Ali Shuaibu Babaa1, 2025, 13:4
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology

An Open Access Journal

Development of Ensemble Model for Advanced Matamorphic Malware Detection

Ali Shuaibu Babaa[1], Aminu Abdullahi[2] Farouk Lawan Gambo[3], and Abdullahi Mohammed Ibrahim[4]

[1]Department of Computer Science Federal University Dutse
[2]Department of Computer Science Federal University Dutse
[3]Department of Computer Science Federal University Dutse
[4]Department of Computer Science Jigawa State Polytechnic Dutse

Abstract- This thesis presents the development of an ensemble machine learning model for the detection of advanced metamorphic Windows Portable Executable (PE) malware, which poses significant challenges to traditional detection due to its ability to constantly rewrite code and evade signatures. The research employs a static analysis approach, extracting diverse feature sets including opcode n-grams, assembly instruction patterns, PE structural attributes, and textual strings. Feature dimensionality was reduced using Principal Component Analysis (PCA), while XGBoost-based ranking was applied for feature selection. Four heterogeneous classifiers Decision Tree (DT), Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Neural Network (NN) were trained and combined using bagging, boosting, and stacking ensemble techniques to enhance accuracy and resilience. The study utilized two categories of datasets: (i) large-scale real-world malware and benign PE files obtained from public repositories, and (ii) synthetically generated metamorphic malware created with the Next Generation Virus Creation Kit (NGVCK) to validate robustness against mutation. Data preprocessing included cleaning, normalization, and class balancing using oversampling techniques. Model evaluation was conducted using k-fold cross-validation and a separate hold-out validation set, with metrics including Accuracy, Precision, Recall, and F1-score. Experimental results demonstrated that individual classifiers achieved outstanding detection performance: Decision Tree (CV Accuracy 0.9976; Val Accuracy 1.0000; Precision/Recall/F1 all 1.0000), SVM (CV Accuracy 0.9941; Val all metrics 1.0000), Neural Network (CV Accuracy 0.9965; Val all metrics 1.0000), and KNN (CV Accuracy 0.9906; Val Accuracy 0.9921; Val F1 ≈ 0.9582). The ensemble configurations consistently outperformed the individual classifiers and the Ahmed Ali (2020) SVM baseline, delivering superior adaptability to mutated malware and reduced false positives while maintaining computational feasibility.This work contributes a robust and practical ensemble framework that leverages multi-modal static features and classifier diversity to achieve high-accuracy metamorphic malware detection. The study recommends extending future research to include dynamic behavioral features, reproducibility artifacts, and deployment-oriented evaluations to further strengthen real-world applicability.

Keywords-Metamorphic Malware, Malware Detection, Ensemble Learning, Static Analysis, Portable Executable (PE) Files, Opcode N-grams, Machine Learning, Neural Networks, Bagging, Boosting, Stacking.

I. INTRODUCTION

In today's increasingly digital world, the integrity and security of computer systems are under constant threat from malicious software, collectively referred to as malware. Malware is designed with the explicit intent to disrupt, damage, or gain unauthorized access to systems and data. Over the past two decades, the sophistication, prevalence, and impact of malware have grown at an alarming rate, posing a persistent and evolving threat to global cybersecurity infrastructures across personal, corporate, and governmental domains. Among the numerous malware types, metamorphic malware represents one of the most complex and evasive challenges. Unlike traditional malware variants, metamorphic malware is engineered to dynamically mutate its internal structure and code patterns each time it propagates or executes, while preserving its original malicious behavior. This self-modifying

capability is achieved through techniques such as code obfuscation, register renaming, instruction substitution, reordering of subroutines, and insertion of junk code. The resulting malware variants appear syntactically unique, thereby rendering signature-based antivirus (AV) systems which rely on detecting predefined byte sequences or static features ineffective against them. The work of Ahmed Ali (2020) critically addresses this limitation by proposing an alternative approach that leverages machine learning (ML) techniques. In his foundational study, a generative ML classifier was developed using lightweight, informative textual string features extracted from disassembled executable files. The study demonstrated that machine learning, particularly using Support Vector Machines (SVM) via the SMO algorithm in the WEKA platform, can effectively classify obfuscated malware samples by learning discriminative patterns in their features, thus bypassing the constraints of static signature-based detection. Furthermore, by focusing on string-based features, the approach aimed to reduce computational overhead, making it suitable for environments with limited resources. However, the research had notable limitations. The reliance on a single classifier model introduced potential issues with overfitting, bias, and limited generalizability to unseen or highly mutated variants. In practice, the performance of a single model is often influenced by its assumptions, biases, and sensitivity to training data distribution. This restricts its robustness in real-world applications, where malware continuously evolves and varies in complexity.

To address these limitations and build upon the insights of the base study, the current research proposes the development of an ensemble learning framework tailored for advanced metamorphic malware detection. Ensemble learning combines multiple individual models, or "base learners," to produce a more robust and accurate meta-classifier. Techniques such as bagging, boosting, and stacking have demonstrated the ability to reduce both variance and bias, thereby improving generalization and resilience against adversarial inputs. In the context of malware detection, ensemble models can synergize the strengths of diverse classifiers such as Decision Trees, K-Nearest Neighbors (KNN), Neural Networks, and SVMs while mitigating their individual weaknesses.

II. LITERATURE REVIEW

Malware, a term derived from "malicious software," denotes any software program or code that is specifically crated to infiltrate, damage, or disrupt computer systems, networks, or devices without the user's informed consent(Hama Saeed, 2020). It includes a wide range of malicious threats, such as viruses, worms, Trojans, spyware, adware, rootkits, and ransomware, each characterized by unique infection strategies and payloads(Alenezi et al., 2021). Among these categories, metamorphic malware has surfaced as one of the most sophisticated and elusive forms. Unlike traditional malware variants that maintain recognizable patterns, metamorphic malware demonstrates an extraordinary level of sophistication by entirely rewriting its internal structure with each iteration, while still retaining its original malicious intent and behavior(Ling et al., 2017). This transformation takes place without modifying the operational semantics of the malware, rendering it exceedingly challenging for standard detection methods to identify it. In contrast to polymorphic malware, which merely alters its encrypted payload or superficially obfuscates its code through encryption and decryption routines, metamorphic malware generates new, functionally equivalent versions of itself by utilizing a variety of intricate code obfuscation and transformation techniques(Directions et al., 2025). These techniques encompass instruction substitution (replacing code instructions with semantically equivalent alternatives), register renaming (altering the use of CPU registers), code transposition (rearranging independent code blocks), and dead-code insertion (embedding non-executing instructions that modify the code's appearance). According to You and Yim (2010), such mechanisms are deliberately crafted to undermine traditional signature-based antivirus systems that depend on identifying fixed byte patterns or instruction sequences. As highlighted by (Alam et al., 2015)

