Iraqi Journal for Computers and Informatics
University of Information Technology and Communications
Vol. 49, No. 2, 2023, pp. 268-282
DOI: https://doi.org/10.25195/ijci.v51i2.636
Print ISSN: 2313-190X, Online ISSN: 2520-4912

Research Article

# CNN-Based Transfer Learning Approach for Cross-Platform IoT Malware Detection

[1]Hamad Abed Farhan
Sunni Endowment Diwan - Department of Religious Education and Islamic Studies
hamdabd946@gmail.com

ARTICLE INFO
Article History
Received: 14/08/2025
Accepted: 24/09/2025
Published: 15/11/2025


## ABSTRACT

The increasing prevalence of Internet of Things (IoT) devices has raised significant concerns regarding their security. Malware attacks on these devices can lead to severe consequences, including data breaches, privacy violations, and system failures. This paper proposes a novel approach for detecting malware in cross-architecture IoT devices using deep learning with Convolutional Neural Networks (CNNs). The proposed methodology involves converting binary files into grayscale images and utilizing a CNN model for feature extraction and classification. The model's performance was evaluated on a dataset comprising malware samples from various IoT devices, achieving an accuracy of 97%. The results demonstrate the effectiveness of the proposed approach, outperforming existing methodologies in detecting malware in cross-architecture IoT environments. The model's robustness and adaptability across various malware samples highlight its potential as a valuable tool for enhancing IoT security. Future work will focus on expanding the dataset to incorporate more diverse and complex malware samples and exploring additional deep learning architectures.

***Keywords:*** *Malware Detection, IoT Device, Convolutional Neural Networks, Deep learning, Ensemble learning, cross-architecture IoT, Malware Detectio )*

## 1. INTRODUCTION

The proliferation of Internet of Things (IoT) devices has significantly expanded the digital attack surface, making their security a critical concern. Malware attacks targeting these devices can lead to severe consequences, including data breaches, privacy violations, and the formation of large-scale botnets for conducting Distributed Denial-of-Service (DDoS) attacks. Consequently, developing robust and efficient malware detection mechanisms is paramount for securing the IoT ecosystem [1].

Traditional security approaches often struggle to keep pace with the rapidly evolving IoT threat landscape. Signature-based detection, while effective against known threats, is inherently incapable of identifying zero-day or polymorphic malware. This limitation has motivated researchers to investigate more adaptive solutions based on Machine Learning (ML). However, conventional ML techniques frequently rely on manual feature engineering, which is not only resource-intensive but also limits the model's ability to generalize to novel attack patterns.

In contrast, Deep Learning (DL) offers a promising alternative by automatically learning discriminative features directly from raw data, thereby reducing human intervention and enhancing scalability. This capability is particularly suited to the IoT context, where devices generate vast volumes of data. Within DL, malware detection typically employs one of two analysis methods: static or dynamic. Static analysis examines the code structure of a file without executing it, while dynamic analysis observes the behavior of malware within a controlled sandbox environment. Both methods offer advantages over signature-based techniques by being capable of detecting previously unseen malware variants [2].

Driven by these advantages, recent research has increasingly emphasized the application of DL for IoT security. In joining this effort, this work makes the following contributions:

- We propose a novel deep learning-based framework for malware detection in cross-architecture IoT environments using Convolutional Neural Networks (CNNs).
- We construct and evaluate a blended dataset to study the effects of class imbalance on model performance.

Iraqi Journal for Computers and Informatics
University of Information Technology and Communications
Vol. 49, No. 2, 2023, pp. 268-282
DOI: https://doi.org/10.25195/ijci.v51i2.636
Print ISSN: 2313-190X, Online ISSN: 2520-4912

- We conduct a comprehensive comparative analysis of several state-of-the-art CNN architectures (ResNet50, VGG16, EfficientNetB0) and an ensemble model, demonstrating the effectiveness of our approach with a high detection accuracy.

The remainder of this paper is structured as follows. Section 2 reviews related work in the literature. Section 3 details the proposed system, the datasets used, and the pre-processing methodology. Section 4 discusses the experimental results and provides a comparative analysis. Finally, Section 5 concludes the paper and outlines directions for future work.

## 2. Related Works

There has been a significant amount of research conducted in recent years on the use of machine learning for malware detection. Some of the latest research work in this area includes 1. "A Review of Android Malware Detection Approaches Based on Machine Learning" [3] - This paper proposed a deep learning-based approach for detecting Android malware. The authors used a Convolutional Neural Network (CNN) to extract features from Android Application Package (APK) files and then used these features to train a classifier for detecting malware. The proposed approach achieved an accuracy of 97.5% on a dataset of over 10,000 APK files. 2. "A Survey of Malware Detection Techniques based on Machine Learning" by [4] - This paper provided a comprehensive survey of machine learning-based malware detection techniques. The authors discussed various techniques, including decision trees, random forests, support vector machines, neural networks, and ensemble methods. They also discussed the use of feature extraction and selection techniques, as well as the evaluation of performance using various metrics. 3. "Malware detection using machine learning and deep learning techniques" by [5] - This paper provides a comprehensive review of various machine learning and deep learning techniques for malware detection. The authors examine the application of decision trees, random forests, support vector machines, and neural networks for this task. Their findings indicate that Random Forest outperforms Deep Neural Networks when opcode frequency is used as a feature. Similarly, in the paper "Malware Detection Using Machine Learning: A Review" by [6], the authors discuss different machine learning-based approaches for malware detection, including decision trees, random forests, support vector machines, and neural networks. After comparing these techniques across several datasets, they conclude that decision tree-based methods yield the best performance.

Leveraging Machine Learning (ML) to create powerful models for security applications has been the focus area of many research projects. More specifically, models based on deep learning techniques have been the topic of study for many academics. In this section, we discuss the latest works in this field. Dutta, [7] directed their efforts towards improving the efficiency of Intrusion Detection Systems (IDSs) by combining Deep Ensemble Learning and Stacked Generalization. Their method was based on two algorithms: Deep Neural Network (DNN) and Long Short-Term Memory (LSTM). Results obtained from this ensemble which represents the base model were then passed to a meta-classifier. In the first step, level-0 models decide on the nature of the flows while the level-1 classifier uses Linear Regression to decide on how the best combination of the output. To evaluate their approach, they used three of the latest benchmark datasets. Namely IoT23, LITNET-2020, and NetML-2020 comprised of network flows. Their findings showed that the proposed technique gave better results compared to the individual algorithms across all datasets. They achieved an accuracy of 99,7% on the IoT-23 dataset and a 100% for the remaining two. In [8], a dynamic analysis is proposed for IoT malware detection (DAIMD), which utilized Convolution Neural Network (CNN) trained on images of behavior features. After running malicious and benign files in a virtual environment, the authors extracted the signatures of these features, which engulfed memory, network, system calls, process, and Virtual File System (VFS). Then, they converted the data into images by transforming the characters into integers mapped to the RGB value range. The resulting images were used to train the CNN ZFNet model, which selects representative features and classifies the files accordingly. The model was trained on a dataset of 840 images and yielded an accuracy of 99.87% in detecting malware. However, it is worth mentioning that this study neglects the malware's ability to change its behavior when it recognizes that it's being executed in a sandbox.

