CS300

Innovative Security Measures: Classifying Malware through Visual Features

Abhishek Kumar (2101012)
Department of Computer Science and Engineering
Indian Institute of Information Technology Guwahati
Advisor: **Dr. Rakesh Matam**

April 2024

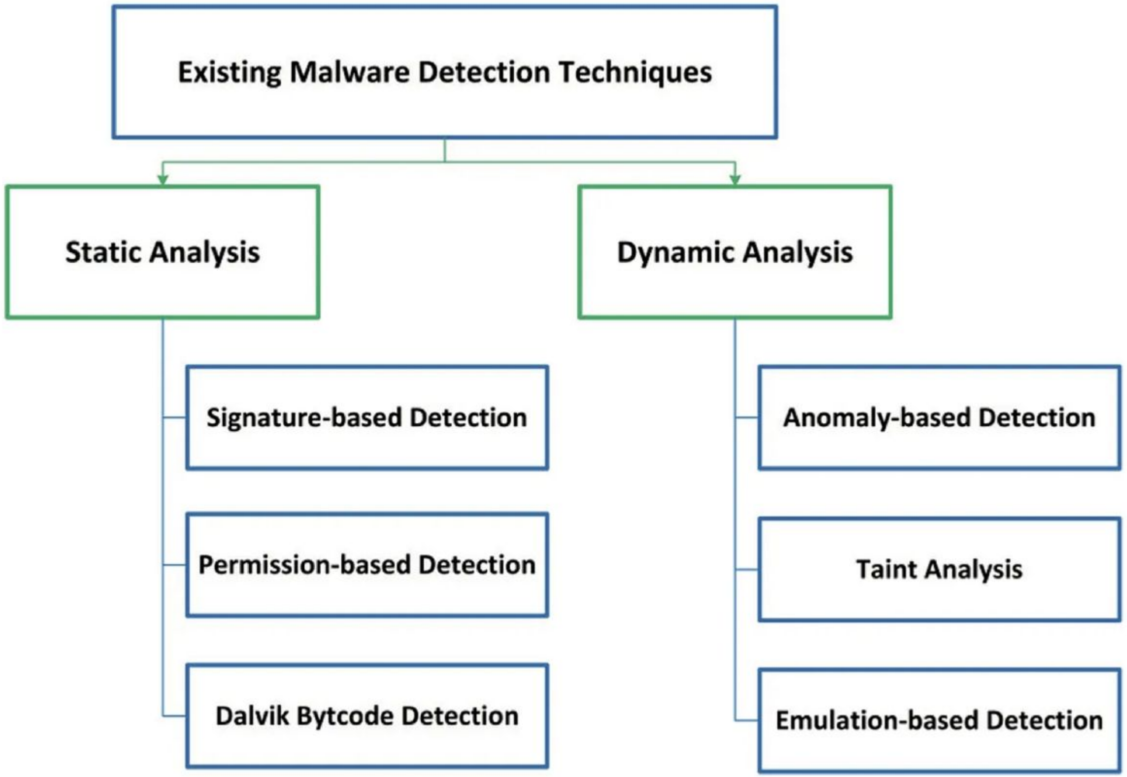
What is Malware?

- A program or piece of code that is intentionally designed to harm a computer system or its users.
- Malware typically enters a system covertly, often without the user's knowledge or consent. This can happen through various means, such as: Downloading infected files from the internet, Visiting compromised websites.
- Malware's primary goal is to compromise the CIA triad of a system:
    - Confidentiality: Malware can steal sensitive data, such as passwords, financial information, or personal data
    - Integrity: Malware can modify or corrupt data, making it unreliable or unusable.
    - Availability: Malware can disrupt system operations, making it difficult or impossible to access data or use applications.