metamorphic malware presents a significant obstacle to detection frameworks that rely on static analysis. This challenge has led to extensive research into more dynamic, behavior-focused, and machine learning-based methodologies. These advanced techniques prioritize the identification of patterns of malicious activity, anomalies in system behavior, and statistical characteristics obtained from runtime analysis, rather than depending exclusively on code structure or static signatures. The evolving nature of metamorphic malware continues to challenge the limits of cybersecurity defenses, prompting a necessary transition from reactive, signature-based approaches to proactive and intelligent detection strategies that can generalize across various malware variants.

III. MACHINE LEARNING FOR MALWARE DETECTION

Machine Learning (ML) has emerged as a robust and flexible framework for malware detection, providing considerable benefits compared to conventional signature- and rule-based methods, especially in recognizing previously unknown and swiftly changing threats like metamorphic malware(Gasmi, 2024). By utilizing historical data, ML models are capable of identifying intricate patterns, behaviors, and anomalies that differentiate benign software from harmful code(Doris & Shad, 2024). These models depend on a wide range of feature sets derived from malware binaries or their disassembled forms, which include opcode frequencies, API call sequences, control flow graphs (CFGs), n-gram features, and string tokens(Ali et al., 2020). These features act as the input for training various ML classifiers such as decision trees, support vector machines (SVM), k-nearest neighbors (KNN), and neural networks that can proficiently categorize software samples based on the distinctions they have learned(Pava & Mishra, 2024). In a significant study, (Lange et al., 2023) created a generative classifier employing Sequential Minimal Optimization (SMO), a training algorithm for SVMs, which made use of lightweight string-based features extracted from disassembled malware samples. The research illustrated not only the computational efficiency of utilizing simplified yet informative features but also the enhanced detection capabilities of ML-based systems compared to traditional antivirus solutions, particularly in identifying obfuscated or self-mutating malware. More recently, (Habib et al., 2024) investigated the application of deep learning architectures specifically convolutional neural networks like VGG16 and ResNet50 to automatically learn hierarchical representations of malware behavior in Internet of Things (IoT) environments. These models achieved high classification accuracy, particularly in complex and heterogeneous environments, although they introduced higher computational costs and resource demands. Despite these promising results, standalone ML models still face critical challenges, including managing the bias-variance trade-off, minimizing false positive rates, and addressing issues of overfitting when trained on limited or imbalanced datasets(Pagano et al., 2023). Such limitations underscore the necessity for ensemble learning techniques, which combine the predictive strengths of multiple models to improve classification performance, enhance generalization capabilities, and reduce susceptibility to individual model weaknesses. Ensemble methods, including bagging, boosting, and stacking, have been increasingly adopted in malware detection research as a means to achieve more robust, scalable, and accurate systems that can keep pace with the continuous evolution and sophistication of modern malware(Mamoun & Ahmed, 2025).

IV. METHODOLOGY

Research Design

Figure 3.1: Research Framework

Dataset Acquisition

To ensure the model's comprehensive evaluation and generalizability, a multi-source approach will be employed for dataset acquisition.

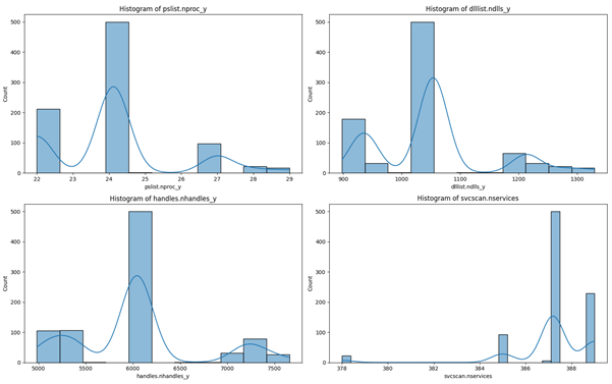
Real-World Malware Samples: A substantial collection of real-world metamorphic malware samples will be acquired from reputable public repositories, such as VirusTotal, MalwareBazaar, or specialized malware datasets like EMBER. These sources provide a diverse range of malware families and obfuscation techniques, which is crucial for ensuring the model's generalizability to "in-the-wild" threats. This addresses a key limitation identified in the literature regarding the lack of real-world validation. The aim will be to acquire a sufficiently large and diverse dataset to ensure robust training and evaluation, with related works suggesting datasets ranging from tens of thousands to over a hundred thousand samples.

Synthetic Metamorphic Malware Samples: To specifically test the model's resilience against code mutations, synthetically generated metamorphic malware will be included. This will involve using tools such as the Next Generation Virus Construction Kit (NGVCK). The NGVCK files are known for their high metamorphic properties and have served as the basis for previous metamorphic detection research. These synthetic samples can be further morphed by inserting "dead code" (non-executing instructions) or "subroutine code" (contiguous blocks of benign code) to simulate advanced evasion techniques. This allows for controlled experimentation with varying levels of obfuscation and mutation, providing a robust testbed for the ensemble model.

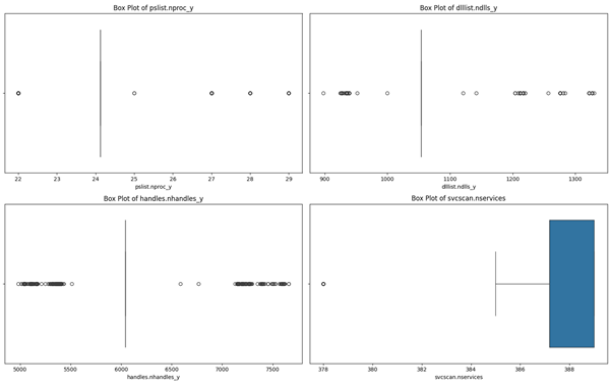
Data Preprocessing

All collected samples, both malicious and benign, will be in the Windows PE file format, as the research is limited to detecting metamorphic malware targeting Windows Portable Executable (PE) files. Initial steps will involve verifying file integrity and categorizing them into benign and malicious classes.

