Advancing Malware Detection: A Deep Learning Approach with Transfer Learning Techniques

MSc Research Project
MSC Cyber Security

Littletresa George
Student ID: X23233346

School of Computing
National College of Ireland

Supervisor: Khadija Hafeez

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Littletresa George ………………………………		
Student ID:	X23233346……………………		
Programme:	MSC CYBER SECURITY………………	**Year:**	2024……………..
Module:	MSC Research Project………………………………………………..…		
Supervisor:	Khadija Hafeez………………………………….…		
Submission Due Date:	12-12-2024…………………………………………………………………..……		
Project Title:	Advancing Malware Detection:A Deep Learning Approach with Transfer Learning Techniques…………………………………………		
Word Count:	……………………………………. **Page Count**……20……….……..		

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Littletresa…………………………………………………………………………………

Date: 12-12-2024…………………………………………………………………

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Advancing Malware Detection: A Deep Learning Approach with Transfer Learning Techniques

Littletresa George
X23233346

Abstract

In the progressing cyber world, malware creates a notable threat to data security, underscoring the need for timely detection to mitigate its impact. Traditional detection systems, such as signature-based methods, are becoming successful against highly complicated attacks. This paper offers deep learning approaches to malware detection, focusing on comparing the performance of advanced convolutional neural network (CNN) architectures such as XceptionNet, EfficientNetB0, and ResNet50. By examining various patterns from the image data, the system predicts the presence of malware. The models show better accuracy compared to traditional methods. It successfully diminishes the likelihood of misclassification, thus improving the precision of detection. The proposed system's accuracy in detecting malware assists the cyber security department, enabling better cyber-threat handling. EfficientNetB0 emerges as the most accurate model, achieving 96% accuracy, outperforming both XceptionNet and ResNet50, which recorded 95% accuracy. Overall, the research paper aims to present how well these deep learning models can detect the presence of malware by identifying known and unknown malware strains and considering challenges such as disproportionate datasets, adversarial attacks, and the need for real-time detection in cybersecurity employment.

Keywords: Xception, EfficientNetB0, ResNet

1 Introduction

Today, the field of technology grows rapidly. Emerging fields like artificial intelligence, cloud computing, etc., are essential in the real world. The modern world is very much integrated with cyberspace in various dimensions. And so, it brings challenges such as cyber-attacks, privacy issues, etc., which leads to the need for a cyber security department. One of the major cyber-attacks is the malware threat. Malicious software, in short, malware, is any program that can harm the security and functionality of devices, networks, or data. There are many varieties of malware, such as viruses, worms, trojans, spyware, and more. The mode of attack from each of them is different, which causes an unauthorized action on a device. For example, a virus is attached to a program. It replicates other programs when they are executed, whereas in the case of ransomware attacks, it will lock the user's data or information and ask for ransom for its release.

According to the 'India Breach Report' by FalconFeeds, the first six-month period of 2024 witnessed 388 data breaches, 107 data leaks, 39 ransomware activities, and 59 cases of access sales or leaks (INTERNATIONAL JOURNAL on CYBERSPACE LAW and POLICY VOLUME 2 and ISSUE 1 of 2024 INSTITUTE of LEGAL EDUCATION, 2024). The Telegram app was a hotbed for data breaches and scams. A hub of hackers, Telegram coordinates attacks and shares information. According to India's Cyber Threat Report 2023, trojans (41%) and Infectors (33%) are dominant threats in malware attacks ("THREATS in

CYBERSPACE," 2023). The Automobile Industry has the highest number of detections, followed by the Government and Education sector. Since online activities through the internet have become a part of daily life, malware threats can disrupt not only an individual device but also public sectors, industries, the education sector, the healthcare industry, banking, and even the defense sector. Ransomware can access and lock commercial data, like locking a company server, affecting the firm's productivity. Then, they demand huge amounts of money to unlock the data, which leads to financial loss. Likewise, malware threat causes privacy loss, data breaches, unauthorized tracking etc. These issues raise the need for robust malware detection systems essential to cyber security. Detecting malware as soon as possible is just as important as identifying it (Aslan & Samet, 2020).

Signature-based detection and behavior-based detection are the most traditional way of finding malware. In the signature-based detection technique, the model tries to detect the malware by analyzing known signature patterns. In the behaviour-oriented detection method, the model monitors the application or files, and when an unusual behaviour appears, it is tagged as malware. But nowadays, the malware is changing its pattern and behaviour to avoid detection by signature-based systems. Also, these traditional models can't detect zero- day-attacks. Advanced deep learning architecture helps to overcome these problems (Prakash Niraj & Tiwari, 2022).

Image-based malware detection offers a new approach to detecting malicious software by turning malware binaries into grayscale images. It reveals unique visual patterns and textures that are common to various malware families. This technique advances computer vision and deep learning. In particular, convolutional neural networks (CNN) have distinct advantages over traditional signature detection methods in analyzing these patterns and classifying malware efficiently. Visual detection is more robust against obfuscation and kaleidoscope malware. Since visual features are usually preserved even after changes in the underlying code, it achieves high accuracy. It also takes advantage of the scalability and performance of modern CNN architectures, making it a promising solution in terms of precision(Venkatraman, Alazab, & Vinayakumar, 2019).

This project focuses on implementing a deep learning-based malware detection system using three state-of-the-art architectures: XceptionNet, EfficientNet, and ResNet50. Our modification of the approach leverages convolutional neural networks to extract complex features that distinguish Malicious files from harmless files(Saxe & Berlin, 2015). The dataset used is the Malevis_split dataset, which is a comprehensive collection of malware samples. They are processed and analyzed using Python libraries such as Pandas and NumPy. The models were trained and tested in Google Collab using GPU capabilities to speed up computation. The performance of each model is evaluated using metrics such as precision, precision, recall, and F1 score, allowing comparative analysis of their ability to detect malware. This study aims to support the development of improved malware detection techniques that can effectively identify different types of malwares. At the same time, it addresses the limitations of traditional methods. This project provides insights to improve cyber security defenses by exploring the application of deep learning architectures.

1.1 Research Problem

The research aims to answer how efficiently detect image-based malwares using deep learning models to achieve high accuracy.

1.2 Objectives

1. Develop an image-based deep learning model to detect malwares.
   Develop and train deep learning model capable of accurately identify the malwares. Explore modern CNN architectures like XceptionNet, EfficientNetB0, and ResNet50 to identify malware patterns in grayscale images. Prepare the dataset and make it suitable for training the model.
2. Enhance the early detection capabilities using CNN architectures.
   Design a model which has the ability to detect malwares as soon as possible since the malwares spreads rapidly that leads to the corruption of the system.
3. Optimize model performance.
   Fine-tune the chosen deep learning models to improve their sensitivity, specificity, and overall performance.
4. Compare the models to identify the best model for malware detection.
   The architectures such as Xception, EfficientNetB0, and ResNet Models will be evaluated for their efficiency in classifying malware.

1.3 Challenges

1. Quality and quantity of data
   For deep learning architectures, the model performance is based on the quality and quantity of the training dataset. The data should also be balanced. So, obtaining a sufficient and efficient dataset is challenging.
2. Computational resource
   The project tries to compare three CNN architectures, which are built using multiple layers, which leads to a high computational cost.
3. Model Complexity
   Deep learning models, specifically advanced architectures, are very complex in structure and are difficult to interpret.
4. Model evaluation
   Accurate evaluation of model performance with representative datasets to ensure reliability and generalizability.
5. Scalability
   Regularly updating the model with new data and evolving malware trends ensures it remains relevant and accurate.