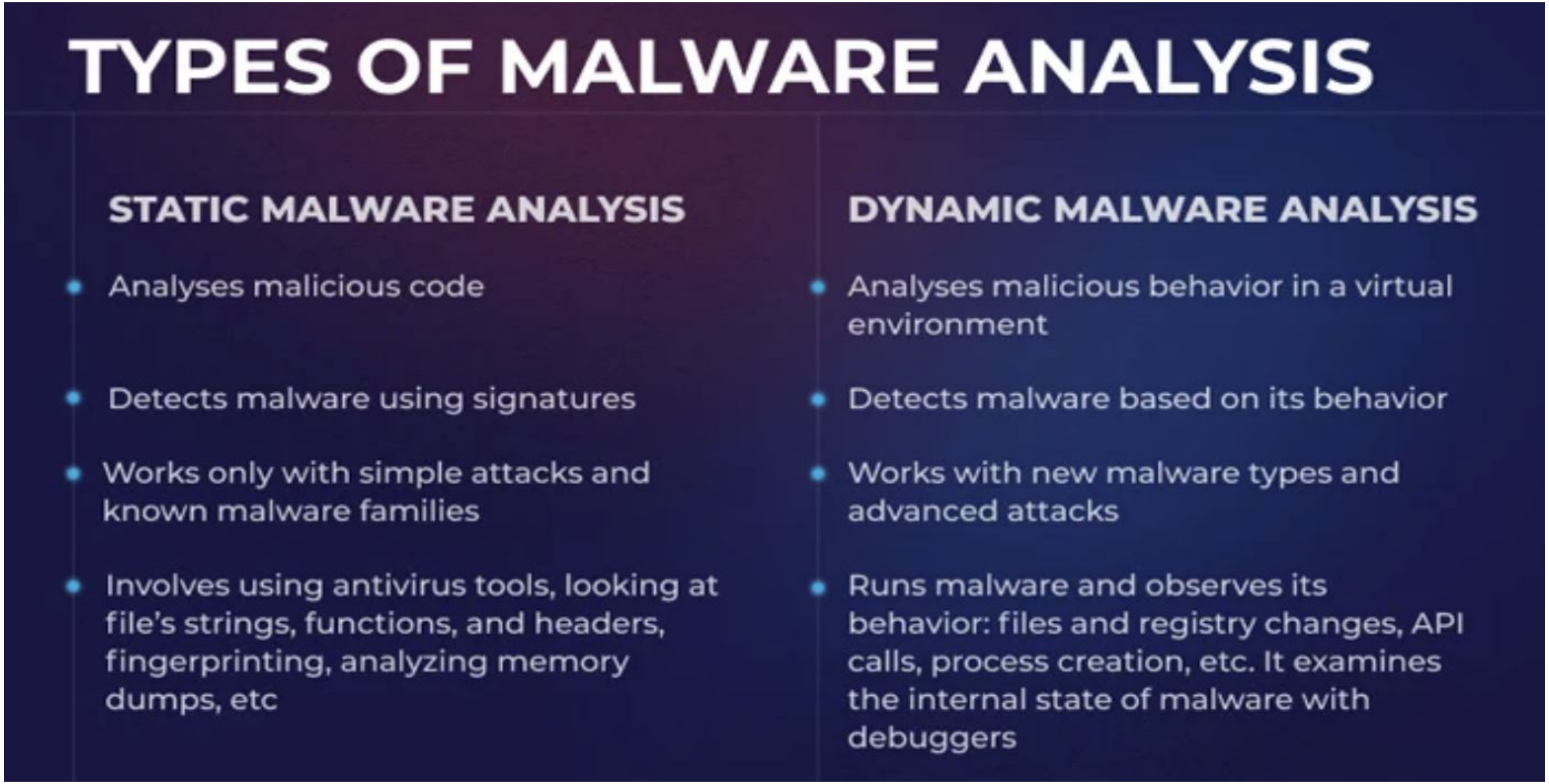
Types of Malware:

Malware comes in various forms, each with its own characteristics and methods of attack. Some common types include:

- Viruses: Self-replicating programs that infect other files or programs on a system.
- Worms: Self-spreading programs that can propagate across networks without user interaction.
- Trojans: Malware disguised as legitimate software, tricking users into installing them.
- Rootkits: Malware that hides its presence and activities from the user and security software.
- Ransomware: Malware that encrypts the victim's data and demands payment for decryption.
- Spyware: Malware that collects information about the user's activities without their knowledge or consent.
- Adware: Malware that displays unwanted advertisements.


Existing Malware Detection Techniques
Static Analysis
Dynamic Analysis
Signature-based Detection
Permission-based Detection
Dalvik Bytcode Detection
Anomaly-based Detection
Taint Analysis
Emulation-based Detection

Source : Android Application Security Scanning Process - Iman Almomani and Mamdouh Alenezi



TYPES OF MALWARE ANALYSIS

STATIC MALWARE ANALYSIS	DYNAMIC MALWARE ANALYSIS
Analyses malicious code	Analyses malicious behavior in a virtual environment
Detects malware using signatures	Detects malware based on its behavior
Works only with simple attacks and known malware families	Works with new malware types and advanced attacks
Involves using antivirus tools, looking at file's strings, functions, and headers, fingerprinting, analyzing memory dumps, etc	Runs malware and observes its behavior: files and registry changes, API calls, process creation, etc. It examines the internal state of malware with debuggers

Malware detection and classification

- Malware detection is the process of investigating the content of the program and deciding whether the analyzed program is malware or benign. The malware detection process includes 3 stages: Malware analysis, feature extraction, and classification.
- After the file is identified as malware, specifying the category or family of malware known as malware classification.
- Malware analysis starts with basic static analysis and finishes with advanced dynamic analysis.

Why Malware Classification?

Enhanced Detection:

- Classifying malware helps in identifying specific types and families of malware, allowing for more precise detection mechanisms that can catch malicious software before it causes damage.

Improved Response:

- By understanding the category or type of malware, cybersecurity professionals can implement targeted countermeasures. Different types of malware may require different response strategies, and classification facilitates appropriate and effective responses.

Enhances Machine Learning Models:

- For cybersecurity systems that use machine learning, malware classification provides labeled data that can be used to train more accurate and robust predictive models, enhancing automated defenses

CONT’D

Trend Analysis:

- Over time, malware classification data can reveal trends and shifts in the methods and targets of attackers, helping to anticipate future security needs and adapt defenses accordingly.

Limitations of Traditional Approaches

Difficulty in Detecting Zero-Day Threats and Polymorphic Malware:

- **Zero-Day Threats:** This method relies on matching known malware signatures (unique patterns or code sequences) to identify threats. It is ineffective against new or unknown malware (zero-day threats) that haven't been added to signature databases.

- **Polymorphic Malware:** Some malware can change its code or appearance to evade detection. Polymorphic malware generates new variants with different signatures, making it difficult for signature-based systems to keep up.

CONT’D

Limited Adaptability and False Negative:

- **Static Analysis:** Traditional methods often rely on static analysis, examining code without execution. This can miss malware that uses obfuscation or dynamic code generation to hide its malicious behavior.

- **Signature-Based Detection:** If signatures are not updated regularly, malware with new variants can evade detection, leading to false negatives.