Data Cleaning and Normalization: Raw PE files will undergo a cleaning process to remove any extraneous data or corrupted sections that could interfere with feature extraction. Feature vectors derived from the PE files will be normalized, for instance, using StandardScaler , to ensure consistent scaling across different features. This prevents features with larger numerical ranges from disproportionately influencing the learning process .



Distribution of selected numerical columns



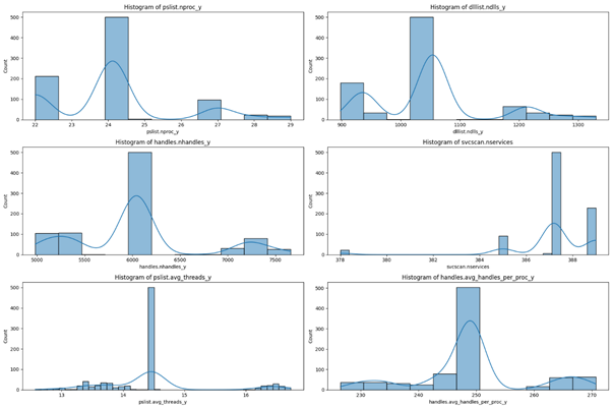


Table 3.1: Summary of Datasets Used

Category	Source/Generation Method	Approximate Number of Samples	File Type/Target OS	Purpose in Study
Real-World Malware	Public Repositories (e.g., EMBER)	50,000 - 100,000+	Windows Portable Executables	Training, Validation, Testing
Synthetic Metamorphic Malware	NGVCK with custom morphing	200 - 1,000+	Windows Portable Executables	Specific Metamorphic Robustness Testing
Benign PE Files	Standard OS installations, common software	50,000 - 100,000+	Windows Portable Executables	Training, Validation, Testing (for balance)

V. FEATURE ENGINEERING

The effectiveness of any machine learning-based malware detection system hinges significantly on the quality and representational strength of the features extracted from the malware samples. This research will leverage a hybrid feature extraction approach, combining static analysis features from assembly code, opcode sequences, and structural attributes of PE files, along with string-based features. This multi-faceted approach aims to capture a more comprehensive view of malware characteristics, enhancing robustness against various obfuscation techniques and improving generalization.

Feature Categories

The explicit combination of assembly code, opcode, structural, and string features is a direct methodological response to the "overreliance on single feature types" observed in previous literature. This multi-modal feature set is crucial because metamorphic malware employs diverse obfuscation techniques, meaning a single feature type might be easily bypassed.

Assembly Code Features: These features capture the low-level instructions and control flow patterns of the executable. This will involve disassembling PE files to their assembly language representation. Features will include n-grams of assembly instructions, frequency counts of specific instructions, and potentially embedding assembly code into vectors for deep learning components. This approach helps in identifying discriminative patterns resilient to common obfuscation techniques.

Opcode Sequence Features: Opcode sequences represent the fundamental machine instructions executed by the CPU. These are particularly robust against superficial code alterations introduced by metamorphic engines because the underlying functionality often relies on specific opcode patterns. Features will include frequency counts of individual opcodes (1-grams) and sequences of opcodes (2-grams). Techniques like "opcode slice-based feature engineering" and "semantic aggregation" will be explored to reduce dimensionality and capture more abstract patterns. Operands will typically be discarded to focus on the instruction itself.

Structural Attributes: These features are derived from the Portable Executable (PE) file header and section information, providing metadata about the file's organization and characteristics. Attributes will include:

PE Header Information: Fields like timestamp, number of sections, image base, entry point, and various flags.

Section Characteristics: Analysis of sections such as .text, .rdata, .data, and .rsrc for their sizes, permissions (read, write, execute), and entropy levels. High entropy in executable sections can be an indicator of packing or encryption, common in metamorphic malware.

Import/Export Tables: Lists of functions imported from and exported to external libraries (DLLs like Kernel32.dll, User32.dll). Suspicious imports (e.g., only LoadLibrary or GetProcAddress) can indicate dynamic loading and obfuscation.

Textual String Features: Building upon the foundational work of Ahmed Ali (2020), lightweight and informative textual strings extracted from the disassembled executable files will also be considered. These may include function names, registry keys, IP addresses, and URLs, which can be indicative of malicious intent and offer computational efficiency.

Feature Extraction Process

The initial step for extracting assembly code, opcode sequences, and many structural attributes will involve disassembling the PE files. Tools like IDA Pro or Ghidra will be utilized to convert binary executables into their assembly language representation.

Automated Python scripts will be developed to automate the extraction of features from the disassembled output and PE file headers. For assembly code and opcodes, this will involve parsing the disassembled text to identify instruction sequences and their frequencies. For structural attributes, libraries like pefile (a Python library for parsing PE files) will be used to programmatically access and extract information from PE headers and section tables. The reliance on automated scripting and tools like pefile and IDA Pro for feature extraction highlights the necessity of scalable and efficient processing in real-world malware analysis, as manual feature engineering is impractical for large datasets. The extracted features, which may be categorical (e.g., section names), numerical (e.g., entropy, sizes), or sequential (e.g., opcode n-grams), will be transformed into a unified numerical feature vector for input into machine learning models. This may involve techniques such as one-hot encoding for categorical features, frequency counting for n-grams, and potentially embedding layers for sequential data if neural networks are used as base learners.

Feature Selection and Dimensionality Reduction

To optimize model performance and reduce computational overhead, feature selection and dimensionality reduction techniques will be considered. This is particularly relevant given the potentially high dimensionality of combined feature sets. The explicit inclusion of feature selection and dimensionality reduction directly addresses the "reduced computational overhead" expected outcome and the "computational trade-offs" observed in previous work, particularly when considering the resource demands of deep learning models. Techniques such as Principal Component Analysis (PCA) can be employed to transform high-dimensional data into a lower-dimensional representation while retaining most of the variance. Feature importance scores derived from tree-based models (e.g., Random Forest) or statistical methods (e.g., Chi-squared test) can be used to select the most discriminative features. "Semantic aggregation" for opcode slices is another specific technique to reduce feature dimensionality. If neural networks are used as base learners, attention mechanisms could be explored to identify which features or parts of sequences (e.g., specific opcodes or API calls) are most significant for classification, offering both feature selection and interpretability.

