A Multi-Stage Ensemble Learning System for Static Malware Detection Using the EMBER Dataset

Devata Anekar, Parth Dongre, Parth Birari, Atharv Patil, Aryan Patil

Department of Engineering, Sciences and Humanities (DESH)
Vishwakarma Institute of Technology, Pune, Maharashtra, India

Abstract *— Malware specimens have increasingly adopted sophisticated evasion mechanisms, including code obscuration, polymorphic behavior, and advanced compression techniques, thereby substantially undermining the efficacy of traditional signature-based detection methodologies. Contemporary cybersecurity frameworks necessitate intelligent, scalable, and dependable architectures capable of sustaining high detection precision while minimizing false positive rates in operationally realistic environments characterized by severe class imbalance. This study proposes a multi-stage ensemble learning framework for static malware detection and classification of Portable Executable files. The methodology incorporates comprehensive static feature extraction and integrates multiple machine learning algorithms to enhance robustness and generalization capability. The framework employs a structured two-stage detection pipeline: the initial stage discriminates between benign and malicious executables, while the subsequent stage performs granular malware family classification. A principal contribution of this research involves an adaptive decision mechanism that identifies predictions with insufficient confidence and selectively re-evaluates ambiguous samples, thereby enhancing classification reliability and reducing misclassification rates. The system undergoes training and evaluation using large-scale benchmark datasets, including EMBER 2018 and EMBER 2024, ensuring exposure to diverse and temporally evolving malware variants. Experimental validation is conducted under realistic conditions reflecting severe class imbalance, where benign files substantially outnumber malicious specimens. Results indicate that the proposed ensemble framework achieves superior accuracy, robustness, and scalability relative to individual classifier implementations. The presented approach offers a practical and extensible solution for static malware analysis, with potential future enhancements to accommodate additional file formats and expanded datasets.*

Keywords *— Class Imbalance, Cybersecurity, EMBER Dataset, Ensemble Learning, Feature Extraction, Machine Learning, Malware Classification, Malware Detection, Static Analysis, Threat Intelligence*

I. INTRODUCTION

THE WIDESPREAD ADOPTION OF DIGITAL TECHNOLOGIES AND UBIQUITOUS INTERNET ACCESS HAS FUNDAMENTALLY ALTERED HOW INDIVIDUALS, ORGANIZATIONS, AND GOVERNMENTS RELY ON SOFTWARE-BASED SYSTEMS FOR DAILY OPERATIONS. ACCOMPANYING THIS TECHNOLOGICAL GROWTH, MALWARE HAS BECOME A SIGNIFICANT CYBERSECURITY CONCERN, CONTINUOUSLY ADVANCING IN COMPLEXITY AND REACH. MODERN MALICIOUS SOFTWARE UTILIZES SOPHISTICATED EVASION METHODS, INCLUDING CODE OBSCURATION, MORPHOLOGICAL VARIATIONS, AND ADVANCED COMPRESSION TECHNIQUES, WHICH EFFECTIVELY BYPASS TRADITIONAL SIGNATURE-BASED ANTIVIRUS SYSTEMS. THIS EVOLVING THREAT LANDSCAPE REQUIRES ADAPTIVE AND INTELLIGENT DETECTION APPROACHES CAPABLE OF ADDRESSING CONTEMPORARY ATTACK METHODS.

MACHINE LEARNING TECHNIQUES HAVE RECEIVED CONSIDERABLE ATTENTION IN MALWARE DETECTION RESEARCH DUE TO THEIR ABILITY TO IDENTIFY COMPLEX PATTERNS WITHIN LARGE DATASETS AND RESPOND TO PREVIOUSLY UNKNOWN THREATS. STATIC MALWARE ANALYSIS OFFERS SEVERAL PRACTICAL ADVANTAGES, INCLUDING FASTER PROCESSING TIMES, LOWER COMPUTATIONAL DEMANDS, AND IMPROVED RESISTANCE TO SANDBOX EVASION COMPARED TO DYNAMIC ANALYSIS APPROACHES. HOWEVER, DETECTION SYSTEMS THAT RELY ON SINGLE MACHINE LEARNING MODELS OFTEN EXHIBIT LIMITED GENERALIZATION CAPABILITIES, VULNERABILITY TO CLASS IMBALANCE ISSUES, AND UNRELIABLE PERFORMANCE WHEN PREDICTION CONFIDENCE IS LOW.