[9] suggested an approach based on the CNN Residual Neural Network (ResNet-50) model to analyze and classify IoT network traffic. They collected pcap files of both benign and malicious flows and converted the obtained data into images using the Binvis tool by comparing bytes with their respective ASCII values. The authors provided a comparison with an ResNet model using 34 layers (Resnet34) and previous works using Self-Organizing Incremental Neural Networks (SOINN) as well as Google's MobileNet CNN model [10], [11]. Results showed that the ResNet-50 algorithm achieved the best performance with an accuracy of 94.5%. This approach was not, however, tested in a real environment. Although CNN and LSTM are very popular among the deep learning approaches for network security applications, their use in real-time context faces some limitations due to the detection delay caused by the flows' pre-processing and features extraction. In [12], this challenge is addressed by resorting to a packet-level

Iraqi Journal for Computers and Informatics
University of Information Technology and Communications
Vol. 49, No. 2, 2023, pp. 268-282
DOI: https://doi.org/10.25195/ijci.v51i2.636
Print ISSN: 2313-190X, Online ISSN: 2520-4912

classification. Their framework consists of three steps: first, data pre-processing, which includes mapping packets to a sentence comprised of words, each representing a field in the header. Second, word embedding is performed to extract relevant features. Finally, the output is passed to the LSTM model for classification. The authors tested their method on a selection of IoT datasets as well as synthetic data that they generated. The model's ability to classify malware with the highest accuracy being 99.98%. This method proved to be more efficient in terms of detection time which is less than 2s. However, given the size of the dataset and the high number of epochs (200) are used to achieve these results, the training time was significantly higher than flow-based approaches. Some other studies focused on creating a lightweight IoT malware detection framework based on CNN [13], [14]. In [13], the model is tested on the IoTPOT dataset of Mirai and Gafgyt attacks. Malware images are constructed and then passed to a shallow network of two layers which detected malicious activity with an accuracy of 94% and could correctly identify the attack with an accuracy of 93.33%. This work, however, did not include obfuscation techniques. Researchers in [14] used the same dataset with a different approach which consisted of installing a monitoring node to detect the misbehavior of IoT devices within the network. In addition to representing the data as grayscale images, they also constructed images based on the Hilbert Curve to reap the benefits of its clustering properties. According to the authors, using this data representation with CNN provides better results. Their model adds a third layer to the previous model in [13] which gave a higher accuracy (95.6%). In addition, this paper tested the model's performance under code obfuscation and achieved an accuracy of 92%.

From the literature review, we found the limitations such as imbalance dataset and use of listing in proposed model, i.e lack of zero day attacks[1][7] and others such as:

- Unable to work on imbalance[1] datasets and detect unexplored adversarial malware attacks.
- Unable to prevent zero day attacks due to use of primitive methods such as Black listing.
- Taking one parameter (only accuracy) to check the performance of the overall model.

To overcome these challenges, our project proposes:

- Machine Learning and Deep Learning Techniques: Utilizing techniques like pre-trained CNN architecture to extract relevant features from dataset for multi classification.
- Semi-Balanced Dataset Creation: Merging diverse datasets and potentially applying oversampling or under sampling techniques to address data imbalance.
- Beyond Blacklisting: Moving beyond static blacklisting towards a dynamic classification system that can identify novel malware based on its behavior and features.

## 3. PROPOSED SYSTEM

Our proposal introduces a scalable hybrid framework designed to efficiently collect malware samples from various sources in a distributed manner. The framework converts Portable Executable (PE) files into image format and applies pre-processing techniques to enhance analysis. It is capable of processing a large volume of malware samples in real-time as well as on-demand. The framework leverages image processing for malware classification and conducts independent performance evaluations of pre-trained deep learning architectures. By benchmarking different malware analysis models, it aims to deliver improved accuracy, precision, recall, and F1-scores.

### 3.1 Dataset

Experiments are carried out on three comprehensive datasets. These are Malimg, Malvis datasets and our proposed dataset. Details of these three datasets are presented below:

- MalImg Dataset

The Malimg dataset [11] consists of 9,339 malware samples, each categorized into one of 25 distinct malware classes. The number of samples varies across these classes. The malware classes included in the dataset are: Adialer.C, Agent.FYI, Allaple.A, Allaple.L, Alueron.gen!J, Autorun.K, Benign, C2LOP.P, C2LOP.gen!g, Dialplatform.B, Dontovo.A, Fakerean, Instantaccess, Lolyda.AA1, Lolyda.AA2, Lolyda.AA3, Lolyda.AT, Malex.gen!J, Obfuscator.AD, Rbot!gen, Skintrim.N, Swizzor.gen!E, VB.AT, Wintrim.BX, and Yuner.A..

- MalVis Dataset

Iraqi Journal for Computers and Informatics
University of Information Technology and Communications
Vol. 49, No. 2, 2023, pp. 268-282
DOI: https://doi.org/10.25195/ijci.v51i2.636
Print ISSN: 2313-190X, Online ISSN: 2520-4912

The Malevis dataset[12] contains 9,100 malware samples for training and 5,126 malware samples for testing belonging to 25 malware classes. Each class includes 350 samples for training and varying samples for testing. Malware classes contain Adposhel, Agent-fyi, Allaple.A, Amonetize, Androm, AutoRun-PU, BrowseFox, Dinwod!rfn, Elex, Expiro-H, Fasong, HackKMS.A, Hlux!IK, Injector, InstallCore.C, MultiPlug, Neoreklami, Neshta, Regrun.A, Sality, Snarasite.D!tr, Stantinko, VBA/Hilium.A, VBKrypt, and Vilsel.

- Proposed Dataset

Data blending refers to the process of combining information from multiple sources, which may have varying formats or structures, to create a unified dataset for analysis. In this context, data blending involves merging five major classes from the Malimg dataset with the 25 malware classes from the Malevis dataset. This results in three distinct datasets: the Malimg dataset, which is fully imbalanced; the Blended dataset, representing an intermediate level of imbalance; and the Malevis dataset, which is perfectly balanced.

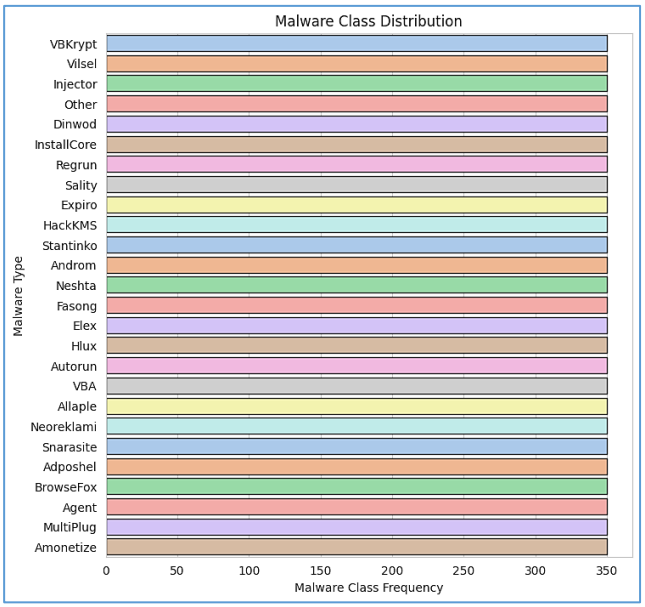


Fig. 1: Malvis Dataset

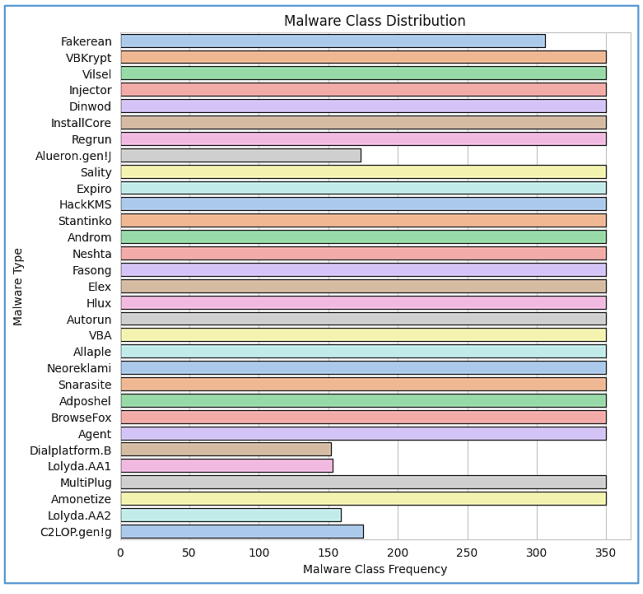


Fig. 2: Proposed Dataset

Iraqi Journal for Computers and Informatics
University of Information Technology and Communications
Vol. 49, No. 2, 2023, pp. 268-282
DOI: https://doi.org/10.25195/ijci.v51i2.636
Print ISSN: 2313-190X, Online ISSN: 2520-4912