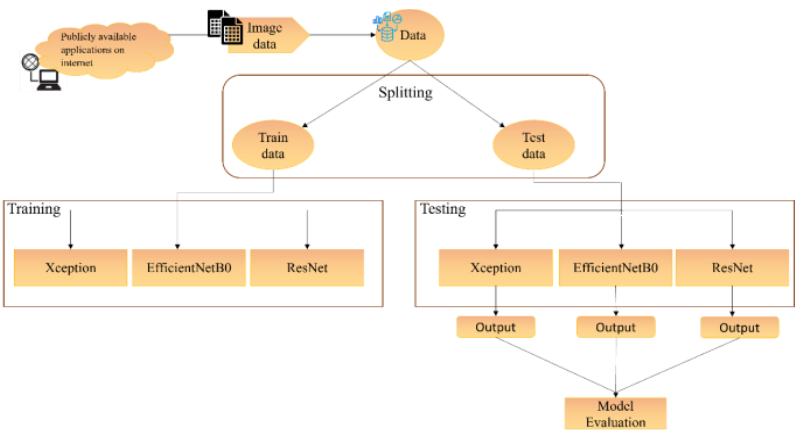


Figure 1: System Diagram

2 Related Work

2.1 Traditional Malware Detection Methods

Traditional malware detection techniques mainly include signature-based malware detection mechanisms, behavior-based malware detection, and heuristic-based malware detection techniques. (Tahir, 2018) detailly reviewed malware types and compared various malware analysis and detection techniques. The research presented three major malware detection techniques: signature-based, heuristics-based, and specification-based. While comparing these techniques, it is noted that signature-based detection methods are good at detecting known malware with fewer resources. At the same time, heuristic and specification-based detection methods can also detect new and unknown malware. Still, with the heuristic-based approach, false positives are high and need more resources. The issue with the specification-based detection approach is developing the specification data of the legitimate program, which is a time and space-consuming process. (Venugopal & Hu, 2008) described signature-based malware detection, specifically the signature matching method, with less memory usage and fast scanning speed. This approach could obtain stable virus features using many virus samples, but it is not efficient in detecting unknown viruses. (Abiola & Marhusin, 2018) developed a detection model by extracting the signatures of the Brontok worms and used an n-gram technique to break down the signatures. This process makes removing redundancies between the signatures of the different types of Brontok malware easier. Hence, it was used in this study to accurately differentiate between the signatures of both malicious and normal files. This method fails against zero-day attacks. ("Research Commons," 2019) discussed behavior-based approaches to ransomware detection. The research combined three Indicators of Compromise (IoC): file changes, file entropy, and canary file. Compared to signature-based detection techniques, this approach is less resource-intensive. However, the paper addressed some difficulties in handling a special type of ransomware called Petya. Martina. (Lindorfer, Kolbitsch, & Milani Comparetti, 2011) implemented a system called Disarm: Detecting Sandbox-AwaRe Malware, which detects differences in behaviour regardless of their cause. This allows us to discard specious differences in behaviour and identify "environment-sensitive" samples that exhibit semantically different behaviour. Any monitoring technology that can detect persistent changes to the system state at the operating system level can take advantage of this technique.

2.2 Modern Malware Detection Techniques

This includes malware detection based on machine learning models and deep learning models, which gave better results than traditional techniques. (Firdausi, lim, Erwin, & Nugroho, 2010) analysed five machine learning techniques for malware detection. The classifiers used in this research are k-Nearest Neighbors (kNN), Naïve Bayes, J48 Decision Tree, Support Vector Machine (SVM), and Multilayer Perceptron Neural Network (MlP). Their study showed that these classical machine learning methods could effectively classify malware based on features extracted from system behaviour but accurate feature selection is very important to ensure the model accuracy. In the paper prepared by (Narayanan & Davuluru, 2020) three different approaches are studied to classify malware programs based on different file formats including CNN-based approach for classifying malware compiled files after visualizing them as images, a Recurrent Neural Network (RNN)-based approach for classifying malware assembly files, and an ensemble approach of combining the features extracted using CNN and RNN techniques and later classifying them either using logistic regression or SVM. Representing malware programs both in terms of compiled and assembly-level files helped overcome a lack of information present in either of those file types. (Saxe & Berlin, 2015b) presented a malware detection system based on a deep neural network. Here, they used a convolutional neural network to detect the binary values by converting them into images. This approach achieves good accuracy with a low false positive rate. However, it is noted that the model's performance depends upon the quantity and quality of the training dataset. The model also wanted high computational resources. (Jha, Prashar, Long, & Taniar, 2020) developed a Recurrent Neural network model to detect malware. The research paper also measured the performance of RNN using three feature vectors such as “hot encoding feature vector,” “random feature vector,” and “Word2Vec feature vector”. They concluded that RNN with Word2Vec feature vector performed well. However, this approach takes a comparatively long time to identify the malware. (Mathew & M. A. Ajay Kumara, 2019) Proposed RNN- Long-Short-Term Memory (LSTM) model to learn from the most informative of sequences from the API dataset based on their relative ranking based on Term Frequency-Inverse Document Frequency (TF-IDF) recommended features.

2.3 Advanced Malware Detection-Deep Neural Network Architecture with Transfer-Learning Techniques

(Haque, Jahin, Labiba Ifrit, Sheikh, Mahmud, & Tasnia, 2024b) presented six Convolutional Neural Network (CNN) models that were utilized, with “InceptionResNetV2 exhibiting the most promising performance. InceptionResNetV2 amalgamates the strengths of two established models: Inception, which is focused on determining computational cost, and ResNet, which prioritizes computational accuracy. Other CNN architectures used in this paper include Inception ResNetV2, DenseNet, VGG16, ResNet50, EfficientNetB0, and XceptionNet. (Raman Kumar, 2022) proposed an IVMCT framework for the multi-class classification of malware using transfer learning on the MalIimg dataset. In the IVMCT framework, 3 pre-trained models are selected: ResNet, AlexNet, and DenseNet. The pre-trained models are trained in some domains, and the models' weights are adjusted accordingly. These models are extended in other domains as well with fewer number resources, and time is also consumed less. (Aslan & Yilmaz, 2021) proposed a hybrid model by optimally combining two well-known pre-trained network models, Alexnet and Resnet, and the method is evaluated on Malimg, Microsoft BIG 2015, and Malevis datasets. Here, the suggested hybrid model is first compared with each individual model separately. This method showed the best accuracy, but a minority of malware samples could not be classified correctly because those malware variants used advanced code obfuscation techniques, which is a limitation. (Naeem et al., 2020) designed an architecture that used a methodology that integrates malware visualization into the DCNN model for detecting malicious activities in the Industrial Internet of Things (IIoT) environment. After converting APK files to colour images by malware visualization technique, DCNN model extracted the malware's dynamic image features and classified them.

The model's accuracy is higher than the previous machine learning techniques (Yadav, Menon, Ravi, Vishvanathan, & Pham, 2022) Developed EfficientNet-B4, a CNN-based architecture to detect Android malware using image-based malware representations of the Android DEX file. EfficientNet-B4 extracts relevant features from the malware images. These features are then passed through a global average pooling layer and fed into a softmax classifier. The proposed method obtained an accuracy of 95.7% in the binary classification of Android malware images, which shows the model's robustness.

Table 1: Comparison of articles

Title	**Methodology**	**Limitation**
A study on malware and malware detection techniques	Signature-based, Heuristic-based, Specification-based	Prone to unknown malware patterns, High false positive rate
Efficient signature-based malware detection on mobile devices	Signature matching method	Not efficient in detecting unknown malware patterns
Signature-based malware detection using sequences of N-grams	Signature-based, N-grams techniques	Fails against zero-day attacks
Behaviour-based ransomware detection	Behaviour-based approach	Faced difficulties in detecting the special type of ransomware called Petya
Detecting environment-sensitive malware	Behaviour-based approach	Need frequent monitoring
Analysis of machine learning techniques used in behavior-based malware detection	k-Nearest Neighbors (KNN), Naïve Bayes, J48 Decision Tree, Support Vector Machine (SVM), and Multilayer Perceptron Neural Network (MLP)	Accurate feature selection is a challenge
Ensemble malware classification system using deep neural networks	CNN, RNN, Ensemble learning	the computational resource is needed
Deep neural network-based malware detection using two-dimensional binary program features	CNN	Model performance depends on the quantity and quality of training data. The high computational resource is needed
Recurrent neural network for detecting malware	RNN, hot encoding feature vector, random feature vector, and Word2Vec feature vector	Take comparatively a long time to identify those malwares.

API call-based malware detection approach using a recurrent neural network— LSTM	RNN-Long-Short-Term Memory (LSTM)	The current feature selection is only done using the TF-IDF technique. Other feature selection techniques, such as Chi-square, Fisher's score, and combinations, are also noted.
MalFam: a comprehensive study on malware families with state-of-the-art CNN architectures with classifications and XAI	Inception, ResNetV2, DenseNet, VGG16, ResNet50, EfficientNetB0 and XceptionNet	Mapping) or Integrated Gradients can provide more Detailed and accurate analysis compared to LIME
IVMCT: Image visualization-based multiclass malware classification using transfer learning	ResNet, Alexnet and DenseNet	Incapable of detecting zero-day malware or obfuscated malware.
A new malware classification framework based on deep learning algorithms.	Alexnet and Resnet	Prone to advanced code obfuscation techniques
Malware detection in industrial Internet of Things based on hybrid image visualization and deep learning model.	DCNN	
EfficientNet convolutional neural networks-based Android malware detection	EfficientNet-B4	

2.4 Identified Gap

The works related to this project topic can divided into three: Traditional malware detection methods, modern methods to detect malware, and advanced malware detection using deep learning architectures. These papers show the need of employing deep neural network architectures for detecting malwares. The feature extraction process is done manually using modern techniques, whereas these advanced architectures automatically identify and extract features from the data. Advanced architectures like EfficientNet, ResNet, and XceptionNet recognize hidden patterns in the data through hierarchical learning that helps to identify newly born malware, but the former struggle to detect these newborn malwares. Since these architectures are pre-trained on large datasets, the amount of the training data needed is less compared to traditional machine learning models. Since Deep learning architecture generalizes from training data, it makes it suitable for effective detection of zero-day attacks where the traditional methods fail.