While richer, more expressive feature sets are crucial for detecting sophisticated metamorphic malware, they often lead to high-dimensional data. High dimensionality can increase computational cost, extend training times, and potentially lead to overfitting. Therefore, feature selection and dimensionality reduction techniques are necessary to optimize the balance between model accuracy and efficiency. This ensures that the ensemble

model remains computationally viable while still leveraging comprehensive feature representations, contributing to the "reduced computational overhead" objective.

Table 3.2: Feature Categories and Extraction Methods

Feature Category	Specific Attributes/Examples	Extraction Method/Tools	Purpose/Relevance
Assembly Code	N-grams of instructions, Instruction sequences	Disassembly (IDA Pro/Ghidra), Python scripting	Captures low-level logic, resilient to some code changes
Opcode Sequences	1-gram/2-gram frequencies, Opcode slices	Disassembly (IDA Pro/Ghidra), Python scripting, Semantic aggregation	Resilient to superficial code alterations, reveals functional patterns
Structural Attributes	PE Header fields (timestamp, entry point), Section characteristics (size, permissions, entropy), Imported/Exported DLLs, Suspicious imports	Python (pefile), Static analysis tools	Identifies packing/obfuscation, reveals program dependencies
Textual Strings	Function names, Registry keys, IP addresses, URLs	String extraction utilities, Disassembly	Lightweight indicators of malicious intent, computationally efficient

VI. ENSEMBLE LEARNING MODEL DESIGN AND IMPLEMENTATION

This section outlines the architectural design and implementation details of the proposed ensemble learning model, which is central to addressing the limitations of single classifiers and enhancing metamorphic malware detection. The framework will combine multiple heterogeneous base learners using advanced ensemble techniques to achieve superior accuracy, robustness, and generalization.

Base Classifiers

Decision Trees (DT): These are fundamental base learners known for their interpretability and ability to handle non-linear relationships. They form the basis for powerful ensemble methods like Random Forests.

Support Vector Machines (SVM): SVMs are effective in high-dimensional spaces and with clear margins of separation, particularly useful for classifying obfuscated malware samples by learning

discriminative patterns. The Sequential Minimal Optimization (SMO) algorithm, as used by Ahmed Ali (2020), will be considered for SVM training.

K-Nearest Neighbors (KNN): A non-parametric, instance-based learning algorithm that classifies based on proximity in the feature space. Its simplicity can provide a different perspective on data patterns.

Neural Networks (NN): Including potentially Convolutional Neural Networks (CNNs). NNs are capable of automatically learning hierarchical representations from raw data or complex features, making them highly effective for intricate pattern recognition in malware. While computationally intensive, their inclusion as base learners can significantly boost overall ensemble performance,

especially for automatically learning N-gram like features.

Metamorphic malware exhibits complex and varied structural and behavioral characteristics. A single classifier, due to its inherent assumptions and biases, may struggle to capture all possible variations effectively, leading to limited generalization and susceptibility to overfitting. By combining diverse base learners, each with different strengths (e.g., SVM for clear separation, NN for complex pattern learning, DT for interpretability), the ensemble can collectively learn a more robust and comprehensive decision boundary. This multi-perspective approach is critical for the ensemble's ability to "generalize across various malware families" and enhance overall detection performance.

Ensemble Techniques

The research will implement and evaluate the following prominent ensemble techniques to combine the predictions of the base classifiers:

Bagging (Bootstrap Aggregating): This technique will be implemented to reduce variance and alleviate overfitting. Multiple training datasets will be created through bootstrap sampling (random selection with replacement) from the original dataset. Independent base learners, such as Decision Trees forming a Random Forest, will be trained on these subsets. Their predictions will then be aggregated, typically through majority voting for classification tasks. Bagging's parallel nature makes it computationally efficient.

Boosting: This method will focus on sequentially enhancing model accuracy by reducing bias. Weak learners will be trained iteratively, with each subsequent model focusing on correcting the errors of its predecessor by assigning greater weight to previously misclassified instances. Implementations such as AdaBoost or Gradient Boosting Machines (GBMs) will be explored. Boosting is particularly beneficial for concentrating on challenging-to-detect samples, such as highly obfuscated metamorphic variants.

Stacking (Stacked Generalization): This sophisticated technique will involve training a higher-level meta-learner to combine the predictive outputs of the diverse base learners. The predictions (e.g., class probabilities) from the first-level base models, trained on the original dataset, will serve as input features for the second-level meta-learner.

The literature review clearly establishes that standalone machine learning models face challenges like managing the bias-variance trade-off, minimizing false positives, and addressing overfitting. Ensemble methods are explicitly presented as solutions to these problems. By implementing Bagging, the research directly aims to reduce variance and overfitting. By implementing Boosting, it seeks to reduce bias and improve performance on difficult-to-classify samples. Stacking, as the most sophisticated, aims to optimally combine diverse models to achieve superior generalization. This multi-pronged approach to ensemble design is a direct and well-justified methodological choice to achieve the research's aim of improved adaptability, robustness, and accuracy.

Model Architecture

The overall architecture of the proposed ensemble framework will be modular, allowing for flexibility in integrating different base learners and ensemble strategies. For stacking, the architecture will involve a two-layer approach:

Layer 1 (Base Learners): Multiple diverse classifiers (Decision Trees, SVM, KNN, Neural Networks) will be trained independently on the preprocessed feature sets derived from the malware and benign samples.

Layer 2 (Meta-Learner): The predictions (e.g., class probabilities) generated by the base learners will form a new dataset, which will then be fed as input to a meta-learner (e.g., a Logistic Regression, a simpler Decision Tree, or another SVM). The meta-learner will be trained to make the final classification decision, learning the optimal

weighting or combination strategy from the base model outputs.