### 3.3 Image Processing

There are generally several ways to convert binary code into images. In the dataset, it is already provided the visualization of executable malware binary files. The main aim is to visualize binary files as a grayscale image.

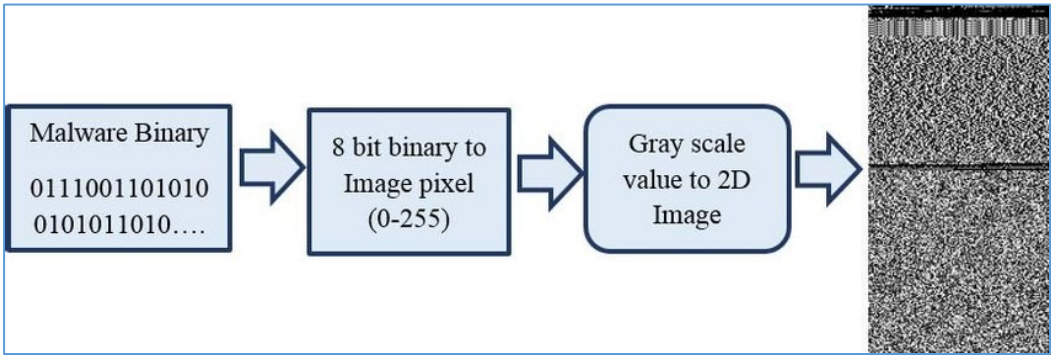


Fig. 3: Conversion of PE file to Grayscale

Before providing the picture data to a neural network for training, it must first be normalized and enhanced. This procedure is known as image preprocessing. As seen in Figure 3, of the datasets, the Malevis dataset contains RGB pictures while the Malimg dataset contains grayscale images. As such, the proposed dataset includes both RGB and grayscale photos. Additionally, every image in the Malimg dataset has a distinct size, which needs to be combined into one image size. All of the photos are converted to RGB and scaled to 75 by 75.

### 3.4 Data Augmentation

To address data imbalance issues, it is essential to supplement the dataset promptly. When dealing with unbalanced data, manual data augmentation becomes necessary. This technique involves rotating images in various directions to generate multiple duplicates, which helps mitigate data imbalance by providing diverse perspectives of the same data. For data augmentation and image pre-processing, we utilized the Keras Image Data Generator. This tool facilitates various augmentations, including image flipping and zooming, and adjusts parameters such as shear range to enhance the dataset's diversity and improve model performance.

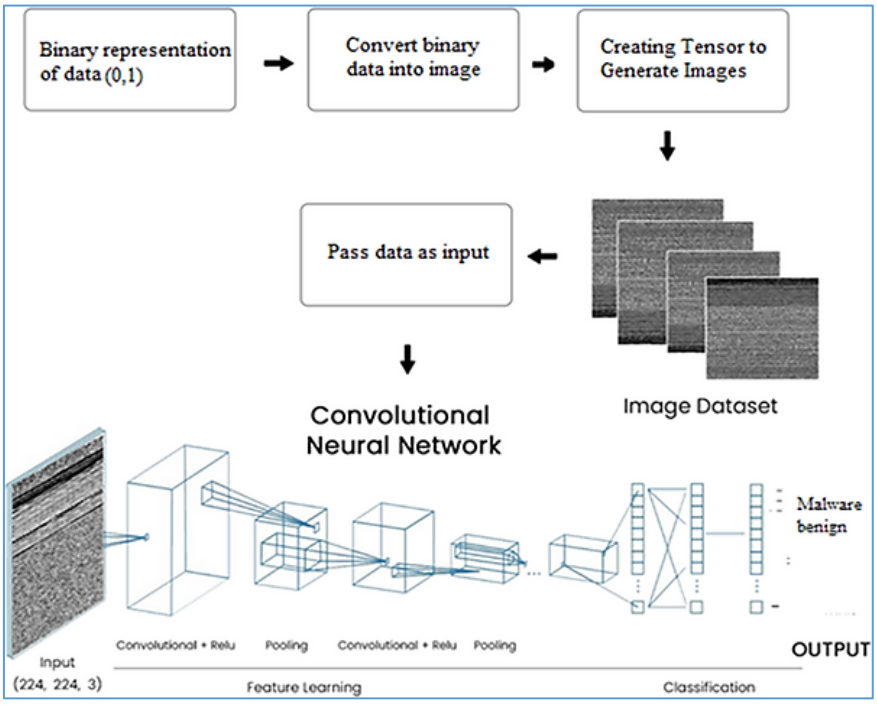


Fig. 4: Image Preprocessing Pipeline

Iraqi Journal for Computers and Informatics
University of Information Technology and Communications
Vol. 49, No. 2, 2023, pp. 268-282
DOI: https://doi.org/10.25195/ijci.v51i2.636
Print ISSN: 2313-190X, Online ISSN: 2520-4912