In case of accuracy, the three models- EfficientNet, ResNet, and XceptionNet- give the same accuracy. If lightweight and speed are priorities, go with EfficientNetB0 or EfficientNetV2. They offer a smaller model size and better inference speed, making them ideal for deployment in environments with limited resources. If you prioritize robustness and don't mind a larger model, ResNet50 is a solid, well-established choice.

2.5 Why this project Approach

Image-based malware detection techniques have shown superior capabilities in analyzing and classifying malicious software by transforming binaries into visual representations. This allows leveraging advanced deep learning methods such as CNNs and transformers to detect sophisticated evasion tactics like obfuscation and polymorphism (Basak & Han, 2024). The effectiveness of deep learning models such as ResNet, EfficientNet, and Xception in image classification has been widely recognized in the literature. These architectures are good at recognizing complex patterns in images. This makes it suitable for malware detection tasks. In addition, the image-based method allows for scalability; once it is trained. Studies utilizing CNNs with architectures like ResNet-152 or Vision Transformers (ViT) report high accuracy (up to 99.62%) in classifying malware using features derived from pixel intensities, further validating the robustness of this method (Ashawa, Nsikak Owoh, Hosseinzadeh, & Osamor, 2024). The availability of datasets such as Malevis_split increases the feasibility of this approach. The dataset provides structured, high-quality data for training and validation. This facilitates experimentation with modern architecture. These points highlight why image-based malware detection is an advanced, reliable, and forward-looking solution for malware research. And why was it chosen for this project.

To create a detailed guide for developing a malware detection model from the ground up, the following steps are outlined, the first step is crucially preparing the dataset. This process starts with gathering an appropriate dataset, such as the Malevis_split dataset, which includes various labeled malware and benign samples. These binaries are then converted into grayscale or color images for further examination. Image normalization and resizing (for example, 224x224 pixels for models like ResNet50 or EfficientNetB0) ensure that they are compatible with the chosen architectures. Finally, the dataset is partitioned into training, validation, and test subsets to allow for a thorough evaluation process. The next phase involves choosing a suitable model architecture. Pre-trained models, such as ResNet50, EfficientNetB0, and XceptionNet, are particularly well-suited for image-based malware detection. These architectures make use of transfer learning, taking advantage of the features, they have learned from large datasets to identify significant patterns in malware images. The third phase centers on the model training. Training begins by locking the base layers of the pre-trained models to preserve their learned features, after which custom layers designed for malware classification are appended. Loss functions such as categorical cross-entropy or binary cross-entropy, combined with optimizers like Adam, enable effective model training. Performance metrics, including accuracy, precision, recall, and AUC, are tracked to ensure steady improvement. In the subsequent evaluation and deployment phases, the model's effectiveness is assessed and readied for real-world use. During testing, metrics like false positives and false negatives are analyzed in detail, while the model is optimized for use in resource-limited settings through methods such as pruning or converting to lighter formats like TensorFlow Lite. Lastly, continuous enhancement is essential to this strategy. Regular retraining with the latest malware samples and frequent error analysis is crucial for preserving the model's effectiveness against changing malware behaviors, particularly for zero-day threats.

3 Research Methodology

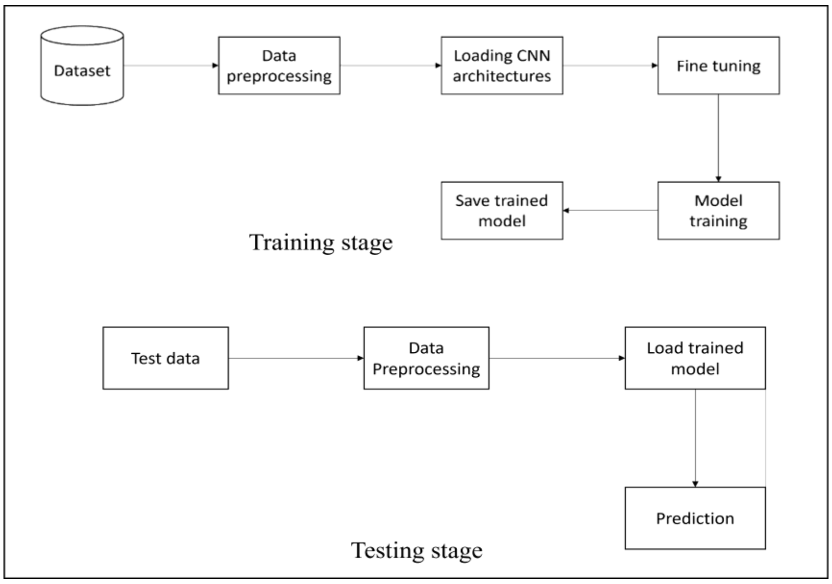


Figure 2: Proposed dataflow diagram

3.1 Dataset Overview

This research uses a standard dataset named Malevis_split, which is taken from Kaggle, a publicly available platform for downloading various datasets. It consists of grayscale images of various malware, and the dataset contains 26 types of malwares.

Why image-based dataset

Converting malware binaries to grayscale images helps to identify unique patterns and textures for different malware families. This binary visualization method improves detection by exploiting confusingly flexible image features. Although malware authors attempt to change the binary to avoid traditional detection, the visual patterns often remain. This makes the image-based method more robust and advances in computer vision and deep learning architectures such as ResNet, EfficientNet, and Xception offer powerful tools for images classification that can be used for highly accurate malware detection. Once the models are trained It provides scalability by efficiently classifying malware without the need for real-time binary analysis.

3.2 Data Preprocessing

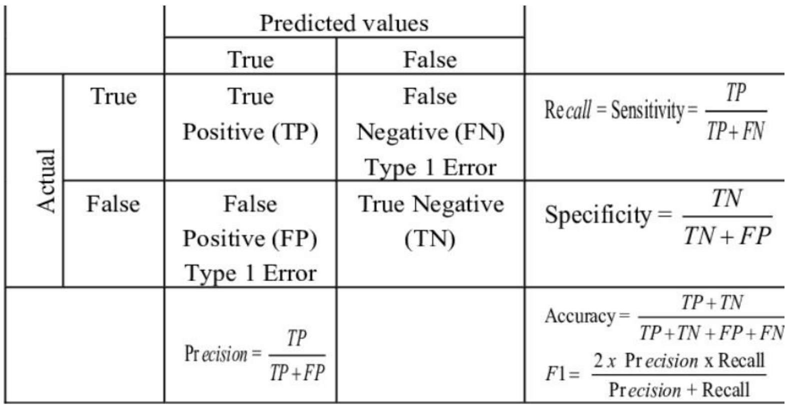
Data preprocessing includes balancing and shaping image data. Data imbalanced means some categories have a large number of data points. It is necessary to balance data for better performance. So here, the category with a large number of data points is removed. The dimension of all images in the dataset is (224, 224, 3). To make the input of CNN architectures, pass the shape as (200, 200, 3).

3.3 Evaluation Metrics

It is very important to assess the results of a work. There is different evaluation metrics are there to measure the performance of machine learning models. Accuracy, Precision, F1 Score, Recall are some popularly used techniques (GeeksForGeeks, 2018)

Confusion matrix:

A confusion matrix sums up the performance of a model on a set of test data. It displays the model's prediction performance by showing the count of correct and incorrect predictions. It contains the values called true positive, true negative, false positive and false negative.[18]



		Predicted values		
		True	False	
Actual	True	True Positive (TP)	False Negative (FN) Type 1 Error	$Recall = Sensitivity = \frac{TP}{TP+FN}$
	False	False Positive (FP) Type 1 Error	True Negative (TN)	$Specificity = \frac{TN}{TN+FP}$
		$Precision = \frac{TP}{TP+FP}$		$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$ $F1 = \frac{2\,x\ Precision\ x\ Recall}{Precision + Recall}$

Figure 3: Confusion Matrix

- True Positive (TP): It is the number of correctly predicted positive outcome. That is the predicted value and the actual value is positive (GeeksForGeeks, 2018).
- True Negative (TN): It is the number of correctly predicted negative outcome. That is the predicted value and actual value is negative (GeeksForGeeks, 2018).
- False Positive (FP): It shows the number of incorrectly predicted positive outcomes. That is the model predicted the value as positive but actually it is negative.it is also known as a Type I error (GeeksForGeeks, 2018).
- False Negative (FN): It shows the number of incorrectly predicted negative outcomes. That is the model predicted the output as negative but actually it is positive.it is also known as a Type II error (GeeksForGeeks, 2018).