TO ADDRESS THESE CHALLENGES, ENSEMBLE LEARNING METHODS HAVE BECOME INCREASINGLY PROMINENT IN MALWARE DETECTION RESEARCH. BY COMBINING MULTIPLE CLASSIFIERS, ENSEMBLE-BASED SYSTEMS CAN IMPROVE DETECTION CONSISTENCY, REDUCE FALSE POSITIVE RATES, AND ACHIEVE BETTER RESILIENCE AGAINST DIVERSE MALWARE BEHAVIORS. THESE CHARACTERISTICS ARE ESPECIALLY IMPORTANT IN PRACTICAL APPLICATIONS WHERE MALWARE DATASETS DEMONSTRATE SIGNIFICANT CLASS IMBALANCE, WITH BENIGN FILES SUBSTANTIALLY OUTNUMBERING MALICIOUS SAMPLES.

THIS RESEARCH PRESENTS A HIERARCHICAL ENSEMBLE LEARNING FRAMEWORK FOR STATIC MALWARE DETECTION AND CLASSIFICATION. THE SYSTEM EXTRACTS COMPREHENSIVE STATIC FEATURES FROM EXECUTABLE FILES AND EMPLOYS A STRUCTURED ANALYTICAL PROCESS THAT FIRST IDENTIFIES MALICIOUS INTENT, THEN CATEGORIZES DETECTED MALWARE INTO SPECIFIC FAMILIES. AN ADAPTIVE DECISION PROTOCOL ADDRESSES UNCERTAIN PREDICTIONS BY SELECTIVELY REASSESSING SAMPLES WITH LOW CONFIDENCE SCORES WITHIN DEFINED THRESHOLD RANGES. THE FRAMEWORK IS TRAINED AND VALIDATED USING LARGE-SCALE BENCHMARK DATASETS, INCLUDING EMBER 2018 AND EMBER 2024, ENSURING COMPREHENSIVE COVERAGE OF DIVERSE AND EVOLVING MALWARE VARIANTS. OVERALL, THIS APPROACH PROVIDES A SCALABLE, RELIABLE, AND PRACTICAL SOLUTION FOR MODERN STATIC MALWARE ANALYSIS.

1

II. LITERATURE REVIEW

Anderson and Roth [1] introduced the EMBER dataset, a large-scale benchmark for static malware detection using PE files with standardized feature representations including PE headers, section statistics, and entropy measures. This dataset has become widely accepted due to its reproducibility and scalability for malware classification research.

Anderson et al. [2] released EMBER 2024, significantly expanding the original dataset with newer malware families, evasive samples, and temporal variations reflecting real-world concept drift. This addresses model degradation concerns by exposing systems to samples across different time periods and distributions.

Saxe and Berlin [3] demonstrated that machine learning models trained on static PE features achieve strong detection performance without requiring dynamic execution. Their work established the viability of scalable, efficient static analysis resistant to sandbox evasion.

Ye et al. [4] emphasized that feature quality plays a more critical role than classifier choice in malware detection tasks. Their findings justify reliance on high-quality static feature representations over complex classifier architectures.

Baldangombo et al. [5] empirically showed that ensemble-based malware detection systems outperform single classifiers, particularly under noisy and imbalanced conditions. This validates the effectiveness of combining multiple classifiers with diverse learning biases.

Gibert et al. [7] further demonstrated ensemble superiority in malware detection through empirical validation. Their work reinforced that classifier combination improves robustness and generalization in adversarial environments.

Dietterich [8] provided theoretical foundations for ensemble methods including bagging and boosting, demonstrating that ensembles reduce variance and improve predictive stability. This work established fundamental principles for combining multiple models effectively.

Zhou [9] emphasized that classifier diversity is a key factor in achieving superior ensemble performance. The work highlighted that heterogeneous model combinations outperform homogeneous ensembles.

Kantchelian et al. [10] demonstrated that attackers can exploit classifier uncertainty near decision boundaries, motivating confidence-aware decision mechanisms. Their findings emphasized the vulnerability of machine learning detectors under adversarial conditions.

Guo et al. [11] showed that well-calibrated probability estimates are essential for reliable decision-making in classification systems. Their work established methods for assessing and improving model calibration.

