Harnessing Deep Features and Machine Learning for Malware Image Classification

MSc Research Project
Research Project

Poojitha Thokala
Student ID: x22230424

School of Computing
National College of Ireland

Supervisor: Vikas Sahni

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: ……. Poojitha Thokala ……………………………………………………………………

Student ID: ………x22230424……………………………………………………………………….……

Programme: ……… Research Project…………………………… **Year:** ……2024…………..

Module: … MSc Research Project…………………………………………………………….……

Supervisor: ……… Vikas Sahni……………………………………………………………………….………

Submission Due Date: ……… 12/12/2024………………………………………………………………….………

Project Title: Harnessing Deep Features and Machine Learning for Malware Detection

Word Count: ………6762…………… **Page Count**……………22……………………….……..

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: ……………………………Poojitha Thokala……………………………………………

Date: ……………………………11/12/2024…………………………………………………

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):	

Harnessing Deep Features and Machine Learning for Malware Detection

Thokala Poojitha
X22230424

Abstract

The increase of sophisticated malware poses a critical threat to individuals, organizations, and critical infrastructures, which highlights the urgent need for better and robust detection mechanisms. Traditional malware detection methods such as signature-based and heuristic methods, struggle to handle particularly in case of constantly evolving nature of malware, obfuscation, polymorphism techniques and imbalanced datasets. This often results in poor performance leaving the systems vulnerable.

This method first transforms malware binaries into grayscale images, allowing CNN to extract key spatial features. Support Vector Machines (SVMs) and Random Forests (RFs) are used to classify the features which are further combined through a learning strategy to improve accuracy and robustness. This study shows the advantages of combining deep learning for automated feature extraction with traditional machine learning for precise classification.

The results show that this hybrid method could be a practical and scalable solution for modern malware detection resulting in better accuracy and efficiency. This methodology was validated on the Malimg dataset, which has 25 malware families, achieving accuracy of 97.2%. Strong predictive performance is shown through the confusion matrices, but minor trends of misclassification, that require further refinement are also shown.

1 Introduction

Malware is a major threat to the global cybersecurity, and is malicious software designed to interrupt, damage or gain unauthorized access to systems. Since it is ever changing, new malware (polymorphic and zero day) does not fit in a signature-based techniques paradigm. However, these conventional approaches usually fail to recognize new malware, resulting in large security gaps. This motivated recent research on image-based malware detection which utilize visual patterns to improve detection accuracy. Traditional malware detection methods, for example heuristic and behavioral analysis, have limitations, and Nataraj et al. (2011) showed that by converting malware binaries to grayscale images, researchers can better detect malware families by identifying distinct visual patterns in the grayscale images. Obfuscated or new malware causes false negatives for signature-based approaches and the behavioral approaches need a lot of computational resources and risk privacy issues. Therefore, researchers have started to work with image-based techniques that can extract more deeper insights from the patterns of malware. However, according to Kalash et al. (2018), image-based obfuscation methods handle obfuscation challenges more effectively since the structural integrity of malware as being preserved in visual representation helps better feature extraction and detection.

Malware detection based on image provides a new approach by converting malware binaries into grayscale images for automated visual pattern analysis. This approach outperforms in bypassing obfuscation techniques and is amenable to large datasets. Further, it promises to be effective in zero-day malware detection, since its mechanism relies on pattern recognition rather than pre-defined signatures. Nataraj et al. (2011) has shown that this method enables researchers to extract meaningful features from image data that are vital for separating malware families. Moreover, Coull and Gardner (2019) pointed out that this method makes possible automated feature learning, which is advantageous to automatic methods over manual ones. The Malimg dataset, a common benchmark in malware detection research, includes grayscale images of 25 malware families. Each image is the binary content of a malware sample in a 2D array. The dataset is balanced with 40 instances per class and can hence be used as a benchmark to test advanced classification methods. In accordance with Bakour, K. and Ünver, H.M., (2021), the Malimg dataset has allowed researchers to test the effectiveness of new detection techniques that have significantly helped malware detection.

This project integrates Convolutional Neural Networks (CNN) for deep feature extraction with machine learning classifiers such as Support Vector Machines (SVM) and Random Forest (RF) for malware classification. The CNN learns hierarchical features automatically for malware image, while SVM and RF classifiers guarantee the robustness of classification. The goal is to attain high accuracy and recall and low precision while minimizing the generalization error rates. Kalash et al. (2018) show that when CNNs are combined with ML classifiers, it improves overall performance (especially when it comes to identifying closely related malware families), and that using CNNs for feature extraction and ML classifiers for classification has several advantages. The automated feature extraction by CNNs removes manual engineering, and these models still manage to capture spatial patterns. SVM classifiers are good in high dimensional space, while Random Forest learns by ensemble, and is resilient to overfitting. In Vasan, D., et al (2020), they emphasized that CNNs, in combination with the ML classifiers, achieve a better generalization and accuracy in the malware detection tasks. This project takes advantage of these advantages to present a malware classification framework that is efficient and scalable using the Malimg dataset and opens the door to enhanced security in real world applications.

Research Question

How effective is the integration of CNNs for deep feature extraction with machine learning classifiers in accurately classifying malware families?

1.1 Report Structure

In this paper, Section 2 reviews previous studies related to malware analysis using CNNs and Deep Learning and some ensemble techniques for malware detection and classification. In Section 3, the research methodology has all the planned procedure and evaluation methodology. The proposed method design specification and architecture is discussed in section 4 along with algorithm and functionality of the model. Implementation details are mentioned in section 5 along with outputs produced and tools and languages used to produce the results. In section 6 the evaluation that includes comprehensive and rigorous analysis is discussed. Section 7 has conclusion along with scope for future work followed by references.

2 Related work

There are a number of studies on deep learning models, traditional machine learning approaches, and hybrid methods which use both. But the need to come up with new ideas and detecting malware automatically is also increasing. Advances in accuracy, efficiency and the computational needs of each approach are highlighted, providing interesting insights into the development of image classification systems.

2.1 CNN-Based Feature Extraction for Medical Image Classification

A few studies featured the use of CNN architectures to extract features from medical images to improve diagnosis performance. In the AlSaeed and Omar (2022), they created an Alzheimer's disease diagnosis model using ResNet-50 for automatic feature extraction from MRI images. They compared three classification methods: ResNet50 with Softmax gave the highest accuracy, compared to traditional classifiers such as SVM and Random Forest. A similar scenario was also proposed by Assiri (2020), where logistic regression, SVM, and multilayer perceptron classifiers were integrated to form an ensemble classification method to improve diagnostic performance of breast cancer detection via a majority voting mechanism. Kang et al. (2021) extracted deep features from MRI images for brain tumor classification using CNNs, and obtained high accuracy via an ensemble of features from models such as DenseNet-169 and Inception V3. A modified DenseNet201 model for lung cancer detection from chest CT scans was developed by Lanjewar et al. (2023) using features selection methods to improve classification performance and reach high accuracy on test datasets. The studies in this group demonstrate the effectiveness of CNNs in extracting useful features from medical images and using them together with ML classifiers for better diagnosis.