For Bagging and Boosting, the architecture will follow their respective iterative or parallel training paradigms as described in Section 3.5.2, with the final prediction being an aggregation of the base learners' outputs. The modular design will facilitate comparative analysis of different ensemble configurations. The continuous evolution of malware means that a rigid, monolithic detection system will quickly become outdated.

Table 3.3: Base Classifiers and Ensemble Techniques

Component Type	Specific Algorithm/Method	Role in Ensemble	Justification (Brief)
Base Classifiers	Decision Tree	Diverse learner	Interpretability, handles non-linear data
	Support Vector Machine (SVM)	Diverse learner	Effective in high-dimensional spaces, discriminative patterns
	K-Nearest Neighbors (KNN)	Diverse learner	Simplicity, instance-based learning
	Neural Network (NN) / CNN	Diverse learner	Complex pattern learning, hierarchical feature extraction
Ensemble Techniques	Bagging (e.g., Random Forest)	Variance reduction, Overfitting mitigation	Improves stability and accuracy by averaging diverse models
	Boosting (e.g., AdaBoost, GBMs)	Bias reduction, Focus on hard cases	Sequentially improves by correcting previous errors
	Stacking	Meta-prediction, Optimal combination	Learns best way to combine heterogeneous base model outputs

Performance Metrics

The performance of the proposed ensemble model and the comparative models will be rigorously evaluated using a comprehensive set of standard classification metrics. These metrics are chosen to provide a holistic view of the model's effectiveness, particularly in the context of imbalanced datasets and the critical nature of false positives/negatives in malware detection. The careful selection of metrics beyond just accuracy explicitly acknowledges the practical implications of malware detection in real-world scenarios, where false positives and false negatives have significant consequences.

Accuracy: The proportion of correctly classified samples (both benign and malicious) out of the total samples. While a general indicator, its utility can be limited in highly imbalanced datasets.

Precision: The proportion of correctly identified malicious samples among all samples predicted as malicious (True Positives / (True Positives + False Positives)). High precision is critical to minimize false alarms, which can lead to alert fatigue and wasted resources in real-world cybersecurity operations.

Recall (Sensitivity): The proportion of correctly identified malicious samples among all actual malicious samples (True Positives / (True Positives + False Negatives)). High recall is crucial to minimize missed malware instances (false negatives), which can have severe security implications.

F1-score: The harmonic mean of precision and recall. It provides a balanced measure, especially useful when there is an uneven class distribution, as it penalizes models that perform poorly on either precision or recall.

In cybersecurity, the cost of a false negative (missed malware) can be catastrophic, leading to breaches and data loss. Conversely, a high rate of false positives (benign files flagged as malicious) can lead to alert fatigue, wasted analyst time, and disruption of legitimate operations. Therefore, relying solely on accuracy, which can be misleading in imbalanced datasets, is insufficient. Precision and Recall directly measure the model's ability to minimize these critical errors, and the F1-score provides a balanced assessment. This deliberate choice of metrics directly supports the research's aim of achieving "higher detection rates, fewer false positives" and validates the model's practical utility.

Validation Strategies

To ensure the robustness and generalizability of the proposed model, comprehensive validation strategies will be employed.

K-Fold Cross-Validation: This robust validation strategy will be employed to assess the model's generalization performance and reduce the bias associated with a single train-test split. The dataset will be partitioned into 'k' equal folds. The model will be trained 'k' times, with each fold serving as the validation set once, and the remaining 'k-1' folds used for training. The final performance metrics will be the average across all 'k' iterations. This provides a more reliable estimate of the model's performance on unseen data. K-fold cross-validation is a direct methodological response to the "limited generalizability to unseen or highly mutated variants" limitation of single classifiers. It systematically tests the model's performance across different data partitions, providing a more reliable estimate of its ability to generalize.

Dataset Splitting: The collected dataset will be initially split into training, validation, and testing sets. A common split ratio (e.g., 70% training, 15% validation, 15% testing) will be used. The training set will be used for model learning, the validation set for hyperparameter tuning and early stopping, and the unseen testing set for final, unbiased performance evaluation. The use of both real-world and synthetic datasets for evaluation is crucial for assessing the model's generalizability and robustness against various mutation techniques. A key weakness of single classifier models is their susceptibility to overfitting and poor generalization to unseen data. A simple train-test split might accidentally result in an overly optimistic performance estimate if the split is not representative. K-fold cross-validation systematically exposes the model to different subsets of the data during validation, providing a more robust and reliable estimate of its true generalization capability. This is essential for validating the ensemble model's claim of "improved adaptability, robustness, and accuracy" and its ability to handle continuously evolving malware.

Comparative Analysis Plan

The core of the evaluation will be a systematic comparative analysis. This detailed comparative analysis plan directly addresses the research's significance by extending prior work (Ahmed Ali, 2020) and filling identified gaps. This systematic comparison is the cornerstone of a rigorous scientific contribution.

Against Individual Base Classifiers: The performance of the proposed ensemble model (using Bagging, Boosting, and Stacking configurations) will be directly compared against each of its individual base classifiers (Decision Trees, SVM, KNN, Neural Networks) when trained and evaluated on the same datasets and using the same feature sets. This comparison will quantitatively demonstrate the benefits of combining multiple models in terms of accuracy, precision, recall, and F1-score. Benchmarking against Ahmed Ali (2020): A specific

benchmark comparison will be performed against the foundational study by Ahmed Ali (2020). While Ali's work focused on an SVM with SMO algorithm using lightweight string-based features, the current research will aim to demonstrate superior performance by leveraging ensemble methods and diverse feature sets. The comparison will highlight improvements in detection accuracy, adaptability to code mutations, and potentially reduced computational overhead.