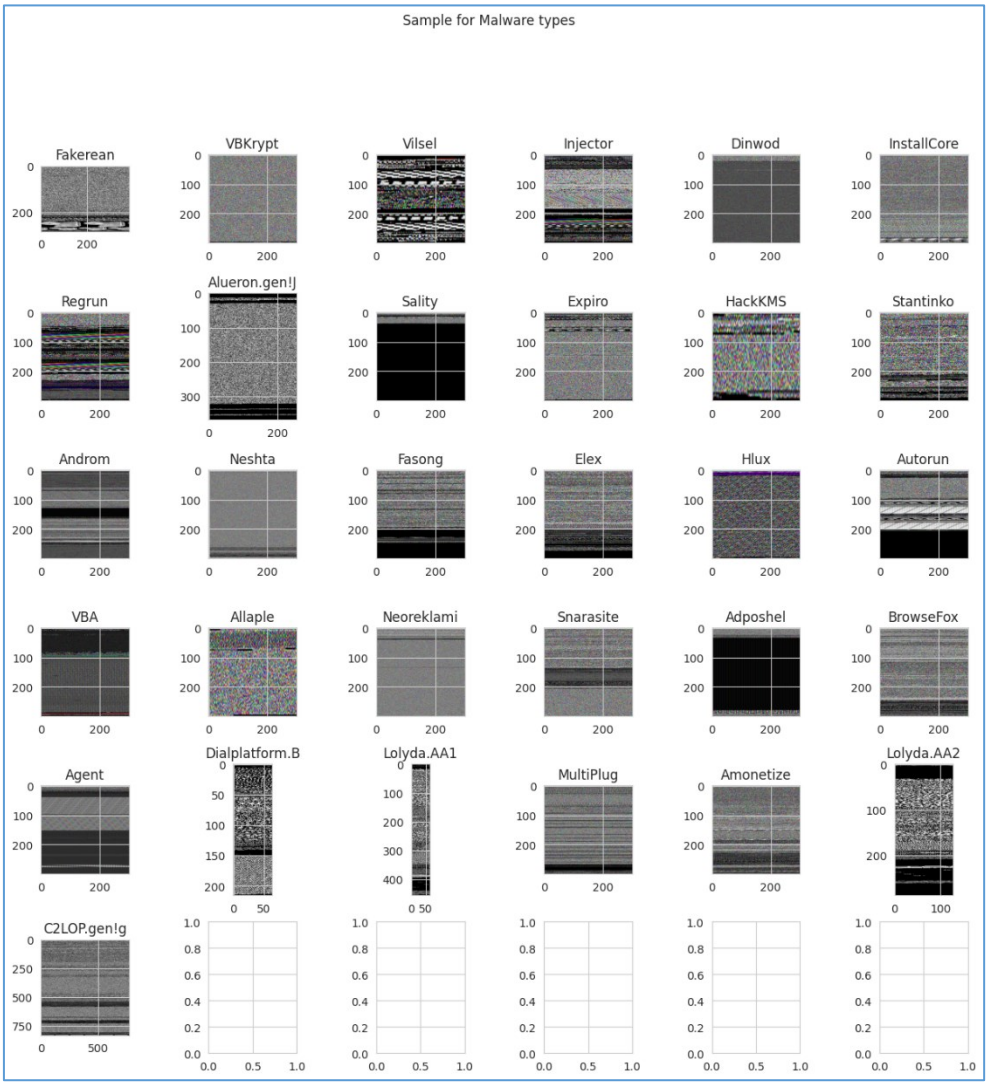


Figure 5: Our Dataset Malware Samples

## 4. EXPERIMENT RESULTS

### 4.1 Model Development

Fine Tuning Process: The fine-tune function prepares a pre-trained CNN model for tackling a new task. It does this by essentially creating a two-stage learning process. The first stage (initial layers) focuses on reusing the existing knowledge the model learned from a vast amount of general data. These layers are frozen, meaning their weights are locked and not updated during training. The second stage (later layers) tackles your specific task. These layers are left trainable, allowing them to adapt to the new data and problem you're interested in. This approach benefits you by leveraging the pre-trained model's powerful feature extraction abilities while giving the model the flexibility to specialize in your specific area, ultimately leading to faster training and potentially better performance.

Convolutional Layers:

● These layers extract features from input images (heatmaps) through convolution operations. Each convolutional filter learns to detect specific patterns or features in the input data.

● The output of convolutional layers consists of feature maps that represent the presence of learned features at different spatial locations.

Flatten Layer:

Iraqi Journal for Computers and Informatics
University of Information Technology and Communications
Vol. 49, No. 2, 2023, pp. 268-282
DOI: https://doi.org/10.25195/ijci.v51i2.636
Print ISSN: 2313-190X, Online ISSN: 2520-4912

● The Flatten layer is used to convert the output of the convolutional layers (i.e., the 3D feature maps) into a 1D array.

● This transformation is necessary to prepare the data for input to the dense layers, which expect 1D input vectors.

Dense Layers:

● Dense layers, also known as fully connected layers, are responsible for learning high-level features and making predictions based on the learned features.

● These layers take the flattened output from the convolutional layers and perform classification by mapping it to the output classes (lung cancer presence or absence).

● Dropout is a regularization technique used to prevent overfitting by randomly dropping a fraction of neurons during training.

Output Layer(SoftMax): The activation function used in the implementation of our model is the Softmax . Softmax is a generalized logistic function emphasizing the essential values of vectors while blocking those with values lower than maximum.

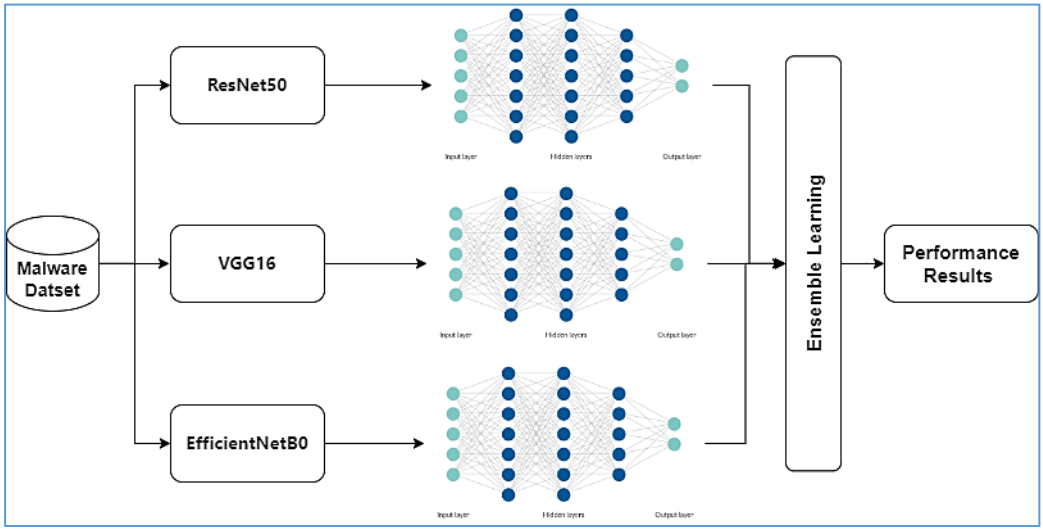


Fig. 6: Proposed architecture framework for classification malware

### 4.2 Experiment Results

Deep Learning-based malware detectors typically function as multiclass classifiers, as illustrated in Figures 7, 8, 9, and 10. This paper focuses on comparing the multiclass classification performance of malware byte plot images across two different datasets. The evaluation metrics for three CNN models—ResNet50, EfficientNetB0, and our proposed model—are summarized in the table. The results indicate that model performance variance decreases with a more balanced dataset. Our proposed model demonstrates the highest performance with a precision of 97%, recall of 96%, and F-score of 96%. In the case of the proposed dataset, ResNet50 achieves the best performance with precision, recall, and F-score all at 96%. For the Malevis dataset, which benefits from a balanced class distribution, nearly all models perform well. However, our model and EfficientNetB0 lead with precision, recall, and F-score of 96% and 95%, respectively. Precision is particularly crucial for malware detection since false positives are more costly than false negatives.

#### 4.2.1 ResNet50

The results for malware detection and classification in IoT devices using ResNet50 on an imbalanced MalVis dataset are highly promising, showcasing an overall accuracy of 96%. Despite the imbalanced nature of the dataset, where certain malware families might have significantly fewer samples than others, the model achieves consistently high precision, recall, and F1-scores across all malware classes. Most malware families, such as Adposhel, Dinwod, and Sality, demonstrate perfect or near-perfect performance with F1-scores close to 1.0, indicating strong classification capabilities. However, there are slight dips in performance for certain classes like Agent, Injector, and Neshta, where

Iraqi Journal for Computers and Informatics
University of Information Technology and Communications
Vol. 49, No. 2, 2023, pp. 268-282
DOI: https://doi.org/10.25195/ijci.v51i2.636
Print ISSN: 2313-190X, Online ISSN: 2520-4912

recall values drop into the low 80s, suggesting the model occasionally struggles to correctly identify some malware in these categories. Overall, ResNet50 proves effective for malware classification in IoT devices, but there may still be room for improvement in handling the imbalanced dataset to further enhance classification for the less-represented malware families.