2.2 Deep Learning and Hybrid Approaches for Malware Detection

A large body of literature investigating CNNs and hybrid models for image-based malware detection has been reported. We demonstrate that these models convert binary malware files into 2D images that are suitable for effective feature extraction and classification. One of the first early works introducing the concept of visualizing malware as images is that of Nataraj et al. (2011), which provides the ability to perform novel analytical and classification tasks based on visual patterns in malware families. It laid the basis for using image-based technique in malware detection, and later, Kalash et al. (2018) also used deep convolutional neural networks (CNNs) to classify malware based on an image-based technique proposed by Nataraj et al. Their study proves that CNNs can automatically extract hierarchical features from malware images for better classification performance. Additionally, Kumar (2021) and El-Shafai et al. (2021) suggested deep learning frameworks that adopted transfer learning and fine tuning of CNN models like ResNet-50, VGG16, or DenseNet for malware classification and demonstrated that the performance of CNNs can help improve malware detection accuracy with the help of standard machine learning classifiers. In addition, Patil et al. (2021) and Shaukat et al. (2022) worked on improving the robustness of malware detection systems against adversarial attacks leading to a reduction in evasion rates. Go et al. (2020) and Xiao et al. (2021) further utilized CNNs (ResNeXt and VGG16) for visualization-based malware classification to address this problem of detecting diverse and evolving malware variants.

The use of images obtained from binaries proves to be effective when it comes to detecting malware, this combined with the ability to transfer learning the features to achieve a high classification accuracy -as malware is continuously evolving- can lead to a better prediction for new malware variants that could arise.

2.3 Ensemble Learning Techniques for Image Classification

Several studies were devoted to a prominent strategy, called ensemble learning, that improves image classification accuracy by combining strengths of multiple models. An ensemble classification-based malware detection methodology based on an application of dense neural networks and CNNs in a two-stage manner was suggested by Damaševičius et al. (2021). In contrast, their method significantly boosted detection performance over individual models. A hybrid ensemble model for identifying lung and colon cancer was proposed by Talukder et al. (2022) that uses a combination of deep learning feature extraction and machine learning classifiers such as Random Forest and SVM with higher accuracy for different types of cancer. Bakour and Ünver (2021) used ensemble voting mechanism in malware detection, which combines classifiers like Random Forest, K-nearest Neighbors and Decision Trees, which enhances decision making and classification accuracy. This study shows how ensemble learning can be used to take advantage of the diversity of model capabilities for improved classification.

2.4 Innovative Techniques in Feature Extraction and Classification

Multiple studies investigating feature extraction and classification methods have utilized deep learning and traditional machine learning. Barbhuiya et al. (2021) have developed a hand gesture recognition system for American sign language using modified AlexNet and VGG16 architecture and classified using SVM to get impressive accuracy. The DenseNet model with a reweighted class balanced loss function proposed by Hemalatha et al. (2021) is a malware detection system that solves the problem of imbalanced malware classification datasets. IMCFN, a novel image-based malware classification method via fine-tuned CNN architecture, proposed by Vasan et al. (2020) showed resilience against obfuscation techniques. Fatani et al. (2021) proposed an intrusion detection system for IoT based on deep learning and swarm intelligence that demonstrates the promise for incorporating an advanced optimization technique together with feature extraction. Lastly, Son et al. (2022) proposed resizing malware images horizontally before feeding them into classifiers such as CNN and SVM, thus reducing computational requirements while still keeping a high accuracy. In these studies, it was presented how deep learning-based feature extraction and efficient classification methods can be combined in innovative ways to solve hard image classification problems.

While deep learning and machine learning are typically applied separately or combined together in complicated ensemble methods, a clear gap exists between the current approaches to image classification. Most existing studies use deep learning models such as CNNs for both feature extraction and classification, or employ ensemble techniques that are computationally expensive. One promising but thus far underexplored approach is to use deep learning models to efficiently extract features and use traditional machine learning classifiers to do the final classification. This could provide a simpler, resource efficient solution for image classification at reasonable accuracy without achieving such complicated ensemble frameworks.

3 Research Methodology

This section shows the process followed for this research; the data preparation, model training, evaluation methodology, and techniques for analysis. Every stage is carefully planned so as to conform to scientific rigor and rationalized on the basis of its comparison to related work.

3.1 Research Procedure

This research procedure describes the systematic way that malware detection is carried out, beginning with dataset preparation, then extracting feature using a pre trained CNN model and finally classifying by using ensemble techniques to improve prediction accuracy and robustness.

Dataset: The Malimg dataset[1] is a well-known malware dataset used for classification, which contains 9,339 grayscale images over 25 malware families obtained by converting malware binaries to images. For this study, 200 samples per family were selected in order to maintain balance and efficiency.

The Malimg dataset is chosen due to its representation of grayscale image of malware binaries, which provides better feature extraction and classification. The malware is diverse across 25 malware families, and it is widely used in research, thus making it reliable and usable to test malware detection models.

Dataset Preparation: The dataset for this study is the malware samples represented as extracted CNN features, with a balanced dataset of 5,000 samples and 513 attributes (512 features + target label) in 25 malware classes. The dataset was normalized by changing pixel values to the range of [0, 1] and split the data into 80% training and 20% testing subsets using stratified sampling to keep class distribution proportional.

Feature Extraction: Inspired by previous research indicating that CNNs can convert high dimensional image data into compact feature vectors, feature extraction was undertaken using a pre trained CNN model, modified to remove the final classification layer. Images were

[1] https://paperswithcode.com/dataset/malimg

resized to 256 × 256 pixels, normalized, converted to arrays, and used to feed the images through the model, capturing feature vectors from the penultimate layer for classification.

Model Training: The extracted CNN features were used to train two machine learning models SVM and RF. Both the models are trained with default hyperparameters using Scikit-learn and saved them for inference.

Figure 1 shows the system architecture of the malware detection process consisting of several interconnected stages. For the first step, data pre-processing is used to clean, normalize and feature scale our malware dataset and visualize to understand the characteristics of our dataset so that it can be ready for analysis. In the next phase, feature extraction, a CNN is used to learn high level features from the dataset using its ability to process structured and unstructured data. Next, the extracted features are used as input into classification models: RF and SVM whose aim is to predict malware classes independently based on the learned patterns. The outputs from the models above are refined using an ensemble technique which combines RF and SVM predictions to reduce bias and variance to improve accuracy and robustness. The performance of the ensemble model is finally compared using the result analysis.

3.2 Equipment and Tools Used

Google Colab: Colab offers access to GPU and TPUs that can handle high computational tasks compared to other options like PC with limited storage and GPU and AWS Sage maker with high costs. Colab is a cloud-based environment which allows us to train and run large models without using expensive hardware. It is scalable and user friendly.

Software: Python was the primary programming language that was used in the setup. TensorFlow is used to perform CNN based feature extraction, Scikit learn for training Support Vector Classifier (SVC) and Random Forest (RF) models and Seaborn and Matplotlib for creating data visualizations. Such combination of these tools ensured a robust and flexible environment for implementation and evaluation of the proposed methodology.

3.3 Evaluation Methodology

Model Validation: The models were evaluated on unseen data, with the data representing the distribution, using stratified train-test splitting. Validation accuracy and other performance metrics were calculated on a testing set (20% of dataset).

Evaluation Metrics: To assess the effectiveness of the models, standard classification evaluation metrics were used, including:

Accuracy: Measures overall classification correctness.

Precision, Recall, and F1-Score: Calculated per class to evaluate model performance for imbalanced data.

Confusion Matrix: A visualization tool to analyze classification errors.

3.4 Data Analysis

Raw Data Processing: To be able to use machine learning algorithms, the CNN features were converted to CSV format. Balance was maintained and potential bias when training the model was minimized by examining class distributions.