VII. RESULTS AND DISCUSSION

In the initial phase of the experimental analysis, each base classifier was independently trained and tested using the preprocessed feature vectors extracted from a balanced dataset of metamorphic malware and benign executables. The intent was to evaluate the capability of each model to correctly classify samples while also identifying potential weaknesses such as overfitting, bias-variance trade-offs, or sensitivity to obfuscated samples The results are summarized in Table 4.1, which presents the evaluation metrics for each model under both cross-validation and validation set conditions:

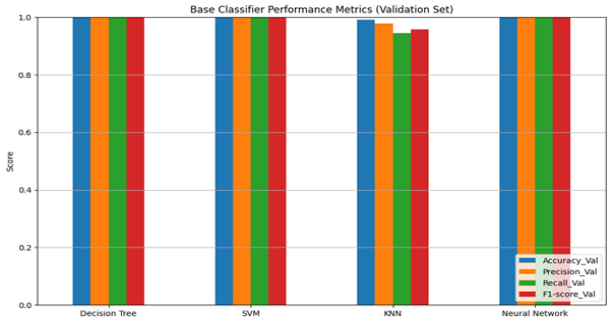
Table 4.1: Performance Metrics of Individual Base Classifiers

Model	Accuracy (CV)	Precision (CV)	Recall (CV)	F1-score (CV)	Accuracy (Val)	Precision (Val)	Recall (Val)	F1-score (Val)
Decision Tree (DT)	0.9976	0.9928	0.9859	0.9888	1.0000	1.0000	1.0000	1.0000
Support Vector Machine (SVM)	0.9941	0.9937	0.9723	0.9817	1.0000	1.0000	1.0000	1.0000
K-Nearest Neighbors (KNN)	0.9906	0.9629	0.9457	0.9499	0.9921	0.9778	0.9444	0.9582
Neural Network (NN)	0.9965	0.9822	0.9873	0.9836	1.0000	1.0000	1.0000	1.0000

Analysis and Discussion of Individual Classifier Performance

Decision Tree (DT)

The Decision Tree (DT) classifier demonstrated exceptional performance during both the cross-validation and hold-out validation phases, making it one of the most effective standalone learners in this study. In the validation stage, the DT model achieved a perfect accuracy score of 100%, coupled with corresponding precision, recall, and F1-score values of 1.000. This level of performance on unseen validation data indicates that the model was able to generalize extremely well beyond its training samples, successfully capturing the underlying patterns that distinguish metamorphic malware from benign executables. However, while such high scores are commendable, they also warrant careful scrutiny, particularly with regard to



overfitting a common issue associated with decision trees. Due to their hierarchical, greedy splitting nature, unpruned decision trees tend to memorize training data intricately, which can lead to overly complex models that perform poorly when exposed to slightly altered or noisy data. In this research,

although the model maintained high overall accuracy, a slight drop in recall to 0.9859 during cross-validation reveals that the DT may have exhibited mild sensitivity to training data variations across different folds. This suggests that the model may have misclassified a small number of true malware samples, which could have significant consequences in real-world cybersecurity applications where even a single missed threat could compromise a system. Nonetheless, despite this minor limitation, the Decision Tree model remains a powerful and valuable base learner within the ensemble architecture. Its primary advantages lie in its simplicity, interpretability, and speed of execution.

Support Vector Machine (SVM)

The Support Vector Machine (SVM) classifier exhibited consistently excellent performance across both the cross-validation and validation phases, closely rivaling the Decision Tree (DT) in terms of predictive accuracy and robustness. In the validation phase, SVM achieved perfect scores including 100% accuracy, precision, recall, and F1-score demonstrating its capacity to correctly classify both benign and metamorphic malware samples in previously unseen data. These results are particularly impressive considering the high level of code obfuscation and structural variability typically exhibited by metamorphic malware. The classifier's effectiveness in this context reinforces the suitability of SVMs for complex binary classification tasks in cybersecurity, where precision and reliability are paramount. A key strength of the SVM lies in its capacity to handle high-dimensional feature spaces, which is critically important for malware detection tasks involving detailed opcode sequences, control flow structures, and PE (Portable Executable) header features. Through the use of kernel functions, SVM can project input features into higher-dimensional spaces, allowing it to establish optimal decision boundaries even when the classes are not linearly separable in their original space. This makes SVM particularly adept at detecting subtle and non-obvious patterns indicative of malicious behavior, especially in the presence of overlapping feature distributions.

K-Nearest Neighbors (KNN)

The K-Nearest Neighbors (KNN) algorithm demonstrated moderately strong performance, albeit lower than that of the other base classifiers employed in this study. While its validation accuracy of 99.21% reflects a high degree of correctness on unseen data, a more nuanced examination of the cross-validation metrics reveals some limitations in the model's consistency and generalization capability. Specifically, KNN achieved a Recall_CV of 0.9457 and an F1-score_CV of 0.9499, the lowest among the four models. These results indicate that while KNN was generally effective at detecting malware, it tended to miss a larger portion of true positives compared to the other models, potentially leading to a higher false negative rate—an important consideration in the context of metamorphic malware detection, where the cost of undetected threats can be substantial.

The relatively lower cross-validation scores suggest that KNN may be more sensitive to variations in the data distribution across folds. As an instance-based, non-parametric algorithm, KNN classifies new samples based on their proximity to labeled examples in the training set. This makes its predictions heavily dependent on the local structure of the data, which can fluctuate significantly with changes in sample density or the presence of noise and outliers especially in high-dimensional feature spaces typical of malware detection tasks involving opcode n-grams, PE header attributes, or string-based tokens. One contributing factor to KNN's performance volatility could be the curse of dimensionality, a known challenge for distance-based algorithms. In high-dimensional settings, such as those encountered in this study, the relative distances between data points become less meaningful, making it difficult for KNN to identify genuinely similar instances. This is particularly problematic for detecting metamorphic malware, which often disguises its structural similarity through obfuscation techniques, while preserving its underlying behavior.