Niculescu-Mizil and Caruana [12] provided additional evidence that probability calibration significantly impacts classifier reliability and trustworthiness. They demonstrated techniques for producing better-calibrated predictions across different model types.

Hou et al. [13] demonstrated that selecting relevant static features reduces computational overhead while maintaining high detection accuracy. Their work justified adaptive feature selection strategies across different classification stages.

Schultz et al. [14] laid foundational work in extracting static features such as PE headers, imported functions, and byte sequences. These feature engineering practices continue to influence modern malware detection systems.

Santos et al. [15] showed that static representations can capture meaningful malicious patterns while remaining computationally efficient. Their work validated the effectiveness of static analysis over resource-intensive dynamic approaches.

Chen and Guestrin [16] introduced XGBoost, a scalable gradient boosting framework demonstrating strong performance across classification tasks including malware detection. The framework's ability to capture complex non-linear feature interactions makes it particularly effective for security applications.

Ke et al. [17] proposed LightGBM, which optimizes gradient boosting through histogram-based learning for large-scale datasets. This algorithm provides both scalability and efficiency advantages for processing datasets like EMBER 2024.

Breiman [18] introduced Random Forests, which have been widely applied in malware detection due to their robustness and resistance to overfitting. This foundational work established tree-based ensembles as dominant approaches in security classification tasks.

Shafiq et al. [19] highlighted that evaluation under artificially balanced datasets can produce misleading performance estimates in malware detection. Their work emphasized the importance of realistic data distributions in security evaluations.

Chawla et al. [20] discussed sampling strategies for imbalanced learning, including SMOTE and other oversampling techniques. However, they noted that such techniques may not always be suitable for security-critical applications.

He and Garcia [21] provided comprehensive evaluation strategies for learning from imbalanced data distributions.

Their work established best practices for handling class imbalance in classification problems.

Moskovitch et al. [22] demonstrated that separating malware detection from family classification reduces error propagation and improves overall accuracy. Their hierarchical approach motivated multi-stage detection pipelines.

Cabrera et al. [23] showed that hierarchical classification frameworks outperform flat classification schemes when dealing with large malware taxonomies. This validates progressive categorization strategies from coarse to fine-grained labels.

Eskandari et al. [24] proposed adaptive detection mechanisms that re-evaluate uncertain predictions to improve robustness against adversarial manipulation. Their confidence-aware approach enables more reliable decision-making under uncertainty.

Raff et al. [25] demonstrated that malware detection models trained on historical data often degrade over time due to evolving threat characteristics. This work highlighted the critical importance of temporal validation in security systems.

Wojnowicz et al. [26] further validated that concept drift significantly impacts malware detector performance over time. Their findings emphasized the need for datasets spanning multiple time periods to ensure robust evaluation.

The reviewed literature establishes that static malware detection utilizing high-quality feature representations proves both effective and scalable, particularly when integrated with ensemble learning methodologies. Prior research has validated the benefits of large-scale benchmark datasets such as EMBER [1], [2], the superiority of ensemble classifiers over individual models [5], [7], and the significance of classifier diversity [9], probability calibration [11], [12], and realistic evaluation under class imbalance conditions [19], [21]. Furthermore, hierarchical classification architectures [22], [23] and confidence-aware decision protocols [10], [24] have demonstrated improved robustness and reduced error propagation across complex malware taxonomies. Gradient boosting frameworks [16], [17] and tree-based ensembles [18] have proven particularly effective for capturing intricate patterns from static features [3], [15], while feature selection strategies enhance computational efficiency [13], [14].

However, despite these advances, existing approaches frequently address these components in isolation. Limited research integrates ensemble learning, hierarchical detection, confidence-aware re-evaluation, and temporal robustness within a unified framework evaluated on evolving, large-scale datasets. Furthermore, insufficient attention has been directed toward systematically handling low-confidence predictions under realistic class imbalance while maintaining scalability across emerging malware distributions and addressing temporal degradation [25], [26]. These limitations highlight the necessity for a comprehensive, multi-stage ensemble framework that combines robust static feature extraction, adaptive decision-making mechanisms, hierarchical classification, and evaluation across temporally diverse datasets. Addressing these gaps is crucial for developing practical and resilient static malware detection systems capable of performing effectively in operational environments.