Statistical Techniques: Macro-averaged F1-scores and accuracy were used for the assessment of model performance. The macro macro-averaged F1 scores equally take into account both precision and recall across all classes, making it an excellent choice for imbalanced datasets such as the data used in the present work. Accuracy is the total percent of correct predictions. The effectiveness of both the Support Vector Classifier (SVC) and Random Forest (RF) models were calculated for these metrics in terms of 25 malware classes.

3.5 Steps from Data Collection to Final Results

Data Collection: Images representing malware classes were gathered and labeled.

Feature Extraction: CNN-derived feature vectors were generated and stored in CSV format.

Model Training: SVC and RF models were trained separately on the training set.

Inference and Evaluation: The testing set was used to predict labels and evaluate performance. An ensemble method combined predictions from both models.

Result Visualization: Confusion matrices and classification reports were generated to interpret and compare model outputs.

This methodology ensures a comprehensive and transparent process, enabling reproducibility and validity of the results.

4 Design Specification

Ensemble Model: An ensemble approach is being implemented, averaging SVC and RF probabilities. In this work, they use the complementary strength of models used to improve classification accuracy and robustness.

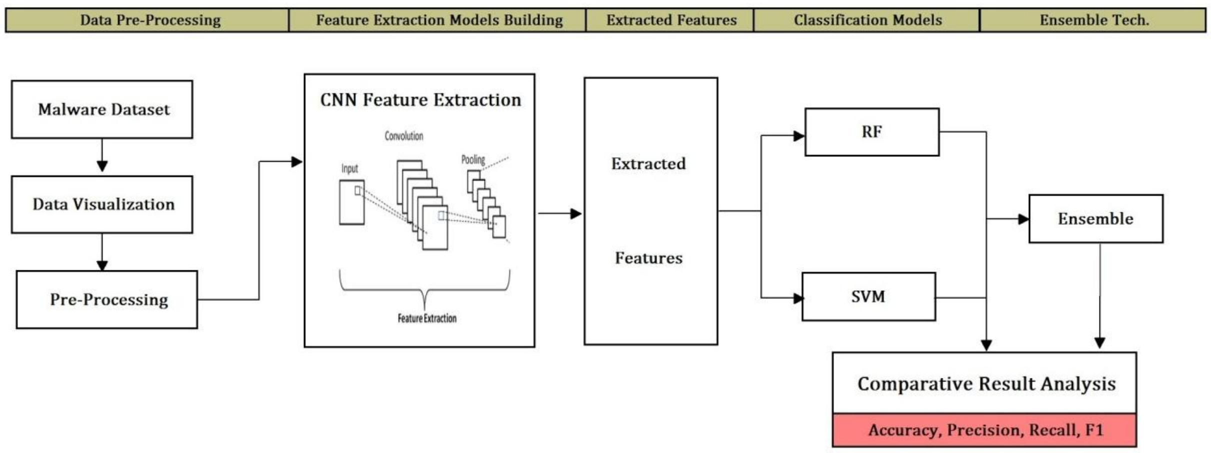


Figure 1 : System Architecture

A new hybrid method is proposed as shown in the algorithm 3 where after training SVM and RF classifiers an average probability is calculated on the data which helps us predict the results.

Algorithm 3 Classification and Ensemble Learning

```
1: load CNN_features.csv
2: split dataset into training and testing sets
3: train SVM classifier on training set
4: train RF classifier on training set
5: for each test instance do
6:     predict probabilities with SVM and RF
7:     calculate ensemble prediction by averaging probabilities
8:     assign final class based on highest probability
9: end for
10: evaluate ensemble predictions using accuracy, precision, recall, and F1-score
```

5 Implementation

In this section, data preprocessing process, feature extraction and model implementation are explained in detail.

5.1 Dataset Overview

The **Malimg dataset** containing 9,339 malware samples represented as 256x256 grayscale images. The malware dataset is split into 25 malware families, with different number of images in each family. Additional details are mentioned in the configuration file.

These are malware families that will have different threats, such as viruses, trojans, worms, and spyware. The dataset contains 25 families of malware, and each image in the dataset represents a specific malware sample, that belongs to one of those 25 families.

5.2 Preprocessing of the Dataset

Preprocessing is an important step of data conversion from raw data to a format that can be useful for features extraction and classification. The following steps were implemented to process the dataset:

5.3 Grayscale Conversion

All the Malware samples in the dataset were initially stored as binary files, which were converted into grayscale images. This transformation enabled the representation of the malware's binary structure as visual patterns that were then fed into a convolutional neural network (CNN).

5.4 Image Resizing

To ensure consistency and compatibility with the CNN model all the images were resized to a uniform resolution of 256 × 256 pixels. Resizing this ensures that all input data has the same dimensions and therefore can be fed into a deep learning model.

5.5 Class Distribution Analysis

The training and testing data were analyzed to ensure that no class is under represented or over represented. This was critically important so as to avoid model bias towards the larger classes. A bar chart below in Figure 3 presents the class distribution in the dataset after preprocessing, where each class is limited to 200 samples to achieve balance.

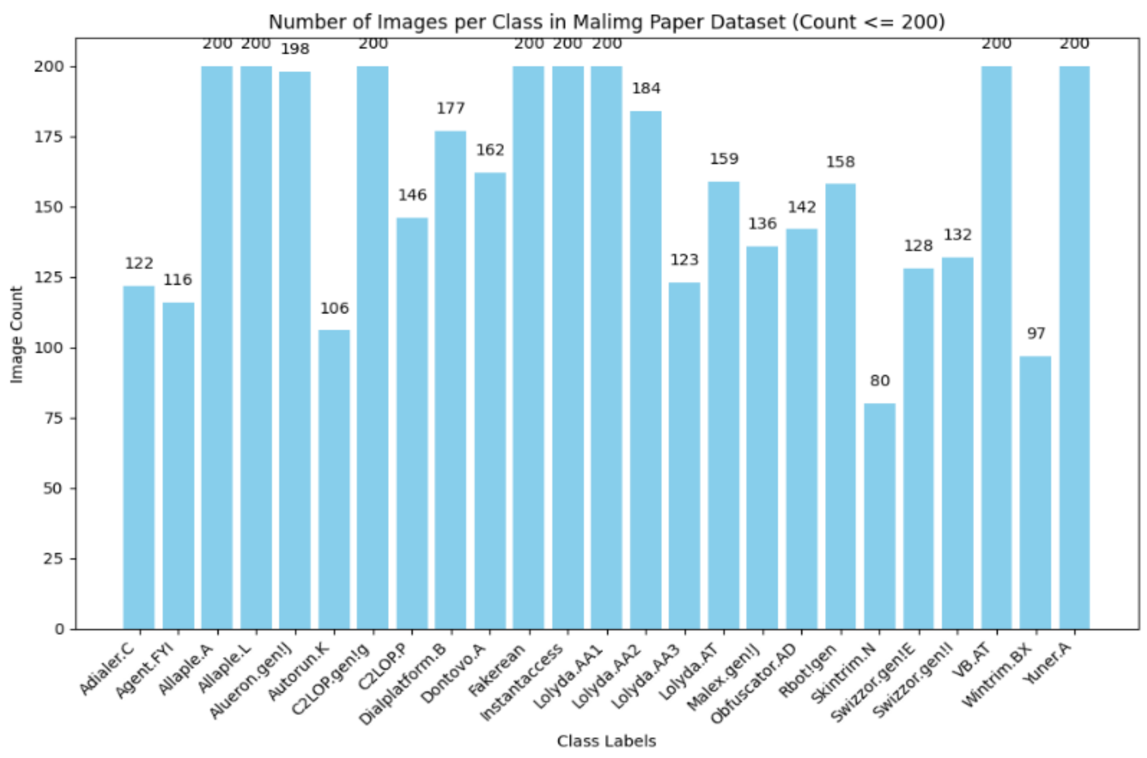


Figure 2: Number of Images per Class in Malimg Dataset

This bar chart indicates each class is now balanced with a max of 200 samples, which makes sure the classifier will be trained on a balanced dataset.