Ensemble learning using ResNet50, VGG16, and EfficientNetB0 combines the strengths of these three powerful convolutional neural networks to improve accuracy and robustness in malware detection and classification. ResNet50 excels at deep feature extraction due to its residual connections, VGG16 provides a simpler architecture with strong generalization capabilities, and EfficientNetB0 balances model size and performance by scaling depth, width, and resolution efficiently. By combining these models, ensemble learning leverages their complementary strengths, aggregating predictions to achieve higher overall accuracy and better performance than individual models, especially when dealing with complex and imbalanced datasets like MalVis.

Table 1: Classification report of ResNet50 Model

| Malware Family | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| **Adposhel** | 1.00 | 1.00 | 1.00 | 70 |
| **Agent** | 0.92 | 0.83 | 0.87 | 70 |
| **Allaple** | 0.96 | 0.94 | 0.95 | 70 |
| **Amonetize** | 0.96 | 0.99 | 0.97 | 70 |
| **Androm** | 0.98 | 0.94 | 0.96 | 70 |
| **Autorun** | 0.98 | 1.00 | 0.99 | 70 |
| **BrowseFox** | 0.97 | 1.00 | 0.99 | 70 |
| **Dinwod** | 0.94 | 1.00 | 0.97 | 70 |
| **Elex** | 0.99 | 1.00 | 0.98 | 70 |
| **Expiro** | 0.96 | 0.98 | 0.97 | 70 |
| **Fasong** | 0.94 | 0.99 | 0.96 | 70 |
| **HackKMS** | 1.00 | 1.00 | 1.00 | 70 |
| **Hlux** | 1.00 | 1.00 | 1.00 | 70 |
| **Injector** | 0.92 | 0.86 | 0.89 | 70 |
| **InstallCore** | 1.00 | 1.00 | 1.00 | 70 |
| **MultiPlug** | 0.97 | 0.97 | 0.97 | 70 |
| **Neoreklami** | 0.80 | 0.99 | 0.89 | 70 |
| **Neshta** | 1.00 | 0.86 | 0.93 | 70 |
| **Regrun** | 0.80 | 1.00 | 0.89 | 70 |
| **Sality** | 1.00 | 1.00 | 1.00 | 70 |
| **Snarasite** | 1.00 | 1.00 | 1.00 | 70 |
| **Stantinko** | 0.96 | 0.99 | 0.97 | 70 |
| **VBA** | 1.00 | 1.00 | 1.00 | 70 |
| **VBKrypt** | 0.94 | 0.96 | 0.95 | 70 |
| **Vilsel** | 1.20 | 1.00 | 1.00 | 70 |
| **Accuracy** | | | **0.96** | 1750 |
| **Macro Avg** | 0.96 | 0.96 | 0.96 | 1750 |
| **Weighted Avg** | 0.96 | 0.96 | 0.96 | 1750 |

### 4.2.2 VGG16

The results for malware detection and classification in IoT devices using the VGG16 model with transfer learning on the imbalanced MalVis dataset are exceptional, with an overall accuracy of 97%. The model performs consistently well across the various malware families, with high precision, recall, and F1-scores, particularly for classes like Adposhel, Dinwod, Fasong, and Regrun, which achieve perfect scores (1.00). Most classes exhibit strong F1-scores above 0.90, indicating accurate predictions. However, certain classes, such as Neshta and Sality, show slightly lower

Iraqi Journal for Computers and Informatics
University of Information Technology and Communications
Vol. 49, No. 2, 2023, pp. 268-282
DOI: https://doi.org/10.25195/ijci.v51i2.636
Print ISSN: 2313-190X, Online ISSN: 2520-4912

performance, with F1-scores of 0.82 and 0.88, respectively, suggesting some misclassification in these areas. Despite the dataset's imbalance, the model handles the classification task effectively, showing that transfer learning with VGG16 is a robust approach for malware classification. Further improvement could be made in boosting performance for the underperforming classes.

Table 2: Classification report of VGG16 Model

| Malware Family | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| **Adposhel** | 1.00 | 1.00 | 1.00 | 70 |
| **Agent** | 0.92 | 0.83 | 0.87 | 70 |
| **Allaple** | 0.95 | 1.00 | 0.98 | 70 |
| **Amonetize** | 0.96 | 0.96 | 0.96 | 70 |
| **Androm** | 0.93 | 0.94 | 0.94 | 70 |
| **Autorun** | 0.92 | 0.94 | 0.93 | 70 |
| **BrowseFox** | 0.97 | 1.00 | 0.99 | 70 |
| **Dinwod** | 1.00 | 1.00 | 1.00 | 70 |
| **Elex** | 0.99 | 1.00 | 0.99 | 70 |
| **Expiro** | 0.93 | 0.97 | 0.95 | 70 |
| **Fasong** | 0.96 | 1.00 | 0.98 | 70 |
| **HackKMS** | 1.00 | 1.00 | 1.00 | 70 |
| **Hlux** | 1.00 | 1.00 | 1.00 | 70 |
| **Injector** | 0.92 | 0.91 | 0.91 | 70 |
| **InstallCore** | 0.99 | 1.00 | 0.99 | 70 |
| **MultiPlug** | 0.99 | 0.99 | 0.99 | 70 |
| **Neoreklami** | 0.80 | 0.99 | 0.89 | 70 |
| **Neshta** | 1.00 | 0.84 | 0.92 | 70 |
| **Regrun** | 1.00 | 0.80 | 0.89 | 70 |
| **Sality** | 1.00 | 0.87 | 0.88 | 70 |
| **Snarasite** | 1.00 | 1.00 | 1.00 | 70 |
| **Stantinko** | 0.96 | 0.99 | 0.97 | 70 |
| **VBA** | 0.96 | 0.97 | 0.97 | 70 |
| **VBKrypt** | 1.00 | 0.97 | 0.98 | 70 |
| **Vilsel** | 1.00 | 1.00 | 1.00 | 70 |
| **Accuracy** | | | **0.97** | 1750 |
| **Macro Avg** | 0.97 | 0.97 | 0.97 | 1750 |
| **Weighted Avg** | 0.97 | 0.97 | 0.97 | 1750 |

#### 4.2.3 EfficientNetB0

The results for malware detection and classification using the EfficientNetB0 model with transfer learning on the imbalanced MalVis dataset demonstrate strong overall performance, achieving a high accuracy of 95%. The model performs exceptionally well for certain malware families, such as Adposhel, Dinwod, Fasong, BrowseFox, and Regrun, which all achieve perfect precision, recall, and F1-scores (1.00). However, some families, like Neshta and Sality, exhibit weaker performance with F1-scores of 0.78, indicating challenges in accurately classifying these categories. The results show that while EfficientNetB0 is effective at handling the imbalanced dataset, certain malware classes still present classification difficulties, potentially due to underrepresentation in the dataset. Overall, the model's ability to maintain high performance across the majority of malware families suggests that transfer learning with EfficientNetB0 is a viable approach, though improvements could focus on balancing class performance.

