made by the model. It reflects the model's ability to correctly identify relevant instances and is particularly important in scenarios where the cost of false positives is high. The equation for precision is:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{7}$$

where TP is True Positives and FP is False Positives. High precision indicates a low number of false positives, meaning the model is good at predicting the positive class correctly.

4.3.3 Recall

Recall, also known as sensitivity or true positive rate, measures the proportion of true positive predictions among all actual positives. It reflects the model's ability to capture all relevant instances. Recall is crucial when the cost of false negatives is high. The equation for recall is:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{8}$$

where TP is True Positives and FN is False Negatives. High recall indicates a low number of false negatives, meaning the model is good at identifying all positive instances.

4.3.4 F1-Score

The F1 Score is the harmonic mean of precision and recall, providing a single metric that balances both. It is useful when you need to balance the trade-off between precision and recall, especially in cases of imbalanced datasets. The equation for the F1 Score is:

$$F1\text{-}score = \frac{2 \times P \times R}{P + R} \tag{9}$$

This metric ranges from 0 to 1, with 1 being the best possible score. It is particularly helpful when the positive class is rare, ensuring that both false positives and false negatives are considered.

4.4 Result Analysis

This section presents the results of the proposed model, including essential classification metrics for evaluating its effectiveness. We performed a comparative analysis to assess the efficiency of our model, contrasting the proposed ensemble architecture with individual models. Table 4 displays the quantitative results for the proposed ensemble model alongside the individual models in detecting different types of malware. The results demonstrate that all models, including MobileNetV3 and Xception, exhibit improved performance, highlighting their effectiveness in classifying malware samples. The proposed ensemble model demonstrates superior performance compared to individual models. In this proposed ensemble model, which incorporates the top two models, attains an accuracy of 98.75% on the Malimg dataset, with precision, recall, and F1-score values of 98.48%, 98.75%, and 98.41%, respectively. On the MaleVis dataset, it achieves an accuracy of 98.55%, with precision, recall, and F1-score values of 98.42%, 98.55%, and 98.33%, respectively. The results indicate that the ensemble approach effectively enhances classification performance by utilizing the strengths of various models. The ensemble model effectively integrates diverse predictions from multiple models, resulting in superior predictive accuracy and increased resilience across diverse malware classification benchmarks.

Table 4: Comparison of the proposed model with single models for malware detection, where bold values indicate the best performance.

Models	Accuracy	Precision	Recall	F1-score
MobileNetV3	97.91	97.94	97.91	97.90
Xception	97.70	97.84	97.70	96.00

(Continued)

Table 4 (continued)

Models	Accuracy	Precision	Recall	F1-score
Proposed Ensemble (Top 2 Models on Malimg)	**98.75**	**98.48**	**98.75**	**98.41**
Proposed Ensemble (Top 2 Models on MaleVis)	**98.55**	**98.42**	**98.55**	**98.33**

Table 5 presents a comprehensive comparison between the proposed MalDetect-IoT ensemble and several state-of-the-art architectures. This evaluation serves to benchmark the efficacy of our dual-model strategy against established deep learning frameworks. The proposed ensemble, which integrates MobileNetV3 and Xception, achieved a superior accuracy of 98.75% on the Malimg dataset and 98.55% on the MaleVis dataset. In contrast, other models such as InceptionV3, ResNet50V2, and VGG16 yield significantly lower accuracies of 94.18%, 94.62%, and 93.30%, respectively. These findings highlight the efficacy of the ensemble strategy, which outperforms baseline models by synthesizing the strengths of distinct learners. This unified approach provides robust feature representation, resulting in consistently high accuracy and operational stability in complex malware classification tasks.

Table 5: Comparative analysis of the proposed ensemble model with other pre-trained models for malware detection, where bold values indicate the best performance.