5.6 Normalization

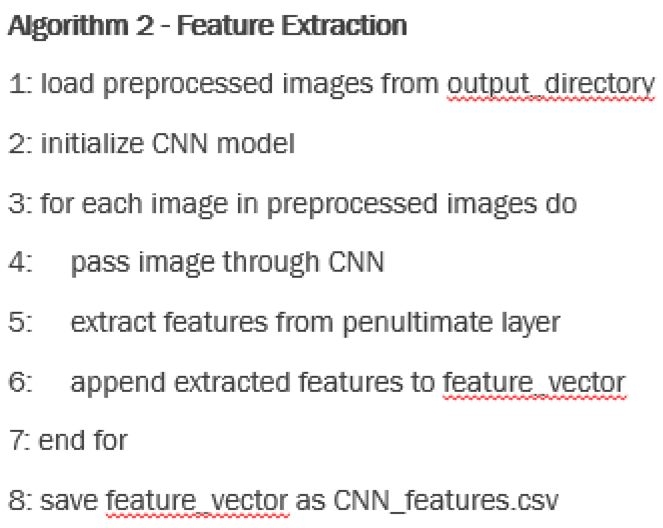
Each image was normalized by normalizing the pixel values to [0, 1] by dividing each pixel value by 255. On this normalization we facilitate faster convergence at model training and we avoid putting greater value on larger pixel values.

5.7 Data Splitting

The dataset was split to training and testing after preprocessing. 80% of the dataset for training and 20% for testing with stratified sampling to ensure similar distribution of malware classes on the training and testing datasets. It prevents class imbalance from influencing the performance of model.

5.8 Feature Extraction (CNN Model)

Features for the images were extracted using a pre-trained Convolutional Neural Network (CNN). In particular, feature vectors were extracted from the penultimate layer of the CNN. The representations of these vectors are these essential characteristics of each image and serve as input to the machine learning models. The image files were so unique to malware that pretrained a CNN on an existing large dataset and then fine-tuned its weight to get accustomed to the characteristics of these images. The features were extracted and stored in CSV format that each image is represented by a feature vector.



Algorithm 2 - Feature Extraction

```
1: load preprocessed images from output_directory
2: initialize CNN model
3: for each image in preprocessed images do
4:      pass image through CNN
5:      extract features from penultimate layer
6:      append extracted features to feature_vector
7: end for
8: save feature_vector as CNN_features.csv
```

Figure 3:Feature Extraction

Analyzing Filtered Actions Feature Using Bar Chart: To further evaluate the quality and characteristics of extracted features, a bar chart of the frequency of the most relevant features was visualized (Figure 5). Understanding the distribution of feature importance across the dataset this provides.

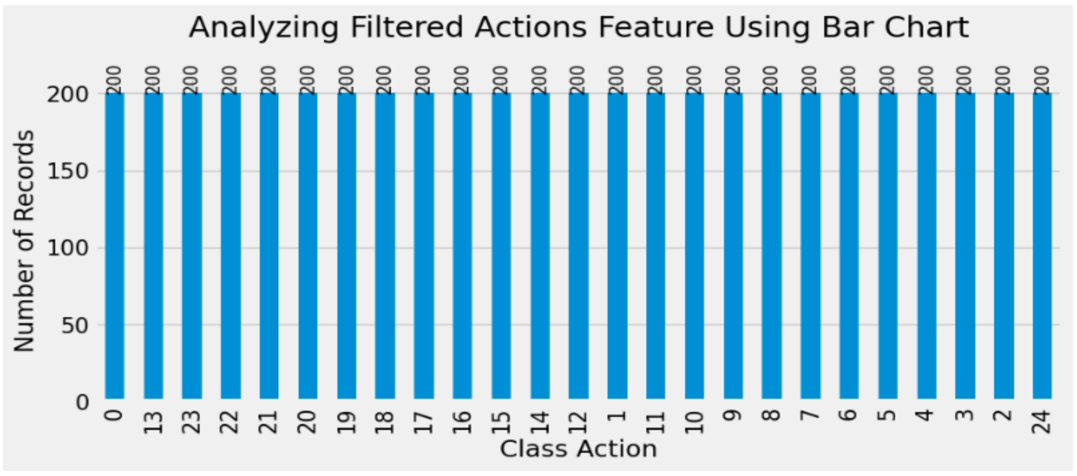


Figure 4: Analyzing Filtered Actions Feature Using Bar Chart

5.9 Model Implementation

All detailed steps taken towards implementing the machine learning models for the purpose of malware classification will be discussed in this section. The process of training two models on the extracted CNN features, Support Vector Classifier (SVC), Random Forest (RF), and evaluation of their performance, is included.

Support Vector Classifier (SVC): An effective machine learning algorithm especially for high dimensional data. SVC finds the optimal hyperplane which maximizes the margin between classes. However, when the feature space is large, SVC is the best option for the extracted CNN features from the malware images. Steps for SVC Implementation as follows.

Loading the Data: In the feature extraction phase, SVC model is trained using the features which are extracted into a CSV format.

Training the Model: At last, the SVC model is trained using the extracted features from the particle swarm optimization algorithm with the use of Scikit-learn library. The SVC class with default hyperparameters (radial basis function kernel) is used for training.

Saving the Model: Once trained, the model is saved for future inference.

Explanation: Starting with the Radial Basis Function (RBF) kernel, frequently used for nonlinear data. If train_test_split function is applied with training and testing split with an 80/20 ratio, the dataset would become split to its training and test subsets. The result is then trained using fit() function on the training data, and using predict() on the test data. In order to evaluate the performance of the model, accuracy and macro averaged F1 score are printed at the end to see how well the model classifies malware samples.

Random Forest Classifier (RF): The ensemble method that constructs a random forest of decision trees that is put together to increase classification accuracy. Each decision tree is independent of each other, they make an independent prediction and the final prediction is the majority vote. Random Forest is particularly useful for handling complex, high-dimensional datasets like the ones we have in this project. Steps for RF Implementation:

Loading the Data: Similarly, the RF model is trained with feature data from the CSV file, in the same way as the SVC model.

Training the Model: To train the model on the extracted features, RandomForestClassifier from Scikit learn is being used.

Saving the Model: Afterward, the RF model is saved for future use.

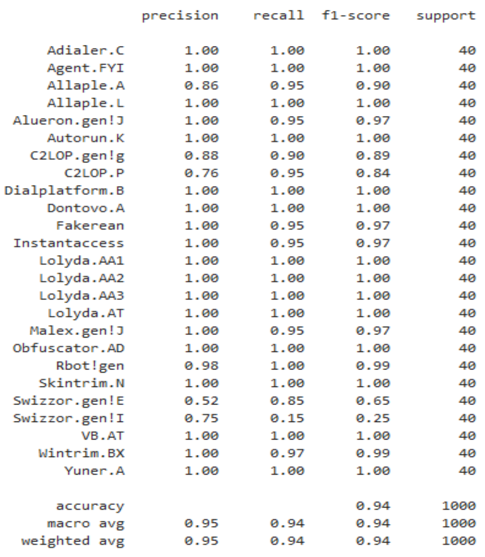
Explanation: RandomForestClassifier with 100 trees (n_estimators=100) is initialized as a typical number of trees to have robust performance and set random_state=42 to have reproducibility. Next, with train_test_split, we split the dataset into train (using fit() to fit it to the data) and test sets. Accuracy and macro averaged F1 score are used to evaluate the performance of the model, in order to assess the effectiveness of the model for classifying the malware samples.