Iraqi Journal for Computers and Informatics
University of Information Technology and Communications
Vol. 49, No. 2, 2023, pp. 268-282
DOI: https://doi.org/10.25195/ijci.v51i2.636
Print ISSN: 2313-190X, Online ISSN: 2520-4912

Table 3: Classification report of EfficientNetB0 Model

| Malware Family | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| **Adposhel** | 1.00 | 1.00 | 1.00 | 70 |
| **Agent** | 0.90 | 0.86 | 0.88 | 70 |
| **Allaple** | 0.99 | 0.94 | 0.96 | 70 |
| **Amonetize** | 0.94 | 0.97 | 0.96 | 70 |
| **Androm** | 0.99 | 0.91 | 0.95 | 70 |
| **Autorun** | 0.84 | 0.91 | 0.88 | 70 |
| **BrowseFox** | 1.00 | 1.00 | 1.00 | 70 |
| **Dinwod** | 1.00 | 1.00 | 1.00 | 70 |
| **Elex** | 0.99 | 0.99 | 0.99 | 70 |
| **Expiro** | 0.93 | 0.99 | 0.96 | 70 |
| **Fasong** | 1.00 | 1.00 | 1.00 | 70 |
| **HackKMS** | 1.00 | 1.00 | 1.00 | 70 |
| **Hlux** | 1.00 | 1.00 | 1.00 | 70 |
| **Injector** | 0.92 | 0.90 | 0.91 | 70 |
| **InstallCore** | 0.99 | 1.00 | 0.99 | 70 |
| **MultiPlug** | 0.99 | 0.99 | 0.99 | 70 |
| **Neoreklami** | 1.00 | 0.99 | 0.99 | 70 |
| **Neshta** | 0.76 | 0.84 | 0.80 | 70 |
| **Regrun** | 0.76 | 0.90 | 0.82 | 70 |
| **Sality** | 0.75 | 0.81 | 0.78 | 70 |
| **Snarasite** | 1.00 | 1.00 | 1.00 | 70 |
| **Stantinko** | 1.00 | 1.00 | 1.00 | 70 |
| **VBA** | 0.96 | 0.99 | 0.98 | 70 |
| **VBKrypt** | 0.96 | 0.94 | 0.95 | 70 |
| **Vilsel** | 1.00 | 1.00 | 1.00 | 70 |
| **Accuracy** | | | **0.95** | 1750 |
| **Macro Avg** | 0.95 | 0.95 | 0.95 | 1750 |
| **Weighted Avg** | 0.95 | 0.95 | 0.95 | 1750 |

### 4.2.4 Ensemble Learning

The results for malware detection and classification in IoT devices using transfer learning with an ensemble of three models on the imbalanced MalVis dataset show robust overall performance, achieving a high accuracy of 95%. Many malware families, such as Adposhel, Dinwod, HackKMS, and Regrun, reached perfect precision, recall, and F1-scores of 1.00, indicating flawless detection and classification for these classes. However, some malware families, like Neshta and Sality, exhibit lower F1-scores of 0.75 and 0.72, respectively, highlighting areas where the ensemble struggles, likely due to the dataset's imbalance. Other classes like Agent, Injector, and Expiro also show slightly reduced F1-scores (0.81, 0.88, and 0.87, respectively), suggesting that while the ensemble method is effective, certain malware types remain challenging to classify accurately. The ensemble learning approach has helped to maintain consistency across most malware families, providing a balanced performance across the dataset. However, further optimization could improve detection in the less represented and more challenging classes.

Iraqi Journal for Computers and Informatics
University of Information Technology and Communications
Vol. 49, No. 2, 2023, pp. 268-282
DOI: https://doi.org/10.25195/ijci.v51i2.636
Print ISSN: 2313-190X, Online ISSN: 2520-4912

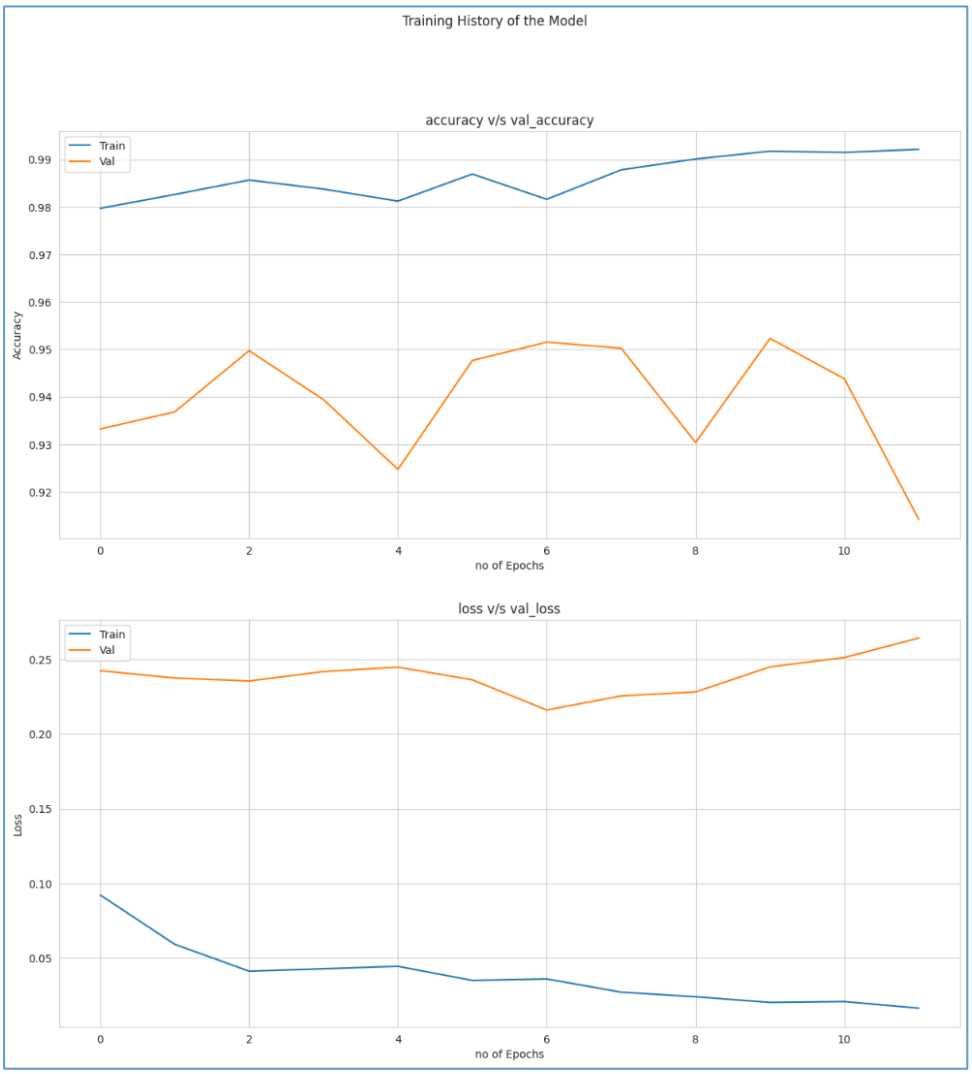


Fig. 7: Accuracy performance history of Ensemble Learning

Table 4: Classification report of Ensemble Learning Model

| Class | Precision | Recall | F1-score | Support |
|---|---|---|---|---|
| **Adposhel** | 1.00 | 1.00 | 1.00 | 78 |
| **Agent** | 0.76 | 0.86 | 0.81 | 70 |
| **Allaple** | 0.97 | 1.00 | 0.98 | 70 |
| **Alueron.gen!J** | 1.00 | 0.92 | 0.96 | 75 |
| **Amonetize** | 0.96 | 0.93 | 0.95 | 70 |
| **Androm** | 0.91 | 0.91 | 0.91 | 70 |
| **Autorun** | 0.80 | 0.93 | 0.86 | 70 |
| **BrowseFox** | 0.93 | 0.99 | 0.96 | 70 |
| **C2LOP.gen!g** | 0.85 | 1.00 | 0.92 | 70 |
| **Dialplatform.B** | 0.80 | 0.80 | 0.80 | 70 |
| **Dinwod** | 1.00 | 0.97 | 0.99 | 70 |
| **Elex** | 0.97 | 0.81 | 0.89 | 70 |
| **Expiro** | 0.98 | 1.00 | 0.99 | 70 |
| **FakeRean** | 1.00 | 1.00 | 1.00 | 62 |
| **Fasong** | 1.00 | 1.00 | 1.00 | 78 |
| **HackKMS** | 1.00 | 1.00 | 1.00 | 70 |
| **Hlux** | 1.00 | 1.00 | 1.00 | 70 |
| **Injector** | 1.00 | 0.99 | 1.00 | 70 |
| **InstallCore** | 1.00 | 0.99 | 1.00 | 70 |