Visual-Based Malware Classification

Visual-based malware classification is a relatively new approach that involves converting malware samples into images. This is done by treating the binary code of the malware as a sequence of bytes and then mapping these bytes to pixel values in an image.

- Byte-to-decimal conversion
- Color mapping

Once malware samples are converted into images, image analysis techniques can be used to extract meaningful features. These features can include:

- Textures
- Shapes
- Edges and contours
- Frequency-domain features

Advantage Of Visual Based Classification

- **Robustness to Variations:** Image analysis techniques can identify similarities and differences between malware samples even if they have different signatures or code structures, making it effective against malware variants.
- **Less time consuming**: The schemes are expected to be efficient, the time for extracting features and classifying malware samples should be limited to some degree that users can withstand.
- **Easy to use**: Even though security engineers may not have enough background knowledge about malware samples, the schemes should be easy to use and comprehend.
- **Automated Feature Learning:** CNNs can automatically learn relevant features from malware images, eliminating the need for manual feature engineering, which can be time-consuming and require domain expertise.

Literature Review

VisMal: A Novel Framework for Efficient Malware Classification

ZHONG ET AL.: MALWARE-ON-THE-BRAIN: ILLUMINATING MALWARE BYTE CODES WITH IMAGES FOR MALWARE CLASSIFICATION

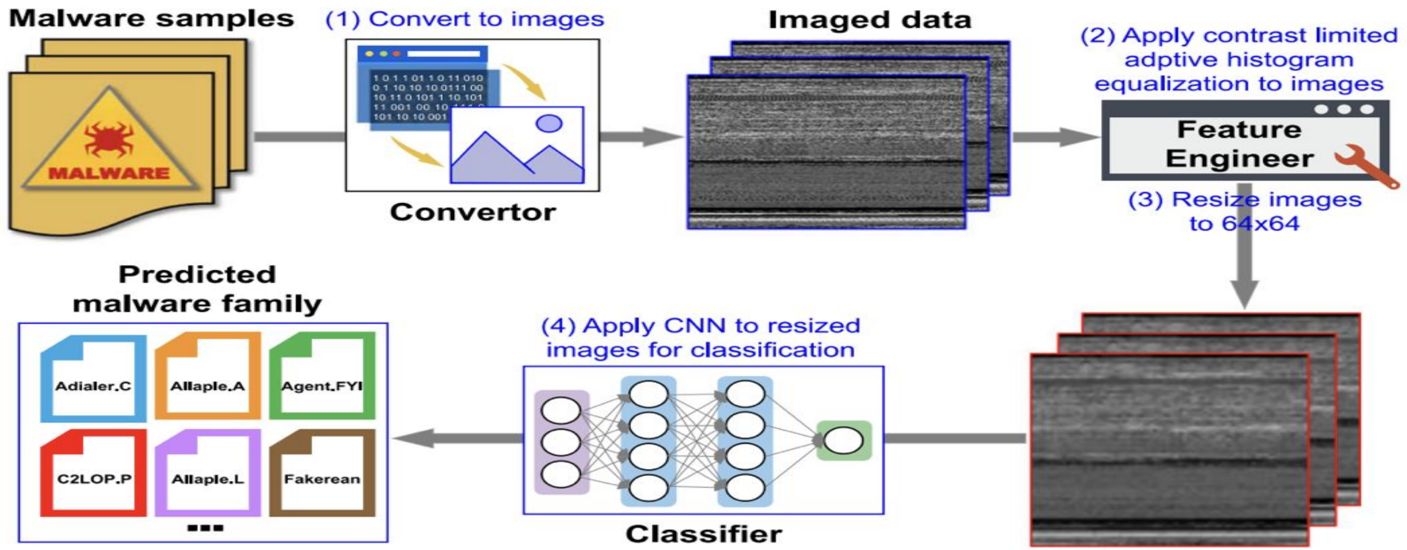


Fig. 1. Overview of the VisMal framework.

Components of the VisMal Framework

- **Converter:**
  - Converts malware binaries to grayscale images using a sequential byte-to-decimal conversion.
  - Reshapes the image based on recommended fixed widths with variable heights.
- **Feature Engineer:**
  - Identifies similar instruction sequences through contrast, improving Classifier accuracy.
  - Utilizes a contrast-limited adaptive histogram equalization algorithm for pixel intensity adjustment.
  - Transformation phase enhances pixel interrelation, promoting better image contrast.

CONT’D

- ❖ **Classifier:**
  - ➢ Employs a CNN algorithm with 12 layers for classification.
  - ➢ Regularization is applied to convolutional layers to prevent overfitting.
  - ➢ Dropout layer enhances generalization.
  - ➢ Softmax activation in the final layer provides probabilities for malware family classification.

Reference	Feature Extraction	Classification	Accuracy	Limitation
F. Zhong[1], 2022	Convert malware binaries to grayscale images	Custom CNN with softmax	average accuracy 96.0%	An adversary who knows the technique can take countermeasures to beat the system
Nataraj et al. [2], 2011	Grayscale image visualization of malware binary, GIST feature	KNN	98%	An adversary who knows the technique can take countermeasures to beat the system
Mahmoud et al. [3], 2018	Convert malware binaries to grayscale images, GIST	CNN, SVM	Malimg- 98.52% , Microsoft- 99.97%	Only .bytes files are used for training the model
Ya-Shu et al. [4], 2019	Grayscale images, Multi-layer dense SIFT, LBP, bag-of-visual-words	KNN, RF	MalingA- 0.990, MalingB- 0.974, CNCERT- 0.938	Higher computational cost, time consuming

Source:Malware Classification Approaches Using Machine Learning Techniques: A Review

Our Approach Towards Classification

Our work on classifying the Malimg dataset using cutting-edge neural network architectures - MobileNetV1 and some other CNN models.