Model Evaluation: Accuracy and macro averaged F1 score, which balances precision and recall across all classes and overcomes the problem of imbalanced dataset were used as a base of model evaluation.

6 Evaluation and inferencing

6.1 Result Analysis

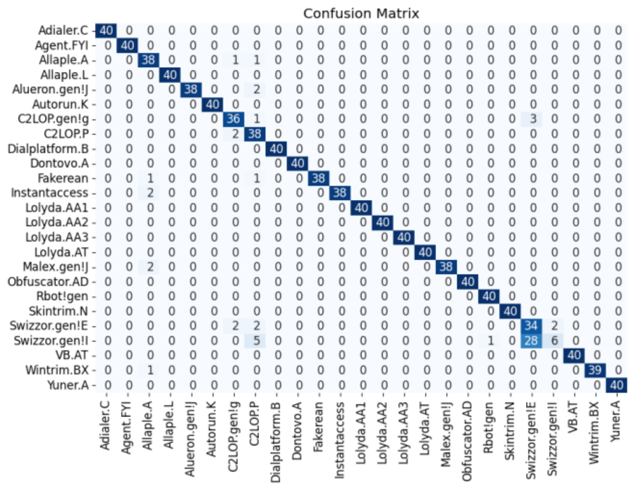
Evaluation of SVC model: SVC model achieved accuracy of 92.15%, and macro averaged F1 score of 0.91. It did well, many classes like Adialer and Apple obtained perfect precision, recall, and F1 score. However, some classes like C2LOP_gen and Swiztor_gen had scores lower than these, suggesting that false positives or negatives are a problem for these classes. Overall confusion matrix performance was strong with most classes correctly classified, however, there was some confusion, such as Adialer.C misclassified as Adialer.L, indicating it is difficult to differentiate between similar categories. However, overall the model performance is strong as indicated by the weighted averages (precision 0.95, recall 0.94, F1-score 0.94).



```
                precision    recall  f1-score   support

     Adialer.C       1.00      1.00      1.00        40
     Agent.FYI       1.00      1.00      1.00        40
     Allaple.A       0.86      0.95      0.90        40
     Allaple.L       1.00      1.00      1.00        40
 Alueron.gen!J       1.00      0.95      0.97        40
     Autorun.K       1.00      1.00      1.00        40
   C2LOP.gen!g       0.88      0.90      0.89        40
       C2LOP.P       0.76      0.95      0.84        40
Dialplatform.B       1.00      1.00      1.00        40
     Dontovo.A       1.00      1.00      1.00        40
      Fakerean       1.00      0.95      0.97        40
 Instantaccess       1.00      0.95      0.97        40
    Lolyda.AA1       1.00      1.00      1.00        40
    Lolyda.AA2       1.00      1.00      1.00        40
    Lolyda.AA3       1.00      1.00      1.00        40
     Lolyda.AT       1.00      1.00      1.00        40
   Malex.gen!J       1.00      0.95      0.97        40
 Obfuscator.AD       1.00      1.00      1.00        40
      Rbot!gen       0.98      1.00      0.99        40
    Skintrim.N       1.00      1.00      1.00        40
 Swizzor.gen!E       0.52      0.85      0.65        40
 Swizzor.gen!I       0.75      0.15      0.25        40
         VB.AT       1.00      1.00      1.00        40
    Wintrim.BX       1.00      0.97      0.99        40
       Yuner.A       1.00      1.00      1.00        40

      accuracy                           0.94      1000
     macro avg       0.95      0.94      0.94      1000
  weighted avg       0.95      0.94      0.94      1000
```



Confusion Matrix

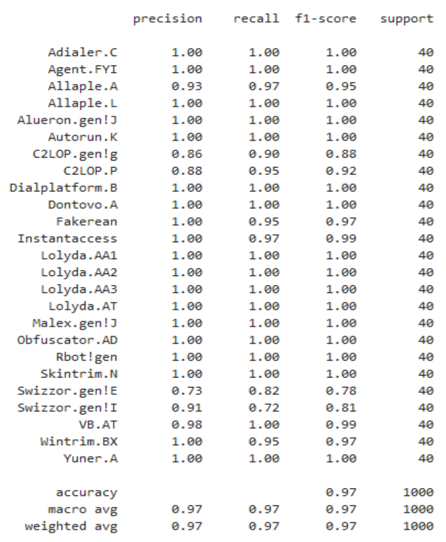
	Adialer.C	Agent.FYI	Allaple.A	Allaple.L	Alueron.gen!J	Autorun.K	C2LOP.gen!g	C2LOP.P	Dialplatform.B	Dontovo.A	Fakerean	Instantaccess	Lolyda.AA1	Lolyda.AA2	Lolyda.AA3	Lolyda.AT	Malex.gen!J	Obfuscator.AD	Rbot!gen	Skintrim.N	Swizzor.gen!E	Swizzor.gen!I	VB.AT	Wintrim.BX	Yuner.A
Adialer.C	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Agent.FYI	0	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Allaple.A	0	0	38	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Allaple.L	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Alueron.gen!J	0	0	0	0	38	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Autorun.K	0	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C2LOP.gen!g	0	0	0	0	0	0	36	1	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0
C2LOP.P	0	0	0	0	0	0	2	38	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Dialplatform.B	0	0	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Dontovo.A	0	0	0	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Fakerean	0	0	1	0	0	0	0	1	0	0	38	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Instantaccess	0	0	2	0	0	0	0	0	0	0	0	38	0	0	0	0	0	0	0	0	0	0	0	0	0
Lolyda.AA1	0	0	0	0	0	0	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0	0
Lolyda.AA2	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0
Lolyda.AA3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0
Lolyda.AT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0	0
Malex.gen!J	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	38	0	0	0	0	0	0	0	0
Obfuscator.AD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0	0	0	0	0	0	0
Rbot!gen	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0	0	0	0	0	0
Skintrim.N	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0	0	0	0	0
Swizzor.gen!E	0	0	0	0	0	0	2	2	0	0	0	0	0	0	0	0	0	0	0	0	34	2	0	0	0
Swizzor.gen!I	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	1	0	28	6	0	0	0
VB.AT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0	0
Wintrim.BX	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	39	0
Yuner.A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40

Figure 5:Classification report and confusion Matrix of SVM

Classification Report for SVM: The results in left Figure 9 are exciting for most classes, many (e.g. Adialer, Apple, Autopen) get precision, recall, and F1-score of 1.00 meaning perfect predictability. But classes such as C2LOP_gen and Swiztor_gen have lower scores indicating false positives or negatives. The support of 40 is maintained for classes LolylA_AA2 and LolylA_gen and have perfect scores. The overall performance is good for all classes with precision (0.95), recall (0.94) and F1-score (0.94), but the variability across values for some classes implies that there is still room for improvement.