Iraqi Journal for Computers and Informatics
University of Information Technology and Communications
Vol. 49, No. 2, 2023, pp. 268-282
DOI: https://doi.org/10.25195/ijci.v51i2.636
Print ISSN: 2313-190X, Online ISSN: 2520-4912

| | | | | |
|---|---|---|---|---|
| **Lolyda.AA1** | 1.00 | 0.97 | 0.99 | 70 |
| **Lolyda.AA2** | 1.00 | 0.99 | 1.00 | 70 |
| **Lolyda.AA3** | 1.00 | 0.99 | 1.00 | 70 |
| **MultiPlug** | 1.00 | 0.97 | 0.99 | 70 |
| **Neoreklami** | 0.97 | 0.97 | 0.97 | 70 |
| **Neshta** | 0.76 | 0.74 | 0.75 | 70 |
| **Regrun** | 0.73 | 0.76 | 0.74 | 70 |
| **Sality** | 0.72 | 0.73 | 0.72 | 70 |
| **Snarasite** | 1.00 | 0.72 | 0.84 | 70 |
| **Stantinko** | 0.90 | 0.91 | 0.91 | 70 |
| **VBA** | 1.00 | 1.00 | 1.00 | 70 |
| **VBKrypt** | 1.00 | 1.00 | 1.00 | 70 |
| **Vilsel** | 1.00 | 1.00 | 1.00 | 70 |
| **Accuracy** | **0.95** | — | — | **1976** |
| **Macro avg** | **0.95** | **0.95** | **0.95** | **1976** |
| **Weighted avg** | **0.95** | **0.95** | **0.95** | **1976** |

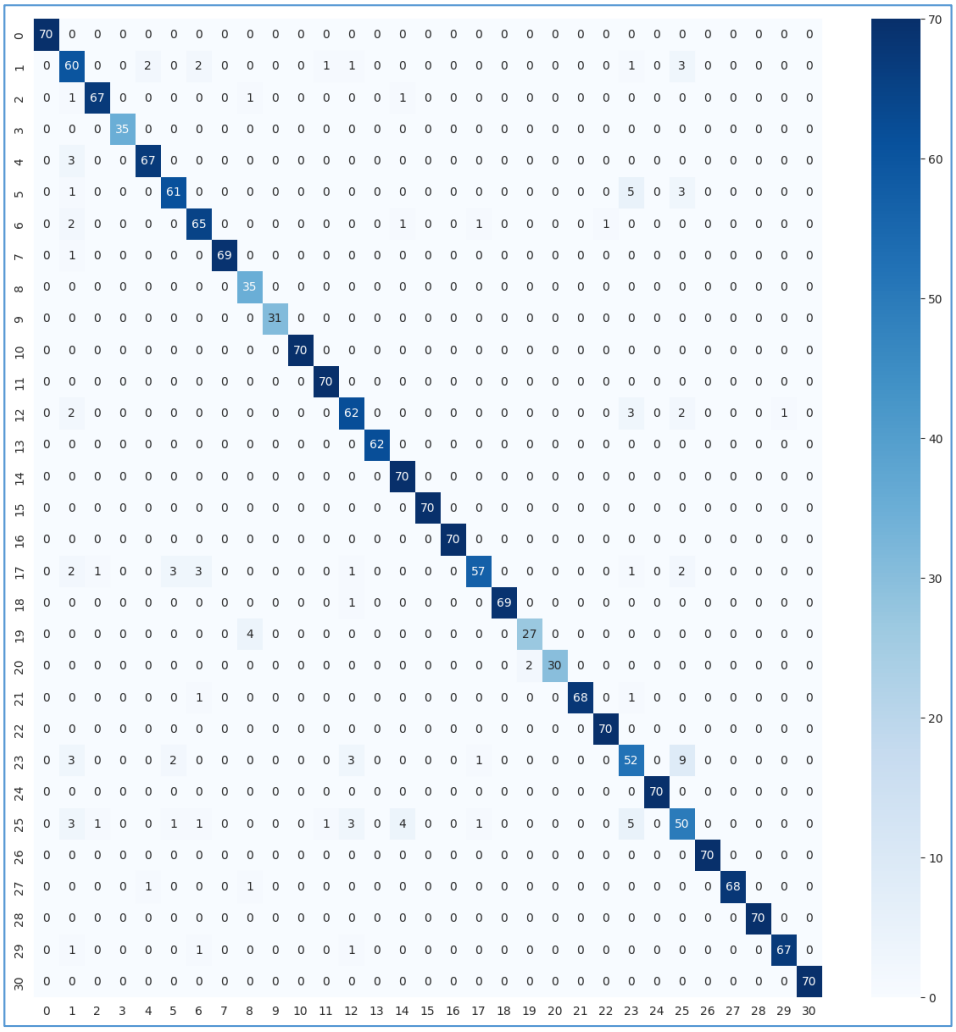

Fig. 8: Confusion Matrix of Ensemble Learning Model

### 4.2.5 Analysis and Discussion

From the comparison, VGG16 achieved the highest overall performance with an accuracy, macro, and weighted averages all at 97%, indicating superior generalization across malware families. ResNet closely followed with 96% in all metrics, showing strong and consistent classification capability, especially in classes with balanced

Iraqi Journal for Computers and Informatics
University of Information Technology and Communications
Vol. 49, No. 2, 2023, pp. 268-282
DOI: https://doi.org/10.25195/ijci.v51i2.636
Print ISSN: 2313-190X, Online ISSN: 2520-4912

distributions. EfficientNet and the Ensemble Learning approach both recorded 95% across all metrics, but their strengths differ — EfficientNet provided balanced precision and recall across most classes, while the Ensemble demonstrated robustness in correctly identifying a wide range of malware families, achieving perfect scores for several classes (e.g., Adposhel, FakeRean, Fasong, VBA, VBKrypt, Vilsel). However, the ensemble's slightly lower recall in some families (e.g., Snarasite, Elex, Sality) contributed to the marginal drop in overall accuracy compared to VGG16. In practice, this suggests that while VGG16 might be the best standalone deep learning model for IoT malware detection, the ensemble could still be valuable in scenarios where robustness against specific families and cross-model reliability are prioritized.