Accuracy
It is a fundamental metric to measure the potential of a model which helps to understand how the model performs in terms of correct predictions. This is a calculation of the ratio between correct number of predictions and total number of input data (GeeksForGeeks, 2018).

- **Equation 1**

$$Accuracy = \frac{(\mathrm{TP} + \mathrm{TN})}{(\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN})}$$

Precision
Precision is another metric that measures model's performance in terms of how many of the positive predictions made by the model are actually correct. It is the ratio of the number of true positive predictions to the sum of true positive and false positive predictions(GeeksForGeeks, 2018).

- **Equation 2**

$$\mathrm{Precision} = \frac{\mathrm{TP}}{(\mathrm{TP} + \mathrm{FP})}$$

Recall
Recall measures the model's sensitivity by determining the percentage of actual positive cases that were accurately predicted as positive. Dividing the number of true positives by the number of positive instances calculates recall (GeeksForGeeks, 2018).

- **Equation 3**

$$\mathrm{Recall} = \frac{\mathrm{TP}}{(\mathrm{TP} + \mathrm{FN})}$$

F1 Score
F1-score is helpful to measure the overall performance of a classification model. It is the harmonic mean of precision and recall (GeeksForGeeks, 2018).

- **Equation 4**

$$\mathrm{F1-score} = 2\,\frac{(\mathrm{precision} * \mathrm{recall})}{(\mathrm{precision} + \mathrm{recall})}$$

4 Design Specification

The CNN architectures used here are Xception, EfficientNetB0 and ResNet50. These are pretrained CNN models- where a model is trained on large dataset- which are used in image classification tasks. The learning using pretrained model to new task is called transfer learning.

4.1 XceptionNet

Xception, developed by Francois Chollet in 2017, is a deep convolutional neural network known for its novel feature learning. It uses depth-separable convolutions, which decompose standard convolutions into depth and pointwise convolutions. In particular, a spatial convolution is performed independently on each channel of an input, followed by a pointwise convolution (1x1 convolution) that projects the channels output by the depth convolution to a new channel space. This unique design reduces the number of parameters while maintaining significant performance. It consisted of 36 convolutional layers organized into 14 modules. All modules except the first and last have linear residual connections around them (Chollet, 2017).

4.2 ResNet50

ResNet50 is a powerful deep convolutional neural network architecture (Rezende, Ruppert, Carvalho, Ramos, & de Geus, 2017). It is the short form of Residual Networks. learn residual functions with reference to the layer inputs, instead of learning unreferenced functions. Instead of hoping each few stacked layers directly fit a desired underlying mapping, residual nets let these layers fit a residual mapping (Rezende, Ruppert, Carvalho, Ramos, & de Geus, 2017). ResNet50 consists of 50 layers and features an innovative use of residual blocks, which allows the network to skip connections and mitigate the vanishing gradient problem during training. This design facilitates the training of very deep networks, enabling better performance in various computer vision tasks. The ResNet family includes several versions like ResNet-50, ResNet-101, and ResNet-152, with increasing depth. The "50," "101," etc., indicate the number of layers, allowing researchers to select models based on their computational constraints.

4.3 EfficientNet

EfficientNet is a family of neural network architectures that were introduced by Google AI researchers in 2019. EfficientNet aimed to create a highly accurate and efficient model in terms of computational resources. Efficient Net achieves this Balance by using a Combination of three Main Techniques - Compound Scaling, Efficient Channel Attention, Neural Architecture Search. Efficient Net uses a compound scaling method, which scales the neural network in a uniform manner in all three dimensions - depth, width, and resolution. This approach involves using a compound coefficient that uniformly scales the neural network's depth, width, and resolution. This allows for much more efficient use of computational resources, leading to higher accuracy with less computing power ("EfficientNet: A Breakthrough in Machine Learning Model Architecture - Javatpoint," 2024-b). Width scaling involves increasing the number of channels in each convolutional layer of the network. This increases the capacity of the network to learn more complex patterns in the input data. Depth scaling involves adding more convolutional layers to the network. This allows the network to learn more abstract and complex features from the input data. Resolution scaling involves increasing the size of the input images. This allows the network to capture more fine-grained details in the input data, which can be particularly important for object detection and segmentation tasks. This approach involves using a compound coefficient that uniformly scales the neural network's depth, width, and resolution. This allows for much more efficient use of computational resources, leading to higher accuracy with less computing power.

4.4 Fine-tuning

Fine-tuning is the process of taking a pre-trained language model and adapting it to perform a particular task or set of tasks. It bridges the gap between a general-purpose language model and a specialized AI solution. Fine-tuning addresses the issue of adaptability.

Fine-tuning is the process of taking a pre-trained model and training it using a similar dataset. This method is used to adopt an already created model for solving another similar problem. Without beginning the training process over, the objective is to maximize the model's performance on a novel, related task. During the process, the architecture of the model will not be affected. The objective is to use the model's valuable characteristics and representations gained from the large dataset on which it was initially trained and adapt them to a narrower job. It bridges the gap between a general-purpose language model and a specialized AI solution. Fine-tuning addresses the issue of adaptability (IBM, 2024b). Fine-tuning can be done through modifying different types of model parameters. Modifying layers of pre-trained models, unfreezing layers of pre-trained models, modifying parameters like learning rate, batch size are some techniques that are used in fine-tuning process.

A pre-trained model can be fine-tuned to perform better with little further training by adapting it to a particular job or domain. Because the model doesn't have to learn from start, fine-tuning saves time and computational resources by utilizing the model's current knowledge. In specialized domains, such as modifying a language model to comprehend medical writings, assessing sentiment in financial papers, or categorizing images in disciplines like radiology, it improves performance. Because the model may build on its general, previously learned understanding and concentrate exclusively on fine-tuning details pertinent to the new task, fine-tuning is particularly helpful when we have little task-specific data. With this approach, we can produce specialized, extremely accurate models quickly and adaptably for a wide range of uses. It can be quite costly and time-consuming to train a deep learning model from start. Contrarily, fine-tuning enables us to improve upon a model that has already been trained, thus cutting down on the time and resources needed to provide quality results.

In this project the fine-tuning process starts with initialization, where the pre-trained model, training data, validation data, and other parameters are set. The pre-trained model's layers are divided into two parts: the initial layers, which are frozen, and the later layers, which are trainable. The frozen layers retain the general features learned from the large dataset, while the trainable layers adapt to the new malware classification task. Additional layers are added to the pre-trained model to improve its representation and classification capabilities. These layers include a flatten layer, fully connected layers with dropout (if regularization is applied), and a final dense layer with a softmax activation function. The updated model is compiled with an optimizer (Adam), a loss function (categorical cross-entropy), and evaluation metrics (accuracy, AUC, false positives, precision, and recall). The model is trained on the training data with validation on the validation data. Early stopping is applied to prevent overfitting, and the best weights are restored to ensure optimal performance.

The fine-tuning process leverages the pre-trained model's general features and adapts the later layers to learn task-specific patterns for malware classification. The regularization option helps avoid overfitting, especially with small datasets. This approach efficiently adapts the pre-trained model for malware image classification while maintaining generalization and avoiding overfitting.

5 Implementation

5.1 System Specification

The current experiment was conducted using Google Colab, which provided enhanced computational resources for model training.

The three CNN architectures – Xception, EfficientNetB0 and ResNet50 – have been implemented and their respective confusion matrix and classification reports were obtained. The classification report contains evaluation metrics such as accuracy, precision, recall and f1-score which helps to analyse the performance of the model. Finally, a comparison study of three models have been performed in the basis of efficiency, model size and computational resources.

5.2 Dataset description

The dataset is taken from Kaggle which is a publicly available platform to download various datasets. It consists of grayscale images of malware classes in PNG format. There are 26 type malwares namely “Adposhel, Agent, Allaple, Amonetize, Androm, Autorun, BrowseFox, Dinwod, Elex, Expiro, Fasong, HackKMS, Hlux, Injector, InstallCore, MultiPlub, Neoreklami, Neshta, Other, Regrun, Sality, Snarasite, Stantinko, VBA, VBKrypt, Vilsel”.