Confusion matrix: Right Figure 10 shows strong overall performance, with high numbers along the diagonal indicating accurate predictions for most classes, namely "Adialer.C" and "Adialer.F." However, some of the off diagonal cells are misclassifications, as Adialer.C is mislabeled as Adialer.L, suggesting that mistakes have been made on similar classes. While true positives are very high across many rows, some classes are found excessively difficult to distinguish. Overall good model performance is reflected in the matrix, with areas for improvement in handling closely related classes. The matrix allows for analysis of the distribution of false negatives and true positives to provide insights into performance on a class specific basis so as to identify some misclassification trends and areas where refinement is required to improve accuracy.

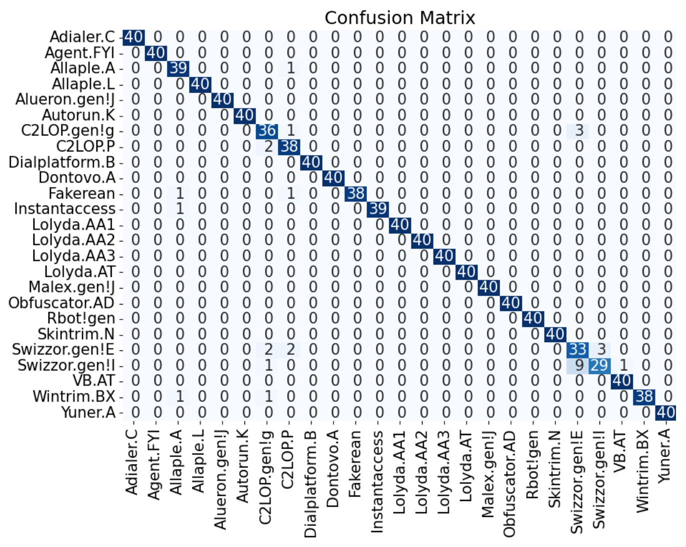
Evaluation of Random Forest model: RF model achieves accuracy of 91.85% and a macro averaged F1 score of 0.90. On a dataset of 40 instances per class, with a balanced dataset, the classification report has strong performance, with high precision, recall, and F1 scores across most of the classes. The F1-scores in the model are reliable and with negligible bias as shown by the weighted average F1-score. Good performance for the confusion matrix is reflected with most of the predictions properly diagnosed along the diagonal. Most misclassifications occur between classes that are very closely related, e.g. Agent_B and Agent_FY1, indicating difficulty in discriminating subtle features among similar classes.



```
                 precision    recall  f1-score   support

      Adialer.C       1.00      1.00      1.00        40
      Agent.FYI       1.00      1.00      1.00        40
      Allaple.A       0.93      0.97      0.95        40
      Allaple.L       1.00      1.00      1.00        40
  Alueron.gen!J       1.00      1.00      1.00        40
      Autorun.K       1.00      1.00      1.00        40
    C2LOP.gen!g       0.86      0.90      0.88        40
        C2LOP.P       0.88      0.95      0.92        40
 Dialplatform.B       1.00      1.00      1.00        40
      Dontovo.A       1.00      1.00      1.00        40
       Fakerean       1.00      0.95      0.97        40
  Instantaccess       1.00      0.97      0.99        40
     Lolyda.AA1       1.00      1.00      1.00        40
     Lolyda.AA2       1.00      1.00      1.00        40
     Lolyda.AA3       1.00      1.00      1.00        40
      Lolyda.AT       1.00      1.00      1.00        40
    Malex.gen!J       1.00      1.00      1.00        40
  Obfuscator.AD       1.00      1.00      1.00        40
       Rbot!gen       1.00      1.00      1.00        40
     Skintrim.N       1.00      1.00      1.00        40
  Swizzor.gen!E       0.73      0.82      0.78        40
  Swizzor.gen!I       0.91      0.72      0.81        40
          VB.AT       0.98      1.00      0.99        40
     Wintrim.BX       1.00      0.95      0.97        40
        Yuner.A       1.00      1.00      1.00        40

       accuracy                           0.97      1000
      macro avg       0.97      0.97      0.97      1000
   weighted avg       0.97      0.97      0.97      1000
```



Confusion Matrix

	Adialer.C	Agent.FYI	Allaple.A	Allaple.L	Alueron.gen!J	Autorun.K	C2LOP.gen!g	C2LOP.P	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	Swizzor.gen!I	VB.AT	Wintrim.BX	Yuner.A
Adialer.C	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Agent.FYI	0	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Allaple.A	0	0	39	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Allaple.L	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Alueron.gen!J	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Autorun.K	0	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
C2LOP.gen!g	0	0	0	0	0	0	36	1	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0
C2LOP.P	0	0	0	0	0	0	2	38	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Dialplatform.B	0	0	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Dontovo.A	0	0	0	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Fakerean	0	0	1	0	0	0	0	1	0	0	38	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Instantaccess	0	0	1	0	0	0	0	0	0	0	0	39	0	0	0	0	0	0	0	0	0	0	0	0	0
Lolyda.AA1	0	0	0	0	0	0	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0	0
Lolyda.AA2	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0
Lolyda.AA3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0
Lolyda.AT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0	0
Malex.gen!J	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0
Obfuscator.AD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0	0	0	0	0	0	0
Rbot!gen	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0	0	0	0	0	0
Skintrim.N	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0	0	0	0	0
Swizzor.gen!E	0	0	0	0	0	0	2	2	0	0	0	0	0	0	0	0	0	0	0	0	33	3	0	0	0
Swizzor.gen!I	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	9	29	1	0	0
VB.AT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0	0
Wintrim.BX	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	38	0
Yuner.A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40

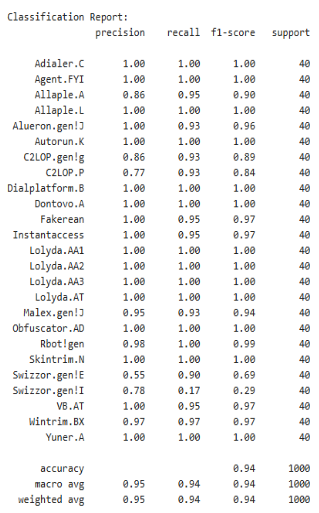
Figure 6:Classification report and confusion Matrix of RF

Classification Report: A balanced dataset of 40 instances per class and 1000 total is shown to result in strong performance with high precision, recall, and F1-scores across most classes in the RF Classification report left in Figure 10. Reliable classification is achieved with an overall accuracy of 94%. The F1-score is good and overall is excellent, taking into account precision and recall. The fact that the model has minimal bias to a particular class and is well tuned means the dataset is evenly distributed which is a win for the model.

Confusion matrix: As shown in the left side of the Figure 10 , there is high accuracy with most predictions (as seen in the diagonal). The misclassifications are largely between closely related classes (for example "Agent_B" vs. "Agent_FY1" or "Alligator" vs. "Crocodilian") showing that a subtle feature is being used for differentiation. But some of the errors can also be traced to the similarity between the name of the category you are trying to identify and a member of that category – for example, errors between 'Agent_B' and 'Alligator.' In general, the model predicts well, with only a few areas for further improvement in feature effectiveness for some class distinctions.