By employing advanced techniques in deep learning, we aim to build robust classifiers capable of distinguishing between different types of malware based on their visual features.

We removed the top layer (usually a fully connected layer with softmax activation) of both MobileNetV1 to extract features from the penultimate layer. This allows us to use different classifiers, such as decision trees, random forests, SVMs, Naive Bayes, and KNN, to perform the final classification task.

Objectives:

- To achieve higher accuracy in classifying malware images into malware families using different CNN architectures with various classifiers efficiently.
- To study different approaches for malware classification.
- Increase accuracy of models using backpropagation.

Malimg Dataset: A Collection of Malware Images

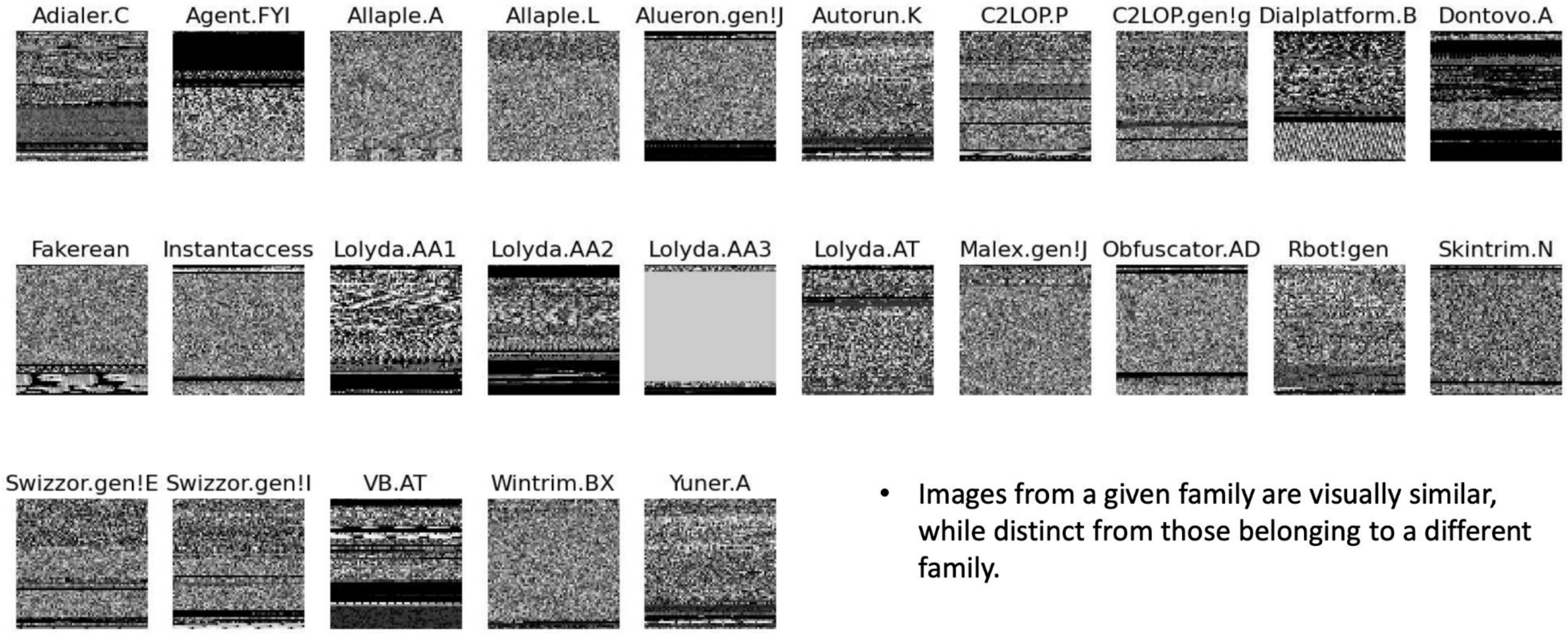
The Malimg dataset is a publicly available collection of malware images that is widely used in research on visual-based malware classification. It was created by converting malware binaries into grayscale images, providing, providing a visual representation of the malware's code structure and patterns.

Key Characteristics:

- **Size:** The dataset contains 9,339 images.
- **Classes:** The images are labeled with 25 different malware families, including common threats like Allaple, Yuner.A, and, and Lolyda.A.

Malimg dataset

- Dataset samples for each class:



- Images from a given family are visually similar, while distinct from those belonging to a different family.

Visualizing Malware as an Image

The similarity of images of malware of particular category when categorized as grayscale images were first seen in Malware Images: Visualization and Automatic Classification.

In this paper, they show how a Trojan Virus would look. The .text part contains the code which gets executed and the .text part towards the end is full black which indicates zero paddings in the end. The .data part contains uninitialized code and .rsrc contains all resources of the module like the icons application may use.