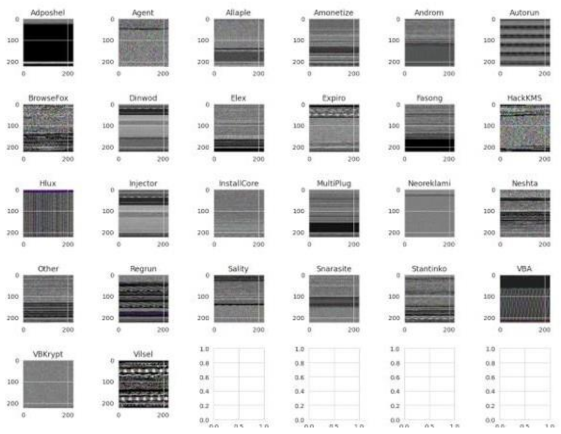


Figure 4: Sample of Malware types

5.3 Package Installation

A small number of libraries that are used in this project are:

- OS: Module used to interact with file systems.
- TensorFlow: Module used for model training and transfer learning
- Sklearn: Module used for model evaluation and selection
- Matplotlib, Seaborn: Modules used for data visualization
-

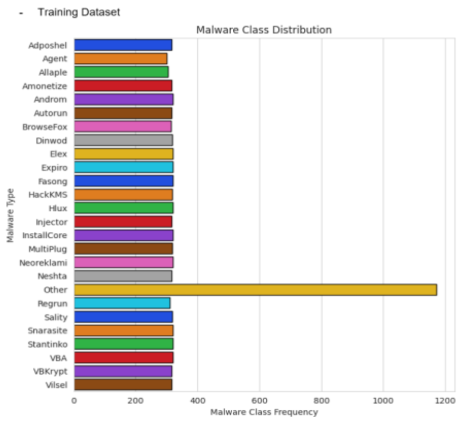


Figure 5: Malware class frequency

5.4 Training Stage

Data Preprocessing

Figure 5 shows image counts that belongs to each malware classes. It is obvious that the class named ‘other’ have large amount of data compared to other classes. It results an imbalance dataset. It should have to remove this imbalance to make good and accurate model’s results. So here, the class named ‘other’ was removed to make a balanced dataset. In figure 6 the dataset is balanced.

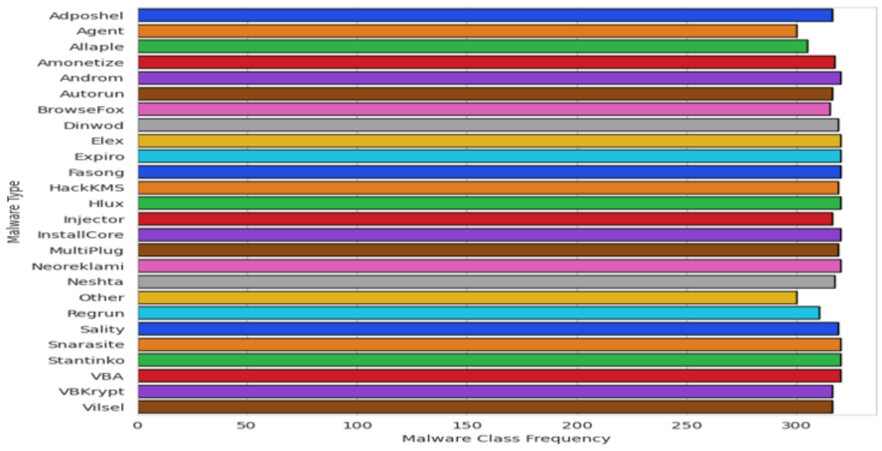


Figure 6: Balanced dataset - class frequency

6. Evaluation

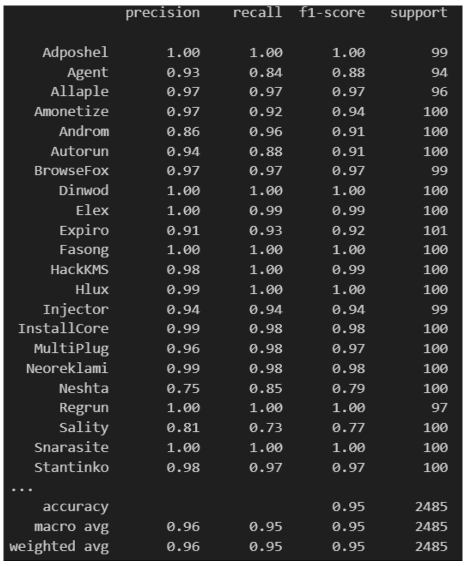
This section comprehensively assesses three CNN-based models—XceptionNet, EfficientNetB0, and ResNet50 used for malware detection. The analysis centers on their classification performance among 24 malware types and incorporates essential metrics such as accuracy, precision, recall, and F1-score. Furthermore, the confusion matrices shed light on each model's ability to differentiate between various malware families, showcasing strengths and identifying areas for improvement. Lastly, the models are evaluated regarding efficiency, computational resource demands, and robustness, providing recommendations.

6.1 Case Study 1: Model evaluation - ‘XceptionNet’

XceptionNet recorded an impressive overall accuracy rate of 96%, showcasing its capability in malware detection. The classification report indicates consistently strong performance across the majority of malware categories, with macro-averaged precision, recall, and F1-scores standing at 0.96, 0.95, and 0.95, respectively. These figures suggest a well-balanced model that can provide dependable results with minimal false positives and false negatives. Certain categories, including Adposhel, HackKMS, and Regrun, display near-perfect precision and recall values of 1.00, underscoring the model's robustness and dependability in identifying these malware families.

However, some classes, such as Neshta and Sality, show lower recall values of 0.85 and 0.75, respectively. These decreased scores imply that the model struggles to accurately identify all samples from these families, leading to false negatives. The confusion matrix illustrates these challenges, revealing that some samples of Neshta were misclassified as belonging to other families. Similarly, Sality has a higher incidence of misclassification, with off-diagonal elements highlighting confusion with other malware types. These difficulties may arise from overlapping feature distributions between Sality, Neshta, and other families or inadequate representation of these categories in the training set.

A significant finding from the confusion matrix is the prevalence of diagonal elements, with most categories achieving correct classifications approaching 100. For example, family 1 achieved 99 correct classifications, while family 10 reached 100 correct classifications, reflecting the model's strong capability to differentiate between malware families. Nevertheless, off-diagonal elements for certain families, such as family 2, reveal misclassifications into families 3, 4, or 5, indicating potential areas for enhancing feature differentiation. Future improvements, such as more advanced feature extraction methods or class-specific data augmentation, could help mitigate these challenges.



```
              precision    recall  f1-score   support

    Adposhel       1.00      1.00      1.00        99
       Agent       0.93      0.84      0.88        94
     Allaple       0.97      0.97      0.97        96
   Amonetize       0.97      0.92      0.94       100
      Androm       0.86      0.96      0.91       100
     Autorun       0.94      0.88      0.91       100
  BrowseFox        0.97      0.97      0.97        99
      Dinwod       1.00      1.00      1.00       100
        Elex       1.00      0.99      0.99       100
      Expiro       0.91      0.93      0.92       101
     Fasong        1.00      1.00      1.00       100
     HackKMS       0.98      1.00      0.99       100
        Hlux       0.99      1.00      1.00       100
    Injector       0.94      0.94      0.94        99
 InstallCore       0.99      0.98      0.98       100
   MultiPlug       0.96      0.98      0.97       100
  Neoreklami       0.99      0.98      0.98       100
      Neshta       0.75      0.85      0.79       100
      Regrun       1.00      1.00      1.00        97
      Sality       0.81      0.73      0.77       100
   Snarasite       1.00      1.00      1.00       100
   Stantinko       0.98      0.97      0.97       100
...
    accuracy                           0.95      2485
   macro avg       0.96      0.95      0.95      2485
weighted avg       0.96      0.95      0.95      2485
```

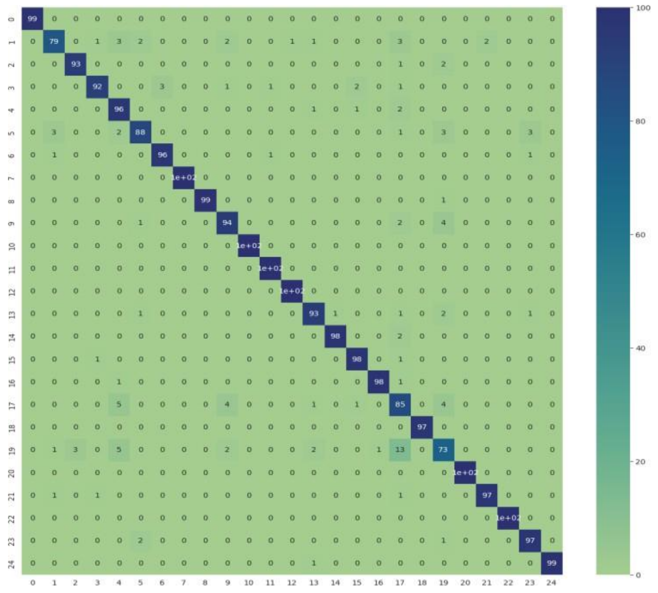


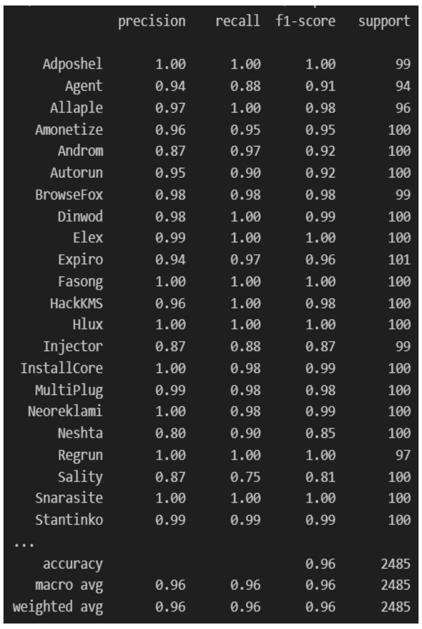
Figure 7: Classification report of xception model **Figure 8: Confusion matrix of Xception model**

6.2 Case Study 2: Model evaluation – 'EfficientNetB0'

EfficientNetB0 emerged as the leading model, achieving the highest overall accuracy of 96% among the three architectures tested. The macro-averaged precision, recall, and F1-scores closely match those of XceptionNet, demonstrating consistent and reliable performance across all malware types. Importantly, the model shows exceptional performance in detecting difficult malware families, attaining perfect scores for categories like Adposhel and Neorekclami, with both precision and recall at 1.00. These findings highlight the model's robustness and its capability to effectively handle well-represented categories.