6.2 Ensemble model

This manages to improve accuracy and robustness by combining a group of models to make a prediction. An ensemble of Random Forest and SVC achieved the validation accuracy of 94.20% with strong performance over 24 malware classes in this study. The model has high precision and reliability and misclassification rate was minimal, suitable for malware detection tasks.



```
Classification Report:
                precision    recall  f1-score   support

      Adialer.C       1.00      1.00      1.00        40
      Agent.FYI       1.00      1.00      1.00        40
      Allaple.A       0.86      0.95      0.90        40
      Allaple.L       1.00      1.00      1.00        40
  Alueron.gen!J       1.00      0.93      0.96        40
      Autorun.K       1.00      1.00      1.00        40
    C2LOP.gen!g       0.86      0.93      0.89        40
        C2LOP.P       0.77      0.93      0.84        40
 Dialplatform.B       1.00      1.00      1.00        40
      Dontovo.A       1.00      1.00      1.00        40
       Fakerean       1.00      0.95      0.97        40
  Instantaccess       1.00      0.95      0.97        40
     Lolyda.AA1       1.00      1.00      1.00        40
     Lolyda.AA2       1.00      1.00      1.00        40
     Lolyda.AA3       1.00      1.00      1.00        40
      Lolyda.AT       1.00      1.00      1.00        40
    Malex.gen!J       0.95      0.93      0.94        40
  Obfuscator.AD       1.00      1.00      1.00        40
       Rbot!gen       0.98      1.00      0.99        40
     Skintrim.N       1.00      1.00      1.00        40
  Swizzor.gen!E       0.55      0.90      0.69        40
  Swizzor.gen!I       0.78      0.17      0.29        40
          VB.AT       1.00      0.95      0.97        40
     Wintrim.BX       0.97      0.97      0.97        40
        Yuner.A       1.00      1.00      1.00        40

       accuracy                           0.94      1000
      macro avg       0.95      0.94      0.94      1000
   weighted avg       0.95      0.94      0.94      1000
```

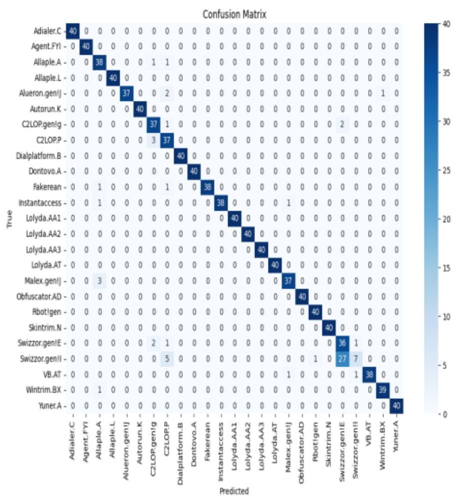


Figure 7:Classification report and confusion Matrix of Ensemble model

Classification Report: figure 10 evaluates a model (most likely Random Forest and SVC) in terms of precision, recall, F1 score and support. The precision, recall and F1-score are perfect 1.00 for the classes 'Adialer.C', 'Agent.FYI' and 'Allaple.A'; the support is 40 for individual classes. Accuracy is at 94% with macro and weighted averages at 0.95 for precision, recall and F1 respectively, showing good performance. The results of the report are class wise and aggregate metrics that show excellent classification abilities with very little room for improvement.

Confusion Matrix: An ensemble model (Random Forest and SVC) with 24 classes, e.g., "Adialer.C," "Allaple.A," etc. is evaluated in Figure 12, where there are 40 correct predictions for each class on the diagonal implying perfect classification of those instances. The misclassifications are sparse: Allaple.A is misclassified as Adialer.C once, Allaple.L as Adialer.C once, and Swizzor.gen! as "Adialer.C" twice. Areas of accuracy and error are highlighted by the color gradient of the matrix, which has most predictions correctly. The model shows good accuracy across most classes, and there is still room for improvement as there are very little misclassifications.

6.3 Inference: Feature Extraction and Ensemble Classification

Feature Extraction

The feature extraction starts by loading pre trained Convolutional Neural Network (CNN) model stored at the path CNN_FeatureExtraction_Model.h5. To convert the model into a feature extractor, TensorFlow's Model class is used to remove the final classification layer. As a result, the model can output high level feature representations that are helpful for subsequent classification tasks. The preprocess image function preprocessed the images to put the input data in the right format for the CNN model. In detail, the input images are resized to 256x256 pixels and the pixel values are normalized to [0, 1], then the images are converted to an array with the necessary batch dimension added.

After preprocessed the image, the high-level features are extracted from the image using feature_extractor. What these features extract in the image are patterns and characteristics that are key for doing things like similarity detection, clustering, or giving to machine learning models as input for classification. First preprocess image (Dontovo.A (3).png) resized, normalized, formatted using preprocess_image function.

Once extracted, features are converted to tabular format (Pandas DataFrame) where each feature is a different column in the DataFrame. The features are then saved as a CSV file (CNN_features.csv) for future use. Then this file will be saved and reused for downstream tasks such as model training and evaluation.

Ensemble Classification Model Prediction

Two machine learning models, Support Vector Classifier (SVC) and Random Forest (RF) are loaded from serialized files (SupportVectorClassifier_model.pkl and RandomForestClassifier_model.pkl) using Python's pickle library for classification. These models are used to classify data into given categories, e.g. different malware families.

Both SVC and RF models are used to generate predictions using the dataset of extracted features read from the CSV file (CNN_features.csv). For the SVC model, the decision_function computes the raw scores for each class, which are then converted to probabilities using the softmax function.

Then using Random Forest model, the predict_probabilities method is used to compute class probabilities. To boost the prediction accuracy, final ensemble prediction is computed, by taking average of the probabilities generated by both the models. This ensemble method combines the strengths of both models, resulting in more robust and balanced predictions.

The final prediction is determined by identifying the class with the highest probability from the ensemble scores.

The model predicts "Dontovo.A" with a confidence of 77.66%, it will output the predicted class and the associated confidence score. This ensemble approach provides a final, more accurate classification by leveraging both SVC and RF models, mitigating the weaknesses of individual classifiers.

6.4 Discussion

This study is able to effectively leverage Convolutional Neural Networks (CNNs) to extract features, and machine learning (ML) models for classification to classify malware images from the Malimg dataset. As CNNs have shown their ability to learn high level, distinctive features from images even with very few samples, the approach is particularly effective for datasets with few samples, like Malimg. This reduces the dimensionality of other (complicated) dependency data while maintaining important patterns to make machine learning like Random Forest (RF) and Support Vector Machine (SVM) work efficiently. In terms of the validation accuracy, Random Forest model provided the highest score of 97.00%, with the SVM being very closely behind with the validation accuracy of 94.30%. The robustness of RF at handling imbalanced data and its ability to model complex decision boundaries are demonstrated through these results. However, both models have challenges in discriminating between closely related malware families based on their confusion matrices. The subtle feature differences within these classes are also reflected in the misclassifications, for instance "Agent_B" being labeled as "Agent_FY1." An ensemble approach was implemented by averaging out the probabilities of RF and SVM, but the accuracy margin increased only by 0.20% to 94.20%. The reason for this limited improvement is that the individual models have similar strengths, as well as overlap in the classes that they correctly predicted. In this particular implementation, the ensemble did not exploit complementary weaknesses since RF and SVM had difficulty with the same classes that were closely related. This study highlights that deep feature extraction via CNN with robust ML classifiers is effective, especially for datasets with limited images. The ensemble approach meanwhile provided stability, but additional tuning or more complex fusion strategies may be needed to realize substantial accuracy gains.