Models	Accuracy	Precision	Recall	F1-score
InceptionV3	94.18	92.82	94.18	93.13
ResNet50V2	94.62	93.26	94.62	93.67
VGG16	93.30	90.32	93.30	91.43
Proposed Ensemble (Top 2 Models on Malimg)	**98.75**	**98.48**	**98.75**	**98.41**
Proposed Ensemble (Top 2 Models on MaleVis)	**98.55**	**98.42**	**98.55**	**98.33**

MalDetect-IoT utilizes a hierarchical feature extraction pipeline to classify malware binaries through a visual lens. The base models automatically identify low-level visual primitives, which are then transformed into high-dimensional feature maps reflecting the binary's structural layout. A meta-learner then synthesizes these inputs, outperforming manual expert-driven procedures by identifying non-linear patterns in the data. This structured classification framework offers a scalable solution for IoT security, with a continued emphasis on transparency and model explainability.

The robustness of the proposed framework is demonstrated by its consistent performance across two distinct benchmark datasets, Malimg and MaleVis. By achieving high Cohen's kappa coefficients of 0.987 and 0.985 in Fig. 5, respectively, the model shows a strong concordance with ground truth labels that is statistically significant across a combined total of 23,561 malware samples. This adaptability is largely attributed to the use of transfer learning with MobileNetV3 and Xception, which allows the ensemble to leverage high-level features while maintaining domain independence from manual feature engineering. The framework's ability to maintain high accuracy across 51 distinct classes suggests it is well-positioned to generalize to diverse malware environments. In contrast, models like VGG16 and ResNet50V2 exhibit lower kappa coefficients, suggesting reduced consistency in their predictions. These findings highlight the enhanced stability and reliability of the proposed model in malware detection tasks.

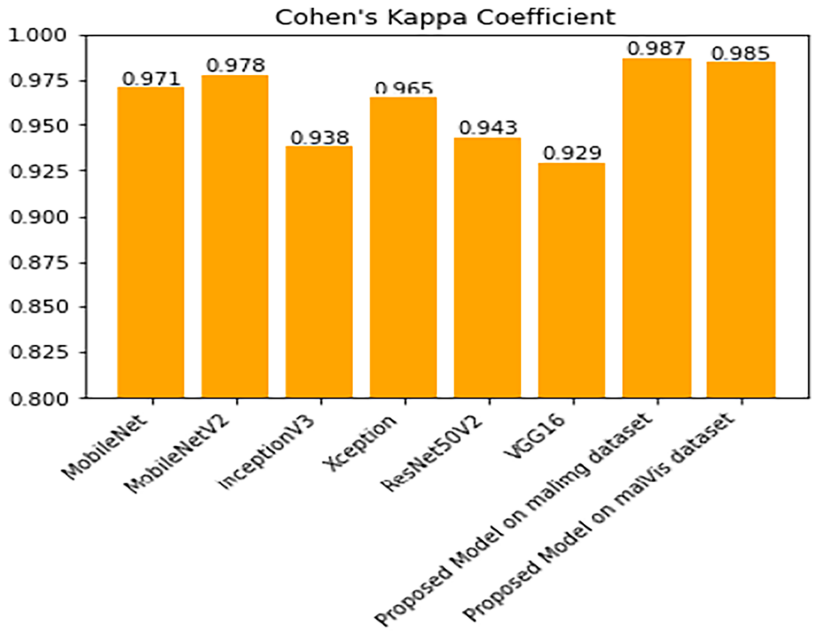

Figure 5: Comparison of cohen's kappa coefficient across different models for malware detection, highlighting the stability and consistency of predictions.

4.5 Statistical Analysis: McNemar's Test

We conducted McNemar's [35] test to evaluate the significance of differences in classification performance between pairs of models. This test helps determine whether any observed differences in accuracy are statistically significant, providing insights into the comparative effectiveness of different models. The results in Table 6 present the *p*-values obtained from McNemar's test for each pair of models. A low *p*-value (<0.05) indicates a statistically significant difference in performance between the models being compared. The *p*-values obtained for MobileNetV3, and Xception are 4.19e−109, and 2.91e−13, respectively. These extremely low *p*-values suggest significant differences in performance between each of these models, indicating that they are not equivalent in their ability to classify malware samples.

Table 6: McNemar's test results for pairwise model comparisons, indicating the significance of differences in classification performance.