However, EfficientNetB0 does display slightly lower precision and recall for specific families, such as Injector. The metrics for this category reflect some level of misclassification, likely due to overlapping features with other families. The confusion matrix also shows that while most predictions align along the diagonal, a few off-diagonal entries point to misclassifications. For example, Injector sometimes overlaps with Automation and other categories, resulting in lower precision and recall scores relative to the best-performing classes.

The lightweight design and computational efficiency of EfficientNetB0 make it especially well-suited for use in real-time malware detection systems or scenarios with limited resources. Its capability to sustain high recall scores across most categories, particularly challenging ones, emphasizes its usefulness for applications where reducing false negatives is crucial.



```
              precision    recall  f1-score   support

    Adposhel       1.00      1.00      1.00        99
       Agent       0.94      0.88      0.91        94
     Allaple       0.97      1.00      0.98        96
   Amonetize       0.96      0.95      0.95       100
      Androm       0.87      0.97      0.92       100
     Autorun       0.95      0.90      0.92       100
   BrowseFox       0.98      0.98      0.98        99
      Dinwod       0.98      1.00      0.99       100
        Elex       0.99      1.00      1.00       100
      Expiro       0.94      0.97      0.96       101
      Fasong       1.00      1.00      1.00       100
     HackKMS       0.96      1.00      0.98       100
        Hlux       1.00      1.00      1.00       100
    Injector       0.87      0.88      0.87        99
 InstallCore       1.00      0.98      0.99       100
   MultiPlug       0.99      0.98      0.98       100
  Neoreklami       1.00      0.98      0.99       100
      Neshta       0.80      0.90      0.85       100
      Regrun       1.00      1.00      1.00        97
      Sality       0.87      0.75      0.81       100
   Snarasite       1.00      1.00      1.00       100
   Stantinko       0.99      0.99      0.99       100
...
    accuracy                           0.96      2485
   macro avg       0.96      0.96      0.96      2485
weighted avg       0.96      0.96      0.96      2485
```

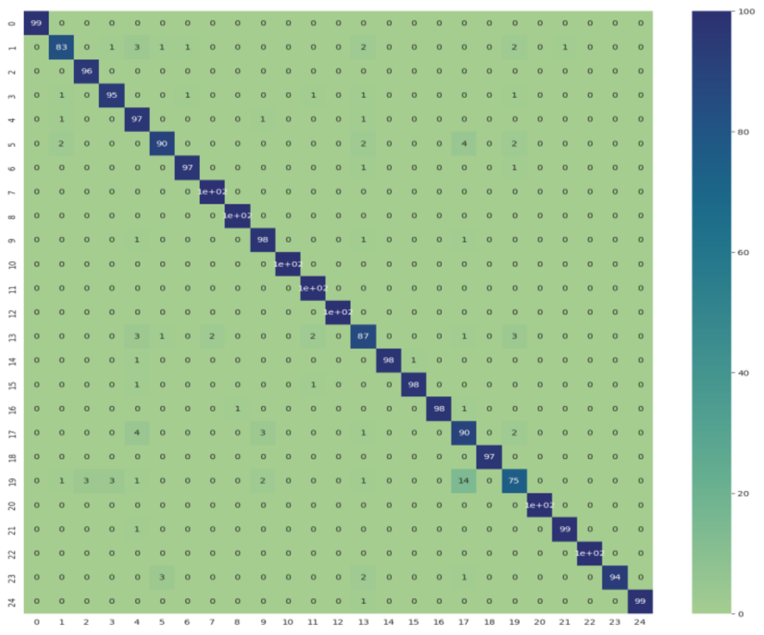
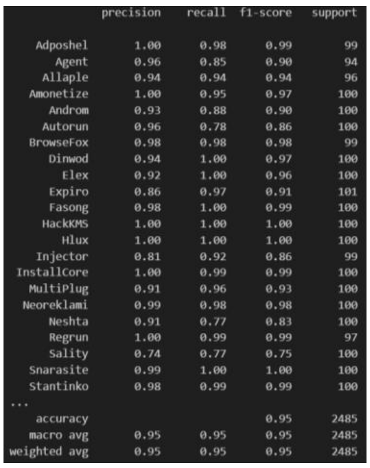


Figure 9: Classification report of EfficientNetB0 model

Figure10: Confusion matrix of EfficientNetB0 model

6.3 Case Study 3: Model evaluation – ‘ResNet50’

The classification report in figure11 shows an overall accuracy of 95% across 2485 samples. Most classes have high precision, recall, and F1-scores, indicating balanced performance. However, classes like "Sality" (F1-score: 0.79) and "Injector" (F1-score: 0.89) have slightly lower scores, indicating some room for improvement. Macro and weighted averages for precision, recall, and F1-score are consistent at 0.95, reflecting strong overall model performance. The model demonstrates reliable predictions with minor misclassifications in a few specific classes. The confusion matrix in figure12 indicates accurate predictions with most values concentrated along the diagonal. Diagonal elements are close to 100, confirming strong performance for individual classes. Non-diagonal elements are sparse, highlighting minimal misclassifications. Classes like 1 and 20 show slightly higher misclassifications compared to others. Overall, the model demonstrates consistent and reliable classification.



```
              precision    recall  f1-score   support

    Adposhel       1.00      0.98      0.99        99
       Agent       0.96      0.85      0.90        94
     Allaple       0.94      0.94      0.94        96
   Amonetize       1.00      0.95      0.97       100
      Androm       0.93      0.88      0.90       100
     Autorun       0.96      0.78      0.86       100
   BrowseFox       0.98      0.98      0.98        99
      Dinwod       0.94      1.00      0.97       100
        Elex       0.92      1.00      0.96       100
      Expiro       0.86      0.97      0.91       101
      Fasong       0.98      1.00      0.99       100
     HackKMS       1.00      1.00      1.00       100
        Hlux       1.00      1.00      1.00       100
    Injector       0.81      0.92      0.86        99
 InstallCore       1.00      0.99      0.99       100
   MultiPlug       0.91      0.96      0.93       100
  Neoreklami       0.99      0.98      0.98       100
      Neshta       0.91      0.77      0.83       100
      Regrun       1.00      0.99      0.99        97
      Sality       0.74      0.77      0.75       100
   Snarasite       0.99      1.00      1.00       100
   Stantinko       0.98      0.99      0.99       100
...
    accuracy                           0.95      2485
   macro avg       0.95      0.95      0.95      2485
weighted avg       0.95      0.95      0.95      2485
```