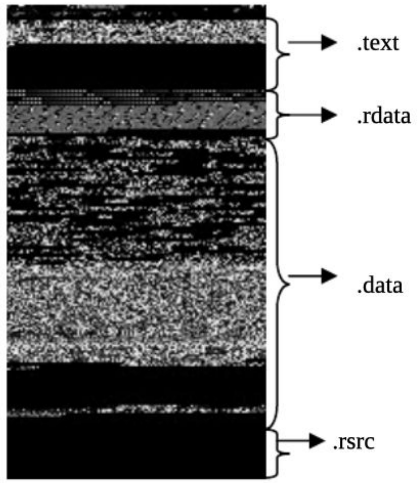


Fig. 2 Various Sections of Trojan: Dontovo.A

source: Malware Images: Visualization and Automatic Classification

Some of the Malware Families

- **Adialer.C(Dialer):** Its primary function is to dial premium-rate phone numbers without your knowledge or consent, resulting in unexpected charges on your phone bill.It can spread through various means, including: Malicious email attachments,Drive-by downloads,Bundled software
- **Allaple.A(Worm):** It is able to scan for computers vulnerable to a number of exploits to spread itself; it can also perform a dictionary attack on network share passwords. The worm performs a Denial of Service (DoS) attack on a number of websites.
- **Alueron.gen!J(Trojan):** It is a complex and multifaceted malware family that poses a significant threat to computer systems. Key characteristics: Trojan,Backdoor,Data Theft,Botnet Creation,keylogging
- **Agent.FYI(Backdoor):** It is a group of malicious programs designed to steal sensitive information from infected computers. It is known for its stealthy nature and ability to evade detection by antivirus software.

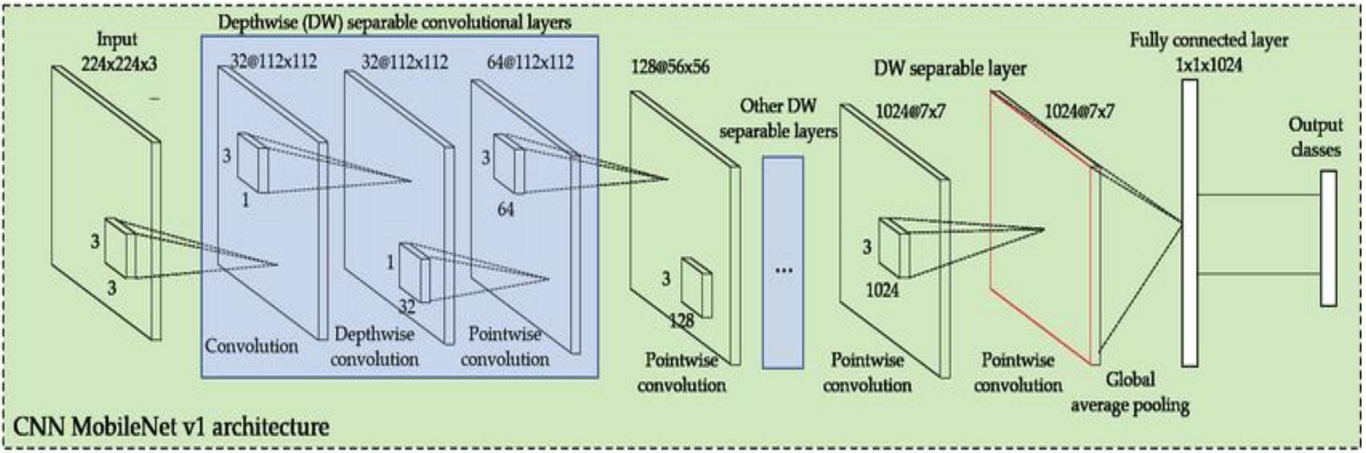
Class ID	Family Name	Description	
		Malware Kind	Sample No.
#1	Adialer.C	Dialer	122
#2	Agent.FYI	Backdoor	116
#3	Allaple.A	Worm	2949
#4	Allaple.L	Worm	1591
#5	Alueron.gen!J	Trojan	198
#6	Autorun.K	Worm	106
#7	C2LOP.P	Trojan	200
#8	C2LOP.gen!g	Trojan	146
#9	Dialplatform.B	Dialer	177
#10	Dontovo.A	Downloader	162
#11	Fakerean	rogue	381
#12	Instantaccess	Dialer	431
#13	Lolyda.AA1	PWS	213

#14	Lolyda.AA2	PWS	184
#15	Lolyda.AA3	PWS	123
#16	Lolyda.AT	PWS	159
#17	Malex.gen!J	Trojan	136
#18	Obfuscator.AD	Downloader	142
#19	Rbot!gen	Backdoor	158
#20	Skintrim.N	Trojan	80
#21	Swizzor.gen!E	Downloader	128
#22	Swizzor.gen!I	Downloader	132
#23	VB.AT	Worm	408
#24	Wintrim.BX	Downloader	97
#25	Yuner.A	Worm	800
	Total	-	9339

Architecture Overview

MobileNet V1: Lightweight and Efficient

- MobileNetV1 employs a lightweight architecture using depthwise separable convolutions, consisting of two key operations: depthwise convolution and pointwise convolution.
- The network consists of 13 blocks, each applying these convolutions, allowing it to process raw input images efficiently.
- The final layers include typical operations like pooling, fully connected layers, and softmax for classification.
- This design strikes a balance between performance and computational efficiency, outperforming earlier models using traditional convolutions while being more cost-effective.
- The extracted features from MobileNet V1 typically have a shape of (batch_size, 1024). This means that each malware image is represented by a 1024-dimensional feature vector.