Models	*p*-Value
MobileNetV3	4.19e−109
Xception	2.91e−13

4.6 Ablation Study and Comparison with Other Existing Methods

To evaluate the contribution of different architectural components in the proposed MalDetect-IoT framework, an ablation study and sensitivity analysis were conducted. The experiments analyze the impact of the ensemble architecture, dropout rate, and the number of neurons in the dense layer. First, the performance of the individual base models, MobileNetV3 and Xception was compared with the proposed ensemble model. The results indicate that the ensemble approach consistently achieves higher accuracy by combining complementary feature representations extracted by both constituent models. As shown in Table 7, the

ensemble model achieved the best performance with an accuracy of 98.75% on the Malimg dataset and 98.55% on the MaleVis dataset.

Table 7: Ablation study of the proposed MalDetect-IoT framework on Malimg and MaleVis datasets, where bold values indicate the best performance.

Experiment	Configuration	Malimg Accuracy (%)	MaleVis Accuracy (%)
MobileNetV3 Only	Single Base Model	97.91	97.38
Xception Only	Single Base Model	97.70	97.35
Proposed Ensemble	MobileNetV3 + Xception	**98.75**	**98.55**
Dropout = 0.1	Dense(64)	98.21	98.02
Dropout = 0.2	Dense(64)	**98.75**	**98.55**
Dropout = 0.3	Dense(64)	98.33	98.19
Dense = 32	Dropout 0.2	98.14	97.96
Dense = 64	Dropout 0.2	**98.75**	**98.55**
Dense = 128	Dropout 0.2	98.29	98.07

Furthermore, the sensitivity of the model to hyperparameters was analyzed by varying the dropout rate (0.1, 0.2, and 0.3) and the number of neurons in the dense layer (32, 64, and 128). The experimental results show that a dropout rate of 0.2 and 64 neurons provide the optimal configuration, achieving the highest detection accuracy on both datasets. Lower dropout values may increase the risk of overfitting, while higher values can lead to loss of useful feature information. Similarly, using too few neurons may limit feature learning capacity, whereas larger dense layers slightly increase model complexity without improving performance. These results confirm the effectiveness and stability of the selected architecture.

The comparison with existing methods serves multiple purposes in validating the efficacy and superiority of our proposed ensemble method for malware classification. Firstly, it offers a benchmark against which our model's performance can be assessed, providing insights into its relative effectiveness in the field. Secondly, by utilizing the Malimg dataset and MaleVis dataset employed by previous studies, we ensure a fair and direct comparison, eliminating potential biases introduced by variations in dataset composition or preprocessing techniques.

Table 8 provides a comprehensive overview of the accuracy results obtained by various existing methods, DL-Network, VGG19+SACNN, Custom CNN, Deep CNN, CNN and our proposed ensemble method. Among these methods, our ensemble approach achieves the highest accuracy of 98.75% on Mailimg dataset Respectively 98.55% on MaleVis dataset, signifying its superior capability in accurately classifying malware samples. The advantages of our proposed ensemble method lie in its utilization of ensemble learning principles, which enable the integration of diverse model predictions to enhance overall performance. By leveraging the strengths of multiple models, our approach mitigates individual model biases and errors, leading to improved classification accuracy and robustness. Furthermore, the flexibility inherent in ensemble learning allows for the seamless incorporation of different model architectures and training strategies, thereby enhancing the model's adaptability and effectiveness in addressing the dynamic landscape of malware threats.

Table 8: Comparison of accuracy results with existing methods, where bold values indicate the best performance.

Method	Accuracy
DL Network	98.00 [13]
VGG19 + SACNN	97.68 [25]
Custom CNN	98.26 [26]
CNN	97.00 [28]
Deep CNN Model	96.00 [29]
Proposed Ensemble Method on Malimg	**98.75**
Proposed Ensemble Method on MaleVis	**98.55**

4.7 Scalability and Computational Efficiency

The MalDetect-IoT framework is intentionally engineered for deployment in resource constrained IoT environments. The integration of MobileNetV3 allows the system to operate efficiently within the limitations of restricted compute power and energy. By utilizing inverted residuals and linear bottlenecks, the model minimizes memory consumption and latency. Furthermore, the conversion of malware binaries into image representations eliminates the need for expensive domain specific feature engineering. This reduction in preprocessing complexity, combined with the two step depthwise separable convolution methodology, ensures that the framework can scale to large-scale networks where processing thousands of samples quickly is essential.