Figure 11: Classification report of ResNet50 model

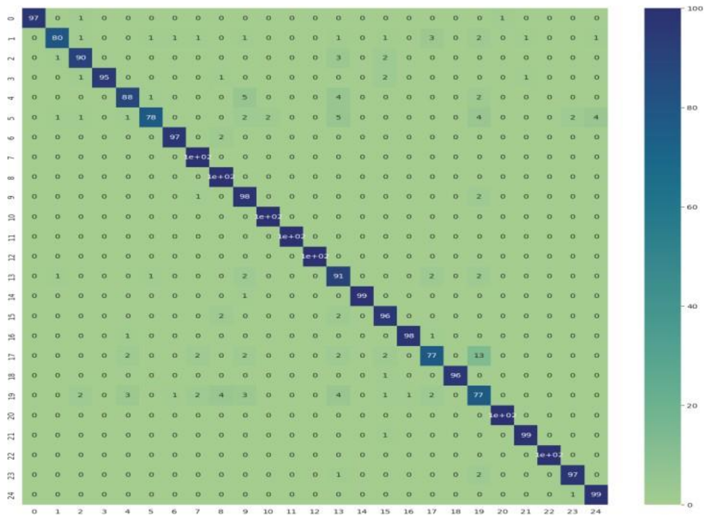


Figure 12: Confusion matrix of ResNet50 model

6.4 Case Study 4: Model Comparison

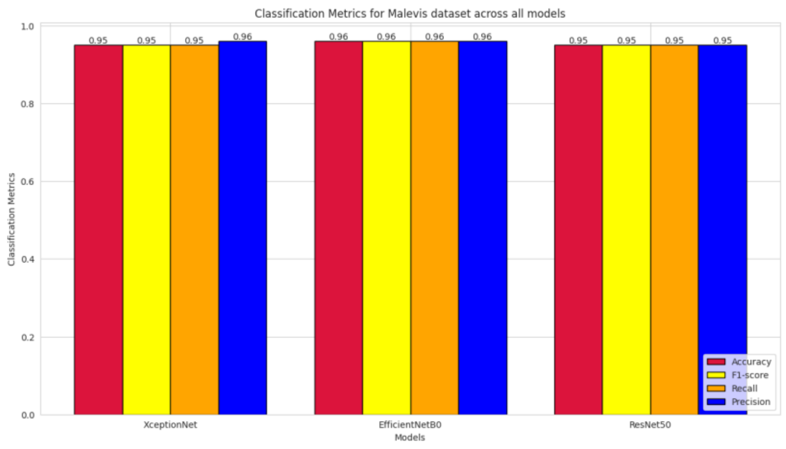


Figure 13: Evaluation metrics comparison

Figure 13 compares various classification metrics accuracy, precision, recall, and F1-score across three CNN models: XceptionNet, EfficientNetB0, and ResNet50, evaluated using the Malevis dataset. Each bar indicates the value of the corresponding metric for a specific model, offering a concise summary of its performance.

- XceptionNet: This model records an accuracy of 95%, precision of 95%, recall of 95%, and an F1-score of 95%, illustrating steady and balanced performance in all metrics.
- EfficientNetB0: This model slightly surpasses XceptionNet with an accuracy of 96%, precision of 96%, recall of 96%, and an F1-score of 96%. These outcomes emphasize EfficientNetB0's strength and capability to classify malware effectively with minimal errors.
- ResNet50: Like XceptionNet, ResNet50 achieves an accuracy of 95%, precision of 95%, recall of 95%, and an F1-score of 95%, indicating competitive performance but falling slightly behind EfficientNetB0 in overall metrics.

EfficientNetB0 is the top-performing model, obtaining the highest scores across all metrics. This comparison illustrates that while all models perform strongly in malware classification tasks, EfficientNetB0 offers a slight edge in accuracy, precision, recall, and F1-score.

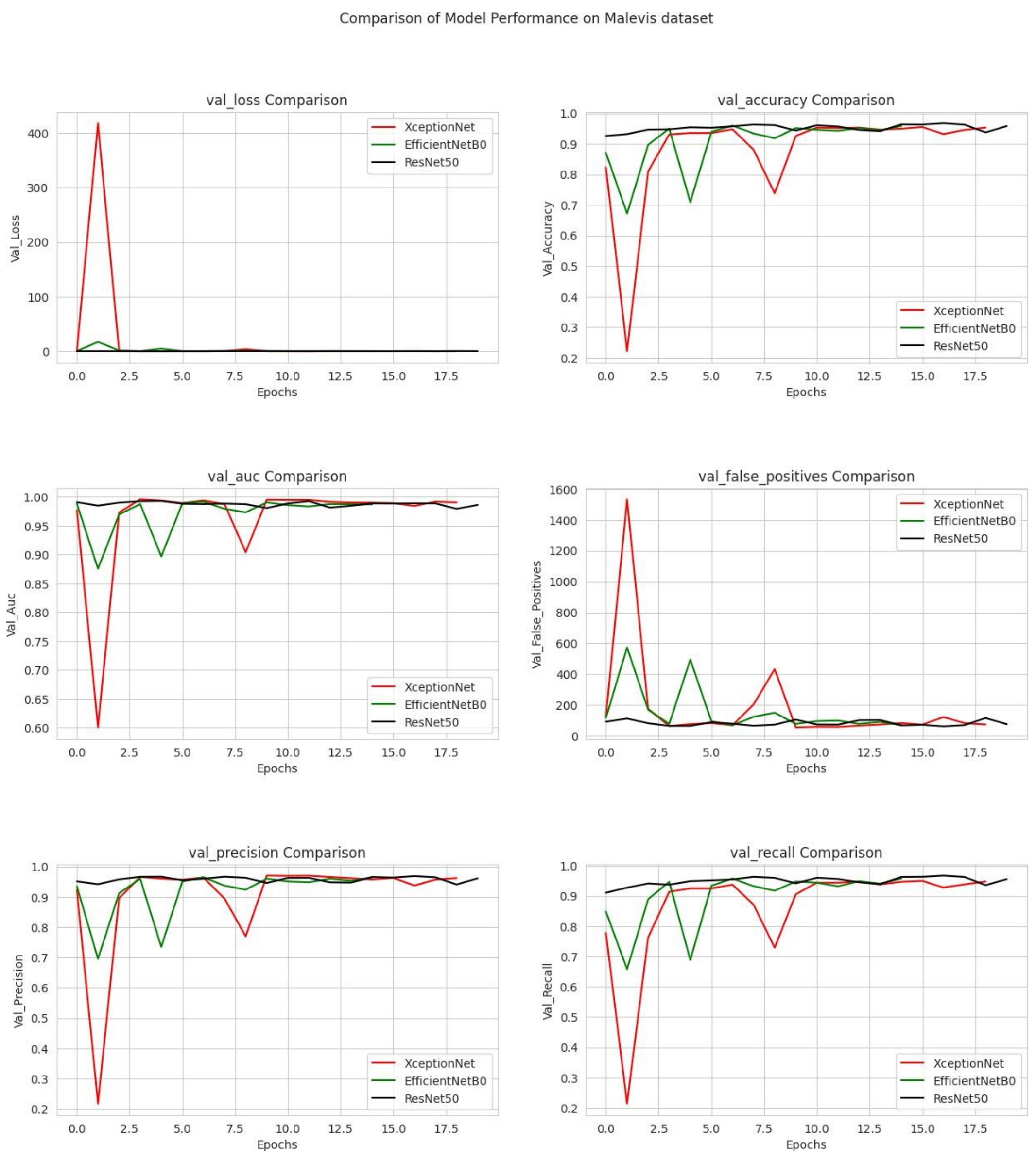


Figure 14: Comparison of training performance of models

The graphs present a comparison of the performance of three models — XceptionNet, EfficientNetB0, and ResNet50 based on various metrics using the validation set from the Malevis dataset. The metrics displayed are as follows:

1. Validation Loss: Initially, XceptionNet shows a validation loss that is significantly higher than that of EfficientNetB0 and ResNet50 but stabilizes after several epochs. ResNet50 consistently exhibits the lowest loss throughout the entire training process.

2. Validation Accuracy: ResNet50 reliably achieves the highest accuracy, closely followed by EfficientNetB0. XceptionNet starts with lower accuracy but shows improvement as the epochs progress.

3. Validation AUC: Each of the three models attains a high AUC, with ResNet50 marginally surpassing the others, which indicates strong discrimination ability.

4. Validation False Positives: In the early stages, XceptionNet experiences a rise in false positives, whereas ResNet50 maintains the lowest rate of false positives, demonstrating greater robustness in this area.
5. Validation Precision: ResNet50 consistently achieves the highest precision rate, trailed by EfficientNetB0, while XceptionNet shows gradual improvement after the initial epochs.

6. Validation Recall: Both ResNet50 and EfficientNetB0 display high recall values, whereas XceptionNet starts with a lower recall but progressively improves over time.

6.5 Discussion

The classification results of XceptionNet, EfficientNetB0 and ResNet50 in the malware detection task provide detailed insights into their strengths and weaknesses based on metrics such as accuracy, precision, recall, and F1 score. EfficientNetB0 achieved a maximum accuracy of 96%, outperforming both Xception and ResNet50, which also achieved 95% accuracy. This slight advantage gives EfficientNetB0 the most balanced model for malware classification tasks especially for imbalanced data sets.
Xception performs well, with balanced precision and recall between classes. This makes it ideal for consistent classification tasks. Its strengths lie in high precision for specific malware types. This ensured fewer false positives. However, it had difficulty remembering challenging categories such as Nesta and Sality, where its performance was slightly lower.
EfficientNetB0, with an optimized architecture, exhibits best-in-class recall in most classes. This makes it highly reliable in detecting hard-to-classify malware. F1 scores are consistently high, especially for difficult classes. This proves suitable for unbalanced datasets; however, it shows slightly reduced accuracy for classes such as injectors and automation. This indicates a susceptibility to false positives in some cases.
ResNet50 has excellent accuracy. This is especially true for important, high-risk malware classes such as Amonetize and HackKMS, making it ideal for applications where reducing false positives is important. However, recall is low for some classes, such as Sality and Autorun. This may limit its usefulness in situations where recall is emphasized. In terms of performance, EfficientNetB0 has become the most lightweight computationally efficient model.