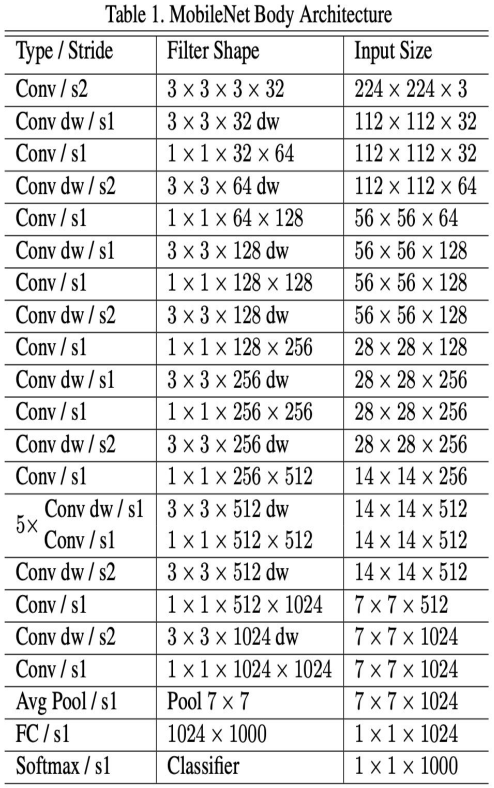


Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
5× Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

Source:https://www.researchgate.net/figure/CNN-MobileNet-v1-architecture_fig2_354985604 [5]

Feature Extraction Process:

1. **Load the pre-trained model:** We use pre-trained versions of MobileNetV1 which have been trained on large image datasets like ImageNet. This allows us to leverage the knowledge learned from these datasets to extract meaningful features from malware images.
2. **Remove the top layer:** The top layer of these models is typically a fully connected layer with softmax activation, which is used for classification. We remove this layer to obtain the feature representations from the penultimate layer.
3. **Pass the malware image through the model:** We feed the malware image as input to the model. The model processes the image through its layers, extracting features at each stage.
4. **Extract features from the penultimate layer:** The output of the penultimate layer is a feature vector that represents the malware image in a lower-dimensional space. This feature vector captures the essential characteristics of the image that are relevant for classification.

Classifier Overview

K-Nearest Neighbors (KNN):

- Utilizes the proximity of data points in the feature space.
- Makes predictions based on the class of the majority of its k-nearest neighbors.

Naive Bayes:

- Probabilistic classifier based on Bayes' theorem, assumes independence among features.
- Naive Bayes calculates the probability of each class given the input features and selects the class with the highest probability as the prediction.

Decision Tree:

- Hierarchical structure of decision nodes for classification.
- Captures complex decision boundaries and feature interactions.

CONT'D

Support Vector Machine (SVM):

- Finds the hyperplane that best separates classes.
- Effective in high-dimensional spaces; capable of handling non-linear decision boundaries.

Random Forest:

- Ensemble method consisting of multiple decision trees.
- Reduces overfitting and enhances generalization.

Approach:

- Each classifier was employed as the last layer on top of the extracted features.
- Comparative analysis of these classifiers provides insights into their performance on malware image classification.

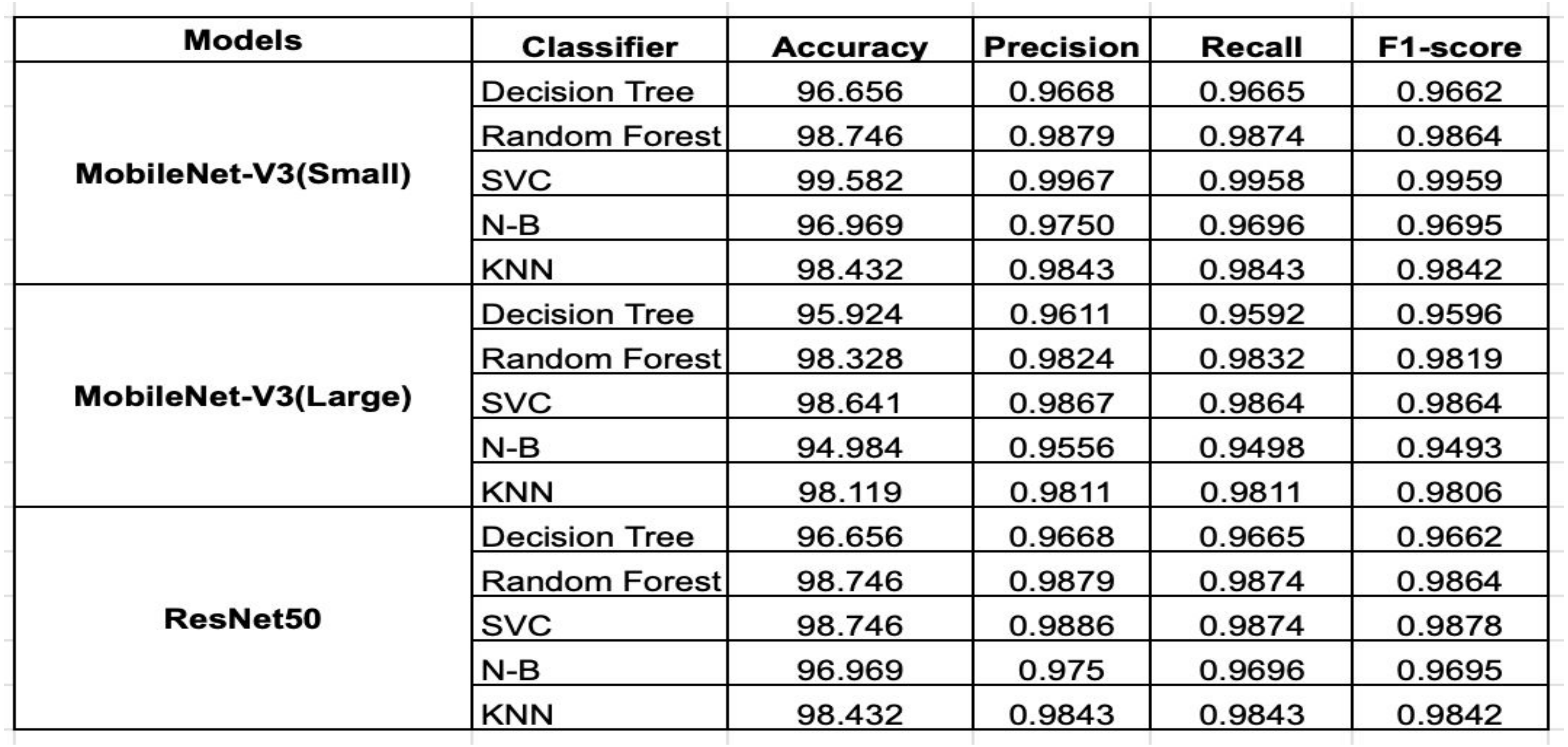
Results on MobilenetV1 using different classifiers