4.8 Limitations

- While MalDetect-IoT achieves high detection accuracy, it is primarily challenged by evasive malware employing advanced runtime packing or obfuscation. Such threats may remain latent during image conversion and only manifest malicious behavior during execution.
- A limitation of MalDetect-IoT is potential adversarial evasion, where attackers inject byte-level noise to alter the malware's image without changing its function. While our heterogeneous ensemble increases the difficulty of such attacks, static analysis remains vulnerable.

4.9 Future Work

- To further address the inherent interpretability limitations of deep ensembles, future research will investigate the integration of techniques specifically tailored for stacked architectures. This includes exploring how individual sub-model gradients contribute to final ensemble predictions, thereby providing a more granular audit trail for cybersecurity analysts.
- To address the interpretability limitations of ensemble methods, future work could investigate the development of explainability techniques specifically tailored for ensemble models. These techniques could provide insights into how individual models contribute to ensemble predictions, enhancing transparency and trust in the model's decisions.
- Further research will explore the development of real-time malware detection systems based on ensemble methods. Deploying lightweight and efficient ensemble models capable of processing data streams in real-time could enhance the timeliness and effectiveness of malware detection in dynamic environments.

5 Conclusion

In conclusion, this research emphasizes the necessity for sophisticated detection strategies capable of addressing the expanding threat landscape of modern malware. While standalone approaches are effective in specific contexts, they are inherently constrained in their ability to capture the diverse, dynamic and evolving nature of modern malware. To address this challenge, we have introduced a MalDetect-IoT ensemble method for malware classification, harnessing the collective strengths of two DL models. Our proposed ensemble approach, which incorporates fine-tuned versions of MobileNetV3, and Xception, demonstrates exceptional performance in enhancing detection accuracy and resilience. By combining the predictive capabilities of these models using a stack ensemble framework, we achieve superior results compared to individual models. The stacking technique employed in our method ensures high generalizability and low variance in the classification outcomes. An important aspect of our approach is the elimination of traditional feature engineering and domain-specific techniques, simplifying the classification process and making it more adaptable to dynamic malware landscapes. This progress underscores a major evolution in detection strategies, providing a foundation for scalable and high-precision security solutions in the IoT domain. The evaluation of our proposed ensemble model on the Malimg dataset, comprising 9335 images across 25 distinct classes, showcases its remarkable accuracy rate of 98.75% and on the MaleVis dataset with 14,225 images across 26 distinct classes with the accuracy rate of 98.55%, respectively. Furthermore, statistical analyses, including McNemar's test, validate the significance of our results and underscore the reliability of our approach. Comparative evaluations with existing methods underscore the efficacy of our proposed ensemble approach in malware detection, positioning it as a promising solution for enhancing digital security in diverse environments. In summary, our study contributes valuable insights into the effectiveness of ensemble methods for malware classification and underscores their pivotal role in fortifying digital security infrastructure against evolving cyber threats.

Acknowledgement: The authors gratefully acknowledge the support of Princess Nourah bint Abdulrahman University. The authors would also like to thank Prince Sultan University for their support.

Funding Statement: This work was funded and supported by the Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2026R757), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia. The authors would also like to acknowledge the support of Prince Sultan University.

Author Contributions: The authors confirm contribution to the paper as follows: Study conception and design, collection, analysis and interpretation of results, draft manuscript preparation: Muhammad Shaheer, Feng Zeng and Muhammad Asim. Review, editing, and supervision paper: Feng Zeng, Aqsa Yasmeen, Muhammad Asim, Mudasir Ahmad Wani and Kashish Ara Shakil. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The data that support the findings of this study are available from the corresponding author upon reasonable request.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Su J, Vasconcellos DV, Prasad S, Sgandurra D, Feng Y, Sakurai K. Lightweight classification of IoT malware based on image recognition. In: 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC); 2018 Jul 23–27; Tokyo, Japan. p. 664–9. doi:10.1109/COMPSAC.2018.10315.