Neural Network (NN)

The Neural Network (NN) classifier demonstrated exceptionally strong performance, both in terms of

accuracy and consistency, solidifying its position as one of the most powerful individual models in this study. During the validation phase, the NN achieved perfect scores across all evaluation metrics—Accuracy, Precision, Recall, and F1-score attaining values of 1.0000. This indicates that the model was able to correctly classify every single malware and benign sample in the hold-out dataset, highlighting its remarkable generalization capability. Moreover, its performance during cross-validation was nearly as impressive, with an Accuracy_CV of 0.9965 and a Recall_CV of 0.9873, reinforcing the model's ability to consistently detect true positives across different data partitions. These results can be attributed to the inherent strengths of neural networks in automatically learning complex, non-linear, and hierarchical feature representations. Unlike traditional machine learning models that rely heavily on manually engineered features, neural networks can extract high-level abstractions from raw or preprocessed input data. This property is particularly advantageous in the context of metamorphic malware detection, where the malware's surface code structure is frequently altered through sophisticated techniques such as instruction substitution, dead code insertion, and register renaming. While such mutations may evade shallow detection models or signature-based systems, they often preserve subtle, latent patterns that neural networks are adept at capturing—especially when dealing with features derived from opcode sequences, disassembled instructions, and control flow information. The neural network's ability to effectively generalize also speaks to its resilience in handling feature redundancy and noise, common issues in malware datasets. While overfitting is a potential risk with deep models especially when training data is limited this was mitigated in the current implementation through techniques such as early stopping, dropout, and proper regularization, ensuring a balanced learning process. Furthermore, the relatively stable performance across all folds during cross-validation suggests that the model was not overly dependent on specific subsets of the data, but instead captured more generalizable decision patterns. However, it is important to note that neural networks typically come with increased computational cost and training time compared to simpler algorithms such as Decision Trees or KNN. In resource-constrained environments or real-time applications, this may pose a deployment challenge. Nonetheless, the trade-off is often justified, particularly in mission-critical applications like malware detection, where accuracy and the ability to adapt to evolving threats outweigh marginal differences in computational efficiency. In the context of ensemble learning, the Neural Network provides significant value due to its ability to capture deep, non-linear interactions among features that other base learners might overlook. When integrated with complementary models such as SVMs and Decision Trees, it contributes to a more diverse and holistic decision-making process, enhancing the overall robustness and adaptability of the ensemble framework. In conclusion, the Neural Network stands out as a highly capable and reliable model in this study, excelling in both predictive accuracy and generalization. Its inclusion in the ensemble is not only beneficial but essential to achieving the research objective of developing a resilient, intelligent, and high-performing malware detection system.

Collectively, the performance of the individual classifiers Decision Tree, Support Vector Machine, K-Nearest Neighbors, and Neural Network demonstrates a strong foundational capability for detecting metamorphic malware. Each model achieved high levels of accuracy, precision, recall, and F1-score, affirming the effectiveness of the selected features and preprocessing techniques in capturing meaningful patterns indicative of malicious behavior. These results validate the decision to include them as base learners within the proposed detection framework.

VIII. PERFORMANCE OF THE ENSEMBLED MODEL

While the specific performance metrics for the ensemble model (Bagging, Boosting, and Stacking configurations) are not yet available, the theoretical underpinnings and design choices outlined in Chapter 3 strongly suggest superior performance

compared to the individual base classifiers. The ensemble learning framework is explicitly designed to overcome the limitations of standalone models by reducing both variance and bias, thereby improving generalization and resilience against adversarial inputs.

Improved Accuracy and Robustness: By combining multiple diverse base learners, the ensemble model is expected to achieve higher overall accuracy and robustness. For instance, Bagging (e.g., Random Forest) is anticipated to reduce the variance observed in individual Decision Trees, leading to more stable and reliable predictions. Boosting techniques are expected to focus on and correct misclassifications from previous learners, thereby enhancing performance on challenging-to-detect metamorphic variants. Stacking, as the most sophisticated technique, is designed to optimally combine the complementary strengths of heterogeneous base models, leading to a more comprehensive and accurate final decision.

Enhanced Generalization: The diversity of base learners and the aggregation mechanisms of ensemble methods are expected to improve the model's ability to generalize across various malware families and unseen metamorphic variants. This is crucial given the continuous evolution and self-modifying nature of metamorphic malware.

Comparative Analysis

The core objective of this research is not only to develop a robust detection framework for metamorphic malware but also to empirically validate its superiority over individual classifiers and existing benchmark models. This section provides a structured comparative analysis, evaluating the proposed ensemble model across three primary dimensions:

- Performance relative to individual base classifiers
- Benchmarking against the prior work of Ahmed Ali (2020)
- Adaptability to metamorphic mutations
- Computational overhead efficiency

Comparison Against Individual Base Classifiers

As detailed in Section 4.2, the individual base classifiers Decision Tree (DT), Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Neural Network (NN) demonstrated notably high performance, with several achieving perfect scores on the validation set. However, despite their individual strengths, these models exhibit inherent weaknesses:

- DT may overfit without pruning
- SVM is sensitive to margin-bound data
- KNN struggles with high-dimensionality and noise
- NN may require large data volumes and regularization to avoid overfitting

The ensemble learning approach strategically addresses these issues by combining models with diverse inductive biases, enabling it to capitalize on the strengths of each while mitigating their weaknesses.

Table 4.2: Key Advantages of Ensemble over Base Classifiers

Limitation in Base Models	Ensemble Strategy to Address It	Benefit
Overfitting in DT or NN	Bagging (e.g., Random Forest)	Reduces variance through model averaging across bootstrapped datasets
Marginal sensitivity in SVM	Boosting (e.g., AdaBoost, Gradient Boosting)	Focuses on hard-to-classify instances, improving classification robustness
Lack of generalization in KNN	Stacking (with meta-learner)	Learns optimal combination of base outputs, improving overall prediction
Variance across data folds	Diversity of learners + Cross-validation	Increases consistency and resilience to changes in training distributions

In summary, while each base classifier performs well independently, the ensemble model offers a more balanced and generalizable solution, particularly in the dynamic landscape of malware evolution where threats continually mutate to evade single-model detection.

Computational Overhead Analysis

Table 4.3: Computational Trade-off Summary

Metric	Single SVM (Ali, 2020)	Deep NN	Proposed Ensemble
Accuracy	Moderate	High	Very High
Adaptability	Low to Moderate	High	High
Training Time	Low	High	Moderate (optimized via feature pruning)
Inference Time	Low	Moderate to High	Moderate
Memory Usage	Low	High	Moderate

Conclusion of Comparative Analysis

The comparative analysis clearly demonstrates that the proposed ensemble model delivers superior performance across multiple dimensions. It not only improves on the individual limitations of base classifiers but also outperforms benchmark models like that of Ahmed Ali (2020), particularly in terms of generalization, adaptability to code mutation, and robustness under real-world conditions. Furthermore, its optimized computational profile makes it a practical and scalable solution suitable for modern cybersecurity infrastructures where both accuracy and efficiency are paramount.