Classification Algorithm	Accuracy	Precision	Recall	F1_Score	TPR	TNR	FPR	FNR	FDR	FOR
Decision Tree	94.252	0.9430	0.9425	0.9426	0.8855	0.9977	0.0022	0.1144	0.1241	0.0022
Random Forest	98.400	0.9846	0.9843	0.9840	0.9628	0.9993	0.0006	0.0371	0.0341	0.0007
Support Vector Classifier	99.373	0.9938	0.9937	0.9937	0.9870	0.9990	0.0002	0.0125	0.0134	0.0002
Naive-Bayes Classifier	94.148	0.9244	0.9414	0.9307	0.8604	0.9976	0.0023	0.1395	0.1320	0.0023
K-Nearest Negihbors	98.533	0.9865	0.9853	0.9854	0.9640	0.9990	0.0005	0.0350	0.0320	0.0006

Analysis on MobilenetV1 using different classifiers

- **SVC and Random Forest** are particularly strong, offering both high accuracy and reliability. **KNN** also performs exceptionally well with high scores across most metrics.
- **Decision Tree and Naive-Bayes**, while effective, display some weaknesses in terms of lower TPR suggesting areas for potential improvement.
- These results underline the capability of MobileNetV1 to serve as a robust feature extractor in malware classification tasks, even without the application of backpropagation to enhance its feature recognition capabilities.

Other CNN Models Result



Models	Classifier	Accuracy	Precision	Recall	F1-score
MobileNet-V3(Small)	Decision Tree	96.656	0.9668	0.9665	0.9662
	Random Forest	98.746	0.9879	0.9874	0.9864
	SVC	99.582	0.9967	0.9958	0.9959
	N-B	96.969	0.9750	0.9696	0.9695
	KNN	98.432	0.9843	0.9843	0.9842
MobileNet-V3(Large)	Decision Tree	95.924	0.9611	0.9592	0.9596
	Random Forest	98.328	0.9824	0.9832	0.9819
	SVC	98.641	0.9867	0.9864	0.9864
	N-B	94.984	0.9556	0.9498	0.9493
	KNN	98.119	0.9811	0.9811	0.9806
ResNet50	Decision Tree	96.656	0.9668	0.9665	0.9662
	Random Forest	98.746	0.9879	0.9874	0.9864
	SVC	98.746	0.9886	0.9874	0.9878
	N-B	96.969	0.975	0.9696	0.9695
	KNN	98.432	0.9843	0.9843	0.9842

Overall Observations

- **SVM** consistently achieves the highest accuracy across all architectures, reaffirming its effectiveness in capturing discriminative features for malware image classification.
- **MobileNetV3Small** and **ResNet50** generally outperform **MobileNetV3Large** in terms of accuracy for most classifiers. This suggests that these architectures might be better suited for extracting features relevant to malware image classification.
- **KNN** consistently shows good performance across all architectures, with fast training times.
- **Decision Tree** and **Naive Bayes** generally have lower accuracy compared to SVM, KNN, and Random Forest.

Now we will apply BackPropagation for enhancing the feature extraction process.

Backpropagation in CNN

Backpropagation in the context of convolutional neural networks (CNNs) is a method used to train the network by adjusting its weights based on the error rate obtained in the previous epoch (i.e., each full pass through the training dataset).

Forward Pass: During the forward pass, input data (like an image) is passed through the CNN layers until it reaches the output layer, where a prediction is made.

Error Calculation: The error (or loss) is calculated at the output by comparing the prediction with the actual label or value. This error tells us how far off our predictions are from the actual results.

CONT'D

Backward Pass (Backpropagation): The error is then propagated back through the network, starting from the output layer to the input layer. This process involves calculating the gradient (or change) of the error with respect to each weight in the network using calculus. This gradient tells us how much a change in each weight contributed to the error.

Weight Update: The weights are then adjusted slightly to decrease the error. The size of the adjustment is controlled by a parameter called the learning rate. If the learning rate is too large, it may cause unstable training; if it's too small, the training process can become very slow.

Iteration: This process is repeated (forward pass, error calculation, backpropagation, and weight update) for many iterations over the entire dataset until the network weights stabilize or a maximum number of iterations is reached.

Why Use BP for Feature Enhancement?

Optimization of Weights: Backpropagation helps in fine-tuning the weights of a neural network, ensuring that the model learns the most discriminative features for the task at hand.

Improved Accuracy: By optimizing the neural network's parameters, backpropagation can lead to higher classification accuracy, as the network better adapts to the complexities of the data.