2. Alzahem A, Boulila W, Driss M, Koubaa A, Almomani I. Towards optimizing malware detection: an approach based on generative adversarial networks and transformers. In: Computational collective intelligence. Cham, Switzerland: Springer International Publishing; 2022. p. 598–610. doi:10.1007/978-3-031-16014-1_47.
3. Ding H, Sun Y, Huang N, Shen Z, Cui X. TMG-GAN: generative adversarial networks-based imbalanced learning for network intrusion detection. IEEE Trans Inf Forensics Secur. 2024;19(27):1156–67. doi:10.1109/TIFS.2023.3331240.
4. Khan AR, Yasin A, Usman SM, Hussain S, Khalid S, Ullah SS. Exploring lightweight deep learning solution for malware detection in IoT constraint environment. Electronics. 2022;11(24):4147. doi:10.3390/electronics11244147.
5. Fiermonte M. The threat of social engineering to networked systems [master's thesis]. Utica, NY, USA: Utica College; 2019.
6. Pai V, Karthik Pai BH, Sudhiksha GS, Kamath V, Varsha K, Manjunatha S. Systematic approach for malware detection in IoT devices: enhancing security and performance. Int J Comput Intell Syst. 2025;18(1):196. doi:10.1007/s44196-025-00939-9.
7. Shone N, Ngoc TN, Phai VD, Shi Q. A deep learning approach to network intrusion detection. IEEE Trans Emerg Top Comput Intell. 2018;2(1):41–50. doi:10.1109/tetci.2017.2772792.
8. Khan SH, Alahmadi TJ, Ullah W, Iqbal J, Rahim A, Alkahtani HK, et al. A new deep boosted CNN and ensemble learning based IoT malware detection. Comput Secur. 2023;133:103385. doi:10.1016/j.cose.2023.103385.
9. Dong H, Kotenko I. Image analysis approach based on VAE-GAN for IoT malware detection in urban environments. Innov Syst Softw Eng. 2025;21(2):381–95. doi:10.1007/s11334-025-00610-8.
10. Javed S, Wu G, Javed H. EffiMalNetv3: a lightweight framework for malware detection using contrastive learning and attention-guided deep learning approach. Appl Soft Comput. 2026;196:114981. doi:10.1016/j.asoc.2026.114981.
11. Zhuang F, Qi Z, Duan K, Xi D, Zhu Y, Zhu H, et al. A comprehensive survey on transfer learning. Proc IEEE. 2021;109(1):43–76. doi:10.1109/jproc.2020.3004555.
12. Smmarwar SK, Gupta GP, Kumar S. Android malware detection and identification frameworks by leveraging the machine and deep learning techniques: a comprehensive review. Telemat Inform Rep. 2024;14:100130. doi:10.1016/j.teler.2024.100130.
13. Kumar R, Zhang X, Khan RU, Ahad I, Kumar J. Malicious code detection based on image processing using deep learning. In: Proceedings of the 2018 International Conference on Computing and Artificial Intelligence; 2018 Mar 12–14; Chengdu, China. p. 81–5. doi:10.1145/3194452.3194459.
14. Schultz MG, Eskin E, Zadok F, Stolfo SJ. Data mining methods for detection of new malicious executables. In: Proceedings 2001 IEEE Symposium on Security and Privacy, S&P 2001; 2000 May 14–16; Oakland, CA, USA. p. 38–49. doi:10.1109/SECPRI.2001.924286.
15. Deng H, Guo C, Shen G, Cui Y, Ping Y. MCTVD: a malware classification method based on three-channel visualization and deep learning. Comput Secur. 2023;126(7):103084. doi:10.1016/j.cose.2022.103084.
16. D'Angelo G, Ficco M, Palmieri F. Association rule-based malware classification using common subsequences of API calls. Appl Soft Comput. 2021;105(5):107234. doi:10.1016/j.asoc.2021.107234.
17. Naeem H, Guo B, Naeem MR. A light-weight malware static visual analysis for IoT infrastructure. In: 2018 International Conference on Artificial Intelligence and Big Data (ICAIBD); 2018 May 26–28; Chengdu, China. p. 240–4. doi:10.1109/ICAIBD.2018.8396202.
18. Qaisar ZH, Faheem M, Ashraf MW. AMALGAN: image-based Android malware classification using generative adversarial network. J Eng. 2025;2025(1):e70146. doi:10.1049/tje2.70146.
19. Vishwakarma M, Kesswani N. DIDS: a deep neural network based real-time intrusion detection system for IoT. Decis Anal J. 2022;5(1):100142. doi:10.1016/j.dajour.2022.100142.
20. Saleh IF. Enhanced malware detection for IoT networks utilizing a 2D-CNN with data augmentation on the malimg dataset. J Al Qadisiyah Comput Sci Math. 2025;17(1):179–91. doi:10.29304/jqcsm.2025.17.11973.
21. Smmarwar SK, Priyadarshi R, Angaitkar P, Mishra S, Rathore RS. AIMD: AI-powered android malware detection for securing AIoT devices and networks using graph embedding and ensemble learning. J Syst Archit. 2026;173(5):103707. doi:10.1016/j.sysarc.2026.103707.