Table 5: Comparison table of models — ResNet, VGG16, EfficientNet, and Ensemble Learning using metrics from the results

| Model | Accuracy | Macro Precision | Macro Recall | Macro F1-Score | Weighted Precision | Weighted Recall | Weighted F1-Score |
|---|---|---|---|---|---|---|---|
| **ResNet** | 0.96 | 0.96 | 0.96 | 0.96 | 0.96 | 0.96 | 0.96 |
| **VGG16** | 0.97 | 0.97 | 0.97 | 0.97 | 0.97 | 0.97 | 0.97 |
| **EfficientNet** | 0.95 | 0.95 | 0.95 | 0.95 | 0.95 | 0.95 | 0.95 |
| **Ensemble Learning** | 0.95 | 0.95 | 0.95 | 0.95 | 0.95 | 0.95 | 0.95 |

#### 4.2.6 Comparative Study

The literature review reveals that many studies [1][2] primarily use accuracy as the metric for evaluating imbalanced malware datasets like the Malimg dataset. However, accuracy alone is not an ideal metric for imbalanced classification, as it can lead to biased model evaluation favoring the majority classes (see Table 6). To address this, we developed and tested a model on the dataset, which demonstrated superior performance in terms of precision, F1 score, and recall. Precision is particularly crucial for malware detection because false positives are costlier than false negatives. This paper also introduces a blended dataset to enhance evaluation and model performance.

Table 6: Comparative table

| | Accuracy | Precision | Recall | F1-Score |
|---|---|---|---|---|
| Abusitta et al. (2023) [5] | 95% | 96% | 95% | 95% |
| VGG16 Model | 97% | 97% | 97% | 97% |
| Ensemble Learning | 95% | 95% | 95% | 95% |

### 4.3 Proposed Deployment Architecture

Given the computational constraints of typical microcontrollers (MCUs) in IoT devices, a direct on-device deployment of the full CNN models is not feasible. Instead, a more practical approach involves a hybrid architecture:

Edge Device: The IoT device acts as a sensor, collecting binary data and performing minimal pre-processing.

Edge Gateway/Aggregator: A more powerful device (e.g., a Raspberry Pi, NVIDIA Jetson, or a home gateway) located within the local network hosts the malware detection model. It receives binaries from multiple IoT devices, performs inference, and flags malicious files.

Cloud (Optional): For aggregated logging, model updates, and analyzing threats across a wider network.

This architecture offloads the heavy computation to the edge gateway, making the system viable without overburdening the end-point IoT devices.

## 5. CONCLUSION

Our work introduced a blended dataset with an intermediate class imbalance to better evaluate IoT malware detection models. On this dataset, our ensemble model achieved a high precision of 97%, outperforming several pre-trained deep learning models and demonstrating strong generalization to unseen samples from known malware families. This presents a significant step towards building robust detectors for the heterogeneous IoT ecosystem. However,

Iraqi Journal for Computers and Informatics
University of Information Technology and Communications
Vol. 49, No. 2, 2023, pp. 268-282
DOI: https://doi.org/10.25195/ijci.v51i2.636
Print ISSN: 2313-190X, Online ISSN: 2520-4912

this study has limitations. The model was not tested against true zero-day threats, and its deployment on resource-constrained devices requires a hybrid edge-based architecture. Furthermore, its vulnerability to adversarial attacks remains an open question.

Future work will focus on: 1) Acquiring and testing on genuine zero-day malware samples, 2) Exploring model compression techniques (pruning, quantization) for on-device deployment, and 3) Analyzing and improving the model's robustness against adversarial evasion attacks. By addressing these challenges, we aim to transition this promising approach from a laboratory setting to a practical security solution.

**Conflicts of Interest**

The authors should pledge that they don't have any conflict of interest in regards of their research. If there are no conflict of interest then authors can declare the following "The authors declare no conflicts of interest".

**Funding**

The funding section of your journal paper template should provide a concise and transparent declaration of the financial support received to carry out the research presented in your paper.

**Acknowledgment**

The preferred spelling of the word "acknowledgment" in America is without an "e" after the "g". Avoid the stilted expression "one of us (R. B. G.) thanks ...". Instead, try "R. B. G. thanks...". Put sponsor acknowledgments in the unnumbered footnote on the first page.

**References**

[1] Jason Brownlee. What is Deep Learning? . In https://machinelearningmastery.com/what-is-deep-learning, 2019.

[2] N. Limbachiya, "Top 10 IoT applications in 2020", DZone, August 18, 2020. [Online Document], Available: DZone Online https://dzone.com/articles/top-5-iot-security-challenges-to-expect-in2020 [Accessed Dec. 16, 2020]. Onlinehttps://dzone.com/articles/top-5- iot-security-challenges-to-expect-in-2020 [Accessed Dec. 16, 2020].

[3] Dutta, V.; Choras, M.; Pawlicki, M.; Kozik, R. A Deep Learning ´ Ensemble for Network Anomaly and Cyber-Attack Detection. Sensors 2020, 20, 4583. https://doi.org/10.3390/s20164583.

[4] J. Jeon, J. H. Park and Y. Jeong, "Dynamic Analysis for IoT Malware Detection With Convolution Neural Network Model," in IEEE Access, vol. 8, pp. 96899-96911, 2020, doi: 10.1109/ACCESS.2020.2995887.

[5] Bendiab, Gueltoum & Shiaeles, Stavros & Alruban, Abdulrahman & Kolokotronis, Nicholas. (2020). IoT Malware Network Traffic Classification using Visual Representation and Deep Learning.

[6] Baptista, Irina & Shiaeles, Stavros & Kolokotronis, Nicholas. (2019). A Novel Malware Detection System Based on Machine Learning and Binary Visualization. 1-6. 10.1109/ICCW.2019.8757060.

[7] Shire, Robert & Shiaeles, Stavros & Bendiab, Gueltoum & Ghita, B.V. & Kolokotronis, Nicholas. (2019). Malware Squid: A Novel IoT Malware Traffic Analysis Framework Using Convolutional Neural Network and Binary Visualisation. 10.1007/978-3-030-30859-9 6.

[8] Hwang, Ren-Hung & Peng, Min-Chun & Nguyen, Van-Linh & Chang, Yu-Lun. (2019). An LSTM-Based Deep Learning Approach for Classifying Malicious Traffic at the Packet Level. Applied Sciences. 9. 3414. 10.3390/app9163414.

[9] P D, Sai Manoj & Sayadi, Hossein & Mohammadi Makrani, Hosein & Nowzari, Cameron & Rafatirad, Setareh & Homayoun, Houman. (2019). Lightweight Node-level Malware Detection and Network-level Malware Confinement in IoT Networks. 776-781. 10.23919/DATE.2019.8715057.

[10] Su, Jiawei & Vargas, Danilo & Prasad, Sanjiva & Daniele, Sgandurra & Feng, Yaokai & Sakurai, Kouichi. (2018). Lightweight Classification of IoT Malware Based on Image Recognition. 664-669. 10.1109/COMPSAC.2018.10315.

Iraqi Journal for Computers and Informatics
University of Information Technology and Communications
Vol. 49, No. 2, 2023, pp. 268-282
DOI: https://doi.org/10.25195/ijci.v51i2.636
Print ISSN: 2313-190X, Online ISSN: 2520-4912

[11] https://www.kaggle.com/datasets/sohamkumar1703/malevis-dataset.

[12] https://www.researchgate.net/figure/Data-description-of-Malimg-Dataset_tbl4_3586 6 4952.

[13] Kim, Jiyeon & Shin, Yulim. (2020). Intelligent Detection of IoT Botnets Using Machine Learning and Deep Learning. Applied Sciences. 10.3390/app10197009.

[14] Reddy, DKK, Behera, HS, Nayak, J, Vijayakumar, P, Naik, B, Singh, PK. Deep neural network based anomaly detection in Internet of Things network traffic tracking for the applications of future smart cities. Trans Emerging Tel Tech. 2020;e4121. https://doi.org/10.1002/ett.4121.

[15] Deldar, Fatemeh, and Mahdi Abadi. "Deep learning for zero-day malware detection and classification: A survey." ACM Computing Surveys 56.2 (2023): 1-37.