Adaptation to Data: It allows the network to adapt to specific characteristics of the dataset, which is crucial for tasks like image-based malware classification where each type of malware might exhibit unique features.

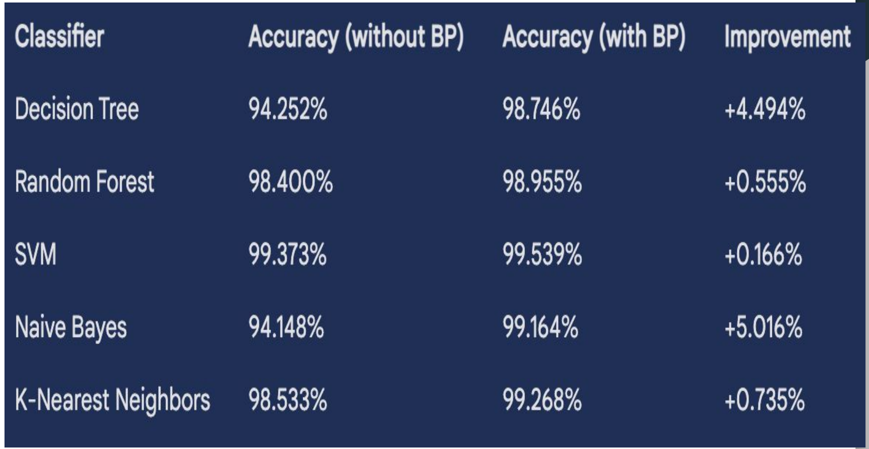
Reduction of Overfitting: By iteratively adjusting the network, backpropagation can help in minimizing overfitting, especially when combined with techniques like dropout or regularization.

Results on MobNetV1 after applying BP

Classification Algorithm	Accuracy	Precision	Recall	F1_Score	TPR	TNR	FPR	FNR	FDR	FOR
Decision Tree	98.746	0.9879	0.9874	0.9875	0.9717	0.9994	0.0005	0.0282	0.0295	0.0005
Random Forest	98.955	0.9904	0.9895	0.9896	0.9777	0.9995	0.0004	0.0222	0.0245	0.0004
Support Vector Classifier	99.539	0.9922	0.9916	0.9918	0.9811	0.9996	0.0003	0.0188	0.0199	0.0003
Naive-Bayes Classifier	99.164	0.9918	0.9916	0.9916	0.9808	0.9996	0.0003	0.0191	0.0160	0.0003
K-Nearest Neighbors	99.268	0.9929	0.9926	0.9927	0.9817	0.9997	0.0002	0.0182	0.0173	0.0002

Analysis after applying BP

- **Substantial Accuracy Gains:** Backpropagation leads to notable accuracy improvements for all classifiers, especially Decision Tree (+4.494%) and Naive Bayes (+5.016%). This highlights the effectiveness of fine-tuning the CNN to extract more discriminative features for malware classification.
- **Enhanced Feature Representation:** Backpropagation allows the CNN to adapt its internal parameters to the specific characteristics of malware images, resulting in features that are more relevant and informative for the classification task.



Classifier	Accuracy (without BP)	Accuracy (with BP)	Improvement
Decision Tree	94.252%	98.746%	+4.494%
Random Forest	98.400%	98.955%	+0.555%
SVM	99.373%	99.539%	+0.166%
Naive Bayes	94.148%	99.164%	+5.016%
K-Nearest Neighbors	98.533%	99.268%	+0.735%

CONT'D

Classifier-Specific Insights:

- **Decision Tree:** The significant improvement suggests that backpropagation helps to create features that are more suitable for the decision-making process of Decision Trees.
- **Naive Bayes:** The substantial gain indicates that the fine-tuned features better align with the assumptions of Naive Bayes (feature independence).
- **Random Forest, SVM, and KNN:** While these classifiers already performed well without backpropagation, they still benefit from the enhanced feature representation, showing further improvements in accuracy.

References

[1]F. Zhong, Z. Chen, M. Xu, G. Zhang, D. Yu and X. Cheng, "Malware-on-the-Brain: Illuminating Malware Byte Codes With Images for Malware Classification," in *IEEE Transactions on Computers*, vol. 72, no. 2, pp. 438-451, 1 Feb. 2023, doi: 10.1109/TC.2022.3160357. keywords: {Malware;Feature extraction;Codes;Security;Static analysis;Libraries;Visualization;Classification;histogram equalization;malware;visualization}

[2]L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, “Malware images: visualization and automatic classification,” In Proceedings of the 8th International Symposium on Visualization for Cyber Security, 2011 (VizSec '11). Association for Computing Machinery, New York, NY, USA, Article 4, 1–7. DOI: https://doi.org/10.1145/2016904.2016908.

[3] M. Kalash, M. Rochan, N. Mohammed, N. D. B. Bruce, Y. Wang and F. Iqbal, "Malware Classification with Deep Convolutional Neural Networks," 2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS), 2018, pp. 1-5, doi: 10.1109/NTMS.2018.8328749.

CONT'D

[4] Y. Liu, Y. Lai, Z. Wang and H. Yan, "A New Learning Approach to Malware Classification Using Discriminative Feature Extraction," in IEEE Access, vol. 7, pp. 13015-13023, 2019, doi: 10.1109/ACCESS.2019.2892500.

[5]Redundant Multi-Object Detection for Autonomous Vehicles in Structured Environments - Scientific Figure on ResearchGate. Available from:https://www.researchgate.net/figure/CNN-MobileNet-v1-architecture_fig2_35498560

Thank You