22. Kumar S, Janet B, Neelakantan S. IMCNN: intelligent malware classification using deep convolution neural networks as transfer learning and ensemble learning in honeypot enabled organizational network. Comput Commun. 2024;216(1):16–33. doi:10.1016/j.comcom.2023.12.036.
23. Kasongo SM, Sun Y. A deep learning method with wrapper based feature extraction for wireless intrusion detection system. Comput Secur. 2020;92(1):101752. doi:10.1016/j.cose.2020.101752.
24. Mohammed TM, Nataraj L, Chikkagoudar S, Chandrasekaran S, Manjunath BS. Malware detection using frequency domain-based image visualization and deep learning. arXiv:2101.10578. 2021.
25. Awan MJ, Masood OA, Abed Mohammed M, Yasin A, Zain AM, Damaševičius R, et al. Image-based malware classification using VGG19 network and spatial convolutional attention. Electronics. 2021;10(19):2444. doi:10.3390/electronics10192444.
26. Ben Jabra M, Cheikhrouhou O, Atitallah N, Ben Amor A, Hamam H. Malware detection using deep learning and CNN models. In: 2023 International Conference on Cyberworlds (CW); 2023 Oct 3–5; Sousse, Tunisia. p. 432–9. doi:10.1109/cw58918.2023.00073.
27. Javed S, Wu G, Javed H, Khashan OA, Hassan H, Ghani A. An automated multi-scale and multi-contextual MobileNetv3 for malware detection based on IoT. Array. 2026;29(1):100681. doi:10.1016/j.array.2026.100681.
28. Kumari M, Hsieh G, Okonkwo CA. Deep learning approach to malware multi-class classification using image processing techniques. In: 2017 International Conference on Computational Science and Computational Intelligence (CSCI); 2017 Dec 14–16; Las Vegas, NV, USA. p. 13–8. doi:10.1109/CSCI.2017.3.
29. Omran musa H, Tahrir Younis M. Image-based malware detection using deep CNN models. Iraqi J Comput Inform. 2025;51(1):64–74. doi:10.25195/ijci.v51i1.542.
30. Arse M, Sharma K, Bindewari S, Tomar A, Patil H, Jha N. Mitigating malware attacks using machine learning: a review. In: 2023 International Conference on Artificial Intelligence and Smart Communication (AISC); 2023 Jan 27–29; Greater Noida, India. p. 1032–8. doi:10.1109/AISC56616.2023.10085630.
31. Al-Khater W, Al-Madeed S. Using 3D-VGG-16 and 3D-Resnet-18 deep learning models and FABEMD techniques in the detection of malware. Alex Eng J. 2024;89(1):39–52. doi:10.1016/j.aej.2023.12.061.
32. Howard A, Sandler M, Chen B, Wang W, Chen LC, Tan M, et al. Searching for MobileNetV3. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV); 2019 Oct 27–Nov 2; Seoul, Republic of Korea. p. 1314–24. doi:10.1109/iccv.2019.00140.
33. Chollet F. Xception: deep learning with depthwise separable convolutions. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 2017 Jul 21–26; Honolulu, HI, USA. p. 1800–7. doi:10.1109/CVPR.2017.195.
34. Rane N, Choudhary SP, Rane J. Ensemble deep learning and machine learning: applications, opportunities, challenges, and future directions. Stud Med Health Sci. 2024;1(2):18–41. doi:10.48185/smhs.v1i2.1225.
35. Dietterich TG. Approximate statistical tests for comparing supervised classification learning algorithms. Neural Comput. 1998;10(7):1895–923. doi:10.1162/089976698300017197.