7 Conclusion and Future Work

This project investigates malware detection using three advanced deep learning architectures: XceptionNet, EfficientNetB0, and ResNet50. The primary goal is to identify the most effective model for malware classification. Research objectives Include Evaluating these models on the malware dataset, analysing precision, recall, F1 score, etc., and determining the suitability of models for various use cases. These objectives were successfully achieved, with EfficientNetB0 emerging as the most efficient and balanced model overall.

EfficientNetB0 Demonstrated the highest accuracy (96%) and superior recall. This makes it ideal for situations involving imbalanced datasets. Xception demonstrates consistent performance with balanced precision and recall, ideal for detecting general malware. ResNet50 has excellent precision for critical high-risk malware categories, minimizing false positives. However, both Xception and ResNet50 struggle with challenging classes such as "Sality" and "Neshta", highlighting several issues for further optimization

These findings have important implications for cyber security especially for designing efficient, accurate, and adaptive malware detection systems. The model's strengths and limitations highlight the need for tailored solutions. It depends on whether precision, recall, or computational efficiency is prioritized. Study's reliance on malware datasets. Although it is strong, but it shows limitations as a single dataset may not fully reflect the reality - the diversity of malware around the world.

7.1Future work

This research lays a solid foundation for the benefits of deep learning in malware detection. Future work could focus on expanding the dataset to include a variety of real-world examples. This will improve the durability and general appearance of the model. Developing an ensemble method that combines the strengths of Xception, EfficientNetB0, and ResNet50 can also increase overall performance which is especially good for challenging classifications. Real-time planning is another important area for exploration, which is research into optimizing these models for real-time malware detection systems. Especially in resource-constrained environments, such as mobile or edge devices, it may greatly increase its practical utility.
Finally, the commercial potential is also significant. A lightweight and powerful framework that uses EfficientNetB0 could be developed for industries that need a scalable malware detection solution. This work, therefore, lays a strong foundation for improving the future of malware detection, highlighting practical applications and opportunities for further academic exploration.

References

[1] Tahir, R. (2018). A Study on Malware and Malware Detection Techniques. International Journal of Education and Management Engineering, 8(2), 20–30. Available at https://doi.org/10.5815/ijeme.2018.02.03

[2] Venugopal, D., & Hu, G. (2008). Efficient Signature Based Malware Detection on Mobile Devices. Mobile Information Systems, 4(1), 33–49. Available at https://doi.org/10.1155/2008/712353

[3] Abiola, A. M., & Marhusin, M. F. (2018). Signature-based malware detection using sequences of N-grams. Int. J. Eng. Technol, 7(4), 120-125.

[4] Chew, C. J., & Kumar, V. (2019). Behaviour based ransomware detection.

[5] Lindorfer, M., Kolbitsch, C., & Milani Comparetti, P. (2011). Detecting Environment-Sensitive Malware. Lecture Notes in Computer Science, 338–357. Available at https://doi.org/10.1007/978-3-642-23644-0_18

[6] Firdausi, I., Lim, C., Erwin, A., & Nugroho, A. S. (2010, December 1). Analysis of Machine Learning Techniques Used in Behavior-Based Malware Detection. Available at https://doi.org/10.1109/ACT.2010.33

[7] Narayanan, B. N., & Davuluru, V. S. P. (2020). Ensemble Malware Classification System Using Deep Neural Networks. Electronics, 9(5), 721. Available at https://doi.org/10.3390/electronics9050721

[8] Saxe, J., & Berlin, K. (2015b). Deep neural network-based malware detection using two dimensional binary program features. 2015 10th International Conference on Malicious and Unwanted Software (MALWARE). Available at https://doi.org/10.1109/malware.2015.7413680

[9] Jha, S., Prashar, D., Long, H. V., & Taniar, D. (2020). Recurrent Neural Network for Detecting Malware. Computers & Security, 102037. Available at https://doi.org/10.1016/j.cose.2020.102037

[10] Mathew, J., & M. A. Ajay Kumara. (2019). API Call Based Malware Detection Approach Using Recurrent Neural Network—LSTM. Advances in Intelligent Systems and Computing, 87–99. Available at https://doi.org/10.1007/978-3-030-16657-1_9

[11] Haque, A. H., Jahin, Labiba Ifrit, Sheikh, K., Mahmud, T. S., & Tasnia, M. (2024). MalFam: a comprehensive study on malware families with state-of-the-art CNN architectures with classifications and XAI. Bracu.ac.bd. id: 20101453

[12] Raman Kumar, M. G. (2022). IVMCT: Image Visualization based Multiclass Malware Classification using Transfer Learning. Mathematical Statistician and Engineering Applications, 71(2). Available at https://doi.org/10.17762/msea.v71i2.65

[13] Aslan, O., & Yilmaz, A. A. (2021). A New Malware Classification Framework Based on Deep Learning Algorithms. IEEE Access, 9, 87936–87951. Available at https://doi.org/10.1109/access.2021.3089586

[14] Naeem, H., Ullah, F., Naeem, M. R., Khalid, S., Vasan, D., Jabbar, S., & Saeed, S. (2020). Malware detection in industrial internet of things based on hybrid image visualization and deep learning model. Ad Hoc Networks, 105, 102154. Available at https://doi.org/10.1016/j.adhoc.2020.102154

[15] Yadav, P., Menon, N., Ravi, V., Vishvanathan, S., & Pham, T. D. (2022). EfficientNet convolutional neural networks-based Android malware detection. Computers & Security, 115, 102622. Available at https://doi.org/10.1016/j.cose.2022.102622

[16] Basak, M., & Han, M.-M. (2024). CyberSentinel: A Transparent Defense Framework for Malware Detection in High-Stakes Operational Environments. Sensors, 24(11), 3406–3406. Available at https://doi.org/10.3390/s24113406

[17] Ashawa, M., Nsikak Owoh, Hosseinzadeh, S., & Osamor, J. (2024). Enhanced Image-Based Malware Classification Using Transformer-Based Convolutional Neural Networks (CNNs). Electronics, 13(20), 4081–4081. Available at https://doi.org/10.3390/electronics13204081

[18] GeeksforGeeks, "Confusion Matrix in Machine Learning," GeeksforGeeks. Available at https://www.geeksforgeeks.org/confusion-matrix-machine-learning

[19] Chollet, F. (2017). Xception: Deep Learning With Depthwise Separable Convolutions. Retrieved from openaccess.thecvf.com website. Available at https://openaccess.thecvf.com/content_cvpr_2017/html/Chollet_Xception_Deep_Learning_CVPR_2017_paper.html

[20] Rezende, E., Ruppert, G., Carvalho, T., Ramos, F., & de Geus, P. (2017). Malicious Software Classification Using Transfer Learning of ResNet-50 Deep Neural Network. 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA). Available at https://doi.org/10.1109/icmla.2017.00-19

[21] “EfficientNet: A Breakthrough in Machine Learning Model Architecture - Javatpoint,” www.javatpoint.com. Available at https://www.javatpoint.com/efficientnet-a-breakthrough-in-machine-learning-model-architecture

[22] IBM, “Fine Tuning,” Ibm.com, Mar. 15, 2024. Available at https://www.ibm.com/think/topics/fine-tuning

[23] THREATS IN CYBERSPACE. (2023). Retrieved January 25, 2025, from Google Books website. Available at https://books.google.co.in/books?hl=en&lr=&id=EuzaEAAAQBAJ&oi=fnd&pg=PA2&dq=According+to+India%27s+Cyber+Threat+Report+2023

[24] Aslan, Ö. A., & Samet, R. (2020). A Comprehensive Review on Malware Detection Approaches. IEEE Access, 8(1), 6249–6271. Available at https://doi.org/10.1109/ACCESS.2019.2963724

[25] Prakash Niraj, S., & Tiwari, A. (2022). Research Journal of Engineering Technology and Medical Sciences, 05(04), 2582–6212. Available at http://www.rjetm.in/RJETM/Vol05_Issue04/Performance%20Analysis%20of%20Signature%20Based%20and%20Behavior%20Based%20Malware%20Detection.pdf

[26] Venkatraman, S., Alazab, M., & Vinayakumar, R. (2019). A hybrid deep learning image-based analysis for effective malware detection. Journal of Information Security and Applications, 47, 377–389. Available at https://doi.org/10.1016/j.jisa.2019.06.006