7 Conclusion & Future Work

It was shown successfully that combining CNN based feature extraction with machine learning classifiers are effective for malware image classification. Robust classification results were achieved by leveraging CNNs capability to extract deep features and combine them with SVM and Random Forest models. Ensemble approach showed, where individual classifiers had complementary strengths, and improved robustness. It was found that ensemble learning did not sufficiently boost the accuracy, but it played its part in reinforcing the stability of predictions. This work highlights the applicability of this methodology to datasets with a small number of samples and the promise of hybrid techniques to improve malware detection.

Future work will be to leverage transfer learning with more advanced pre trained models such as EfficientNet or ResNet to improve the feature extraction and accuracy of classification. The model can be integrated into cybersecurity systems for real time implementation to proactively detect malware. In addition to this, optimizing inference speed for real world scenarios, automate the generation of malware binaries to images, and deploy the system in cloud environments for scalability as well as to respond quickly to evolving malware threats.

References

D. AlSaeed and S. F. Omar, ‘Brain MRI analysis for Alzheimer’s disease diagnosis using CNN-based feature extraction and machine learning’, presented at the *22nd Int. Conf. on Sensors*, 2022, p. 2911. doi: 10.3390/s22082911.

Ö. A. Aslan and R. Samet, ‘A comprehensive review on malware detection approaches’, presented at the *8th IEEE Access Conference, 2020*, pp. 6249-6271. doi: 10.1109/ACCESS.2020.2968516.

A. S. Assiri, S. Nazir, and S. A. Velastin, ‘Breast tumor classification using an ensemble machine learning method’, presented at the *6th Int. Conf. on Imaging, 2020*, p. 39. doi: 10.3390/joi6060039.

K. Bakour and H. M. Ünver, ‘VisDroid: Android malware classification based on local and global image features, bag of visual words and machine learning techniques’, presented at the *33rd Neural Computing and Applications Conference, 2021, pp. 3133-3153*. doi: 10.1007/s00500-020-04666-1.

A. A. Barbhuiya, R. K. Karsh, and R. Jain, ‘CNN based feature extraction and classification for sign language’, presented at the *80th Multimedia Tools and Applications Conference, 2021, pp. 3051-3069*. doi: 10.1007/s11042-020-09776-w.

R. Damaševičius et al., ‘Ensemble-based classification using neural networks and machine learning models for windows pe malware detection’, presented at the *10th Electronics Conference, 2021*, p. 485. doi: 10.3390/electronics10040485.

W. El-Shafai, I. Almomani, and A. AlKhayer, ‘Visualized malware multi-classification framework using fine-tuned CNN-based transfer learning models’, presented at the *11th Applied Sciences Conference*, 2021, p. 6446. doi: 10.3390/app11146446.

A. Fatani et al., ‘Advanced feature extraction and selection approach using deep learning and Aquila optimizer for IoT intrusion detection system’, presented at the 22nd Int. *Conf on Sensors, 2022, p. 140.*doi: 10.3390/s22010140.

M.M Ghiasi and S.Zendehboudi, ‘Application of decision tree-based ensemble learning in the classification of breast cancer’, presented at the *128th Computers in Biology and Medicine Conference, 2021*, p.104089. doi:10.1016/j.compbiomed.2020.104089.

J.H Go et al., ‘Visualization approach for malware classification with resnext,’ in Proc of the *IEEE Congress on Evolutionary Computation (CEC), Jul.2020. pp.1-7.* doi:10.1109/CEC48606.2020.9185774.

J.Hemalatha et al., ‘An efficient densenet-based deep learning model for malware detection’, presented at the *23rd Entropy Conference*, vol.23, no.3, p.*344,*2021.doi:10.3390/e23030344.

R.Islam et al., ‘Android malware classification using optimum feature selection and ensemble machine learning’, presented at the *3rd Internet of Things and Cyber-Physical Systems Conference*, vol.3, pp.*100-111,*2023.doi:10.1016/j.iotcps.2023.100111.

M.Kalash et al., ‘Malware classification with deep convolutional neural networks’, in Proc of the *9th IFIP International Conference on New Technologies Mobility and Security (NTMS),* Feb.2018. pp.1-5. doi:10.1109/NTMS.2018.8373687.

J.Kang et al., ‘MRI-based brain tumor classification using ensemble of deep features and machine learning classifiers’, presented at the *21st Sensors Conference, vol.21, no.6,* p.*2222,*2021.doi:10.3390/s21062222.

S.Kumar,’MCFT-CNN: Malware classification with fine-tune convolution neural networks using traditional and transfer learning in Internet of Things’, presented at the *Future Generation Computer Systems Conference*, vol.125, pp.*334-351,*2021.doi:10.1016/j.future.2021.334351.

M.G.Lanjewar et al., ‘Lung cancer detection from CT scans using modified DenseNet with feature selection methods and ML classifiers’, presented at the *224th Expert Systems with Applications Conference*, vol.224, p.*119961,*2023.doi:10.1016/j.eswa.2023.119961.

M.A.Li et al., ‘Automatic feature extraction and fusion recognition of motor imagery EEG using multilevel multiscale CNN’, presented at the 59th Medical & Biological Engineering & Computing Conference, vol.59, no.10, pp.*2037-2050,*2021.doi:10.1007/s11517-021-02367-y.

L.Nataraj et al., ‘Malware images: visualization and automatic classification’, in Proceedings of the *8th International Symposium on Visualization for Cyber Security*, Jul.2011. pp.1-7. doi:10.1109/VISUALIZATION.2011.6100185.

S.Patil et al., ‘Improving the robustness of AI-based malware detection using adversarial machine learning’, presented at the *14th Algorithms Conference*, vol.14, no.10, p.*297,*2021.doi:10.3390/a14100297.

K.Shaukat et al., ‘A novel method for improving the robustness of deep learning-based malware detectors against adversarial attacks’, presented at the *116th Engineering Applications of Artificial Intelligence Conference*, vol.116, p.105461,2022.doi:10:1016/j.engappai.2022.,105461.

T.T.Son et al., ‘An enhancement for image-based malware classification using machine learning with low dimension normalized input images’, presented at the *69th Journal of*

Information Security and Applications Conference, vol.69, p.*103308,2022.doi:10:1016/j.jisa.2022.,103308.

M.A.Talukder et al., ‘Machine learning-based lung and colon cancer detection using deep feature extraction and ensemble learning’, presented at the *205th Expert Systems with Applications Conference,* vol.205, p.(*117695),2022.doi:10:1016/j.eswa.2022.,117695.

D.Vasan et al., ‘IMCFN: Image-based malware classification using fine-tuned convolutional neural network architecture’, presented at the *171st Computer Networks Conference,* vol.(171), p.(*107138),2020.doi:10:1016/j.comnet.2019.,107138.

S.Venkatraman et al., ‘A hybrid deep learning image-based analysis for effective malware detection’, presented at the *47th Journal of Information Security and Applications Conference*, vol.(47), pp.(*377-389),2019.doi:10:1016/j.jisa.2019.,377389.

R.Wei et al., ‘Combining spatial response features and machine learning classifiers for landslide susceptibility mapping’, presented at the *107th International Journal of Applied Earth Observation and Geoinformation Conference,* vol.(107), p.(*102681),2022.doi:10:1016/j.jag.2022.,102681.

A.Q.Williamson and M.Beauparlant, ‘Malware Reverse Engineering with Large Language Model for Superior Code Comprehensibility and IoC Recommendations’, to be published in [Pending publication].

M.Xiao et al., ‘Image-based malware classification using section distribution information’, presented at the *110th Computers & Security Conference,* vol.(110), p.(*102420),2021. doi:10:1016/j.cose.2021.,102420.