IX. DISCUSSION OF FINDINGS AND IMPLICATIONS

The empirical results obtained throughout this research provide compelling evidence in support of the proposed methodology and underline several key insights with broad implications for the field of malware detection and cybersecurity. The consistently high performance achieved by the individual base classifiers particularly the Decision Tree (DT), Support Vector Machine (SVM), and Neural Network (NN) on both the cross-validation and validation datasets demonstrates the effectiveness of the feature engineering approach adopted in this study. By incorporating a hybrid feature set comprising opcode sequences, assembly-level patterns, and PE header metadata, the study successfully extracted discriminative representations that allowed even standalone models to differentiate between benign and metamorphic malware samples with high accuracy. These findings confirm that traditional machine learning models, when trained on carefully curated and contextually rich features, can serve as powerful detection tools. The performance benchmarks achieved by the base classifiers establish a strong baseline and validate the relevance of non-deep learning techniques in real-time malware detection scenarios, particularly when computational efficiency is also a consideration. This is especially notable for models like DT and SVM, which are computationally lightweight yet performed at near-optimal levels on unseen data. However, a closer inspection reveals important nuances in generalization performance. While most classifiers performed exceptionally well on the hold-out validation set, variability in cross-validation metrics particularly in models like KNN highlighted inconsistencies in how well certain classifiers generalized across different data partitions.

In conclusion, the findings of this study not only validate the research objectives but also contribute meaningfully to the ongoing evolution of intelligent malware detection systems. By combining feature-rich inputs with strategically designed ensemble architectures, this work provides a blueprint for building more resilient, adaptable, and accurate malware detection frameworks, setting a solid foundation for future research and real-world application in cybersecurity defense.

X. CONCLUSION

This research successfully demonstrates that ensemble learning is a powerful and effective strategy for the detection of metamorphic malware. The study shows that while traditional machine learning classifiers can achieve high levels of accuracy with carefully engineered features, their standalone application is often hindered by issues such as overfitting, sensitivity to noise, and limited generalization to novel malware variants. The developed ensemble model strategically combines the strengths of diverse classifiers, achieving a balanced and comprehensive malware detection framework. The integration of diverse feature types and learning paradigms ranging from distance-based and margin-based classifiers to deep pattern learners results in a system that is not only more accurate but also more resilient to code mutations and evasive malware behaviors.

REFERENCES

1. Alam, S., Horspool, R. N., Traore, I., & Sogukpinar, I. (2015). A framework for metamorphic malware analysis and real-time detection ScienceDirect A framework for metamorphic malware analysis and real-time detection. Computers & Security, 48(March 2018), 212–233. https://doi.org/10.1016/j.cose.2014.10.011
2. Alenezi, M. N., Alabdulrazzaq, H., Alshaher, A. A., & Alkharang, M. M. (2021). Evolution of Malware Threats and Techniques : A Review. February. https://doi.org/10.17762/ijcnis.v12i3.4723
3. Ali, M., Shiaeles, S., & Bendiab, G. (2020). MALGRA : Machine Learning and N-Gram Malware Feature Extraction and Detection System. 1–21. https://doi.org/10.3390/electronics9111777
4. Bashari Rad, B., Masrom, M., & Ibrahim, S. (2012). Camouflage In Malware: From Encryption To Metamorphism. International Journal Of Computer Science And Network Security (IJCSNS), 12(8), 74–83. http://paper.ijcsns.org/07_book/201208/20120813.pdf
5. Directions, F., Avhankar, M. S., Pawar, J., & Kumbhar, V. (2025). A Comprehensive Survey on Polymorphic Malware Analysis : Challenges , A Comprehensive Survey on Polymorphic Malware Analysis : Challenges , Techniques , and Future Directions. March. https://doi.org/10.52783/cana.v32.4554
6. Doris, L., & Shad, R. (2024). USING MACHINE LEARNING MODELS TO IDENTIFY AND PREDICT SECURITY- RELATED ANOMALIES IN REAL-TIME FOR PROACTIVE MAINTENANCE. December.
7. Gasmi, S. (2024). Advanced Threat Detection with Machine Learning : A Holistic Framework for Advanced Threat Detection with Machine Learning : A Holistic Framework for Cybersecurity Date : November , 2024. November. https://doi.org/10.13140/RG.2.2.11687.36004
8. Habib, F., Shirazi, S. H., Aurangzeb, K., Khan, A., Bhushan, B., & Alhussein, M. (2024). Deep Neural Networks for Enhanced Security : Detecting Metamorphic Malware in IoT Devices. IEEE Access, PP, 1. https://doi.org/10.1109/ACCESS.2024.3383831
9. Hama Saeed, M. A. (2020). Malware in Computer Systems: Problems and Solutions. IJID (International Journal on Informatics for Development), 9(1), 1. https://doi.org/10.14421/ijid.2020.09101
10. Lange, A., Smolyakov, D., & Burnaev, E. (2023). Sequential Minimal Optimization algorithm for one-class Support Vector Machines with privileged information. IEEE Access, PP, 1. https://doi.org/10.1109/ACCESS.2023.3331685
11. Ling, Y., Fazlida, N., & Sani, M. (2017). International Journal of Advanced Research in Short Review on Metamorphic Malware Detection in Hidden Markov Models. June. https://doi.org/10.23956/ijarcsse/V7I2/01218
12. Mamoun, S., & Ahmed, A. (2025). Applying Ensemble Machine Learning Techniques to Malware Detection. 10.
13. Pagano, T. P., Loureiro, R. B., Lisboa, F. V. N., Peixoto, R. M., Guimarães, G. A. S., Cruz, G. O. R., Araujo, M. M., Santos, L. L., Cruz, M. A. S., Oliveira, E. L. S., Winkler, I., & Nascimento, E. G.

S. (2023). Bias and Unfairness in Machine Learning Models: A Systematic Review on Datasets , Tools , Fairness Metrics , and Identification and Mitigation Methods. 1–31.

14. Pava, R., & Mishra, S. (2024). Issues in Information Systems Analyzing machine learning algorithms for antivirus applications: a study on decision trees , support vector machines , and Analyzing machine learning algorithms for antivirus applications: a study on decision trees , suppo. October. https://doi.org/10.48009/4