III. Methodology/Experimental

A. System Architecture and Block Diagram

The proposed system is a multi-stage hierarchical static malware detection and classification framework designed for Windows Portable Executable (PE) files. The architecture follows a progressive decision pipeline in which samples advance to deeper classification stages only when necessary, thereby improving both efficiency and robustness.

Static features are extracted from input executables using the EMBER feature schema. These features include PE header metadata, section statistics, imported functions, byte histograms, and entropy-based measurements. The resulting feature vector serves as a unified representation across all classification stages.

The first stage performs binary malware detection to determine whether an input sample is benign or malicious. This stage employs a heterogeneous ensemble composed of ExtraTrees, XGBoost, and LightGBM classifiers trained on a combined EMBER 2018 and EMBER 2024 dataset. A curated subset of 534 static features is selected to balance detection performance and computational efficiency. The ensemble produces a calibrated probability score representing the likelihood of malicious behavior.

Samples classified as benign are discarded, while those identified as malicious proceed to the umbrella classification stage, where they are grouped into high-level malware categories such as Trojan, Stealer, Ransomware, Miner, Backdoor, Virus, and Worm. This step reduces label complexity and constrains the downstream classification space.

For samples assigned to the Trojan umbrella, an additional sub-umbrella classification stage is applied, further categorizing them into subclasses including loader, banker, spyware, remote access trojans (RATs), and generic Trojan variants.

The final stage performs malware family classification using a LightGBM-based multi-class classifier trained on 75 malware families with sufficient sample representation. This hierarchical design minimizes error propagation while enhancing scalability and interpretability. Fig.

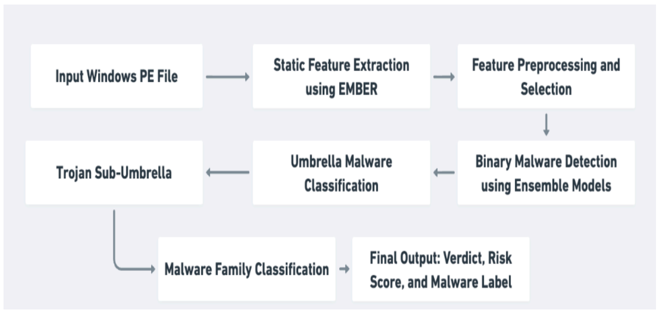

Input Windows PE File
Static Feature Extraction using EMBER
Feature Preprocessing and Selection
Binary Malware Detection using Ensemble Models
Umbrella Malware Classification
Trojan Sub-Umbrella
Malware Family Classification
Final Output: Verdict, Risk Score, and Malware Label

Fig 1. Block diagram of the proposed multi-stage ensemble malware detection framework.

B. Algorithm and Design Methodology

The core design of the proposed framework is based on ensemble learning, hierarchical decision-making, and confidence-aware prediction. The binary detection stage employs a confidence-weighted ensemble strategy to combine predictions from multiple classifiers.

Each ensemble member independently produces a probability estimate indicating the likelihood of malicious behavior. These outputs are dynamically normalized to compute per-sample weights, enabling classifiers with higher confidence to contribute more strongly to the final decision. The weighted predictions are then aggregated to obtain a unified ensemble probability score.

To improve reliability near decision boundaries, a post-hoc probability calibration mechanism is applied to the ensemble output. This calibration reduces mid-confidence false positives while preserving ranking consistency for high-confidence malware samples. Based on the calibrated probability, samples are assigned to predefined risk categories, including Likely Benign, Low-Confidence Suspicious, Suspicious Win32, and High-Risk Malware.

The umbrella and sub-umbrella classification stages integrate rule-guided taxonomy with machine learning. Keyword-based mappings derived from established malware naming conventions are used to assign samples to high-level umbrella categories, ensuring alignment with real-world antivirus taxonomies. Within each constrained label space, dedicated classifiers are employed to improve precision and reduce misclassification.

For malware family classification, a LightGBM-based multi-class model is trained using a minimum family size threshold to mitigate class imbalance and ensure stable learning, while maintaining high classification accuracy.

C. Algorithm Description and Testing Strategy

To ensure consistency with the proposed system architecture, the inference workflow is defined as a hybrid machine learning and heuristic pipeline. The system integrates ensemble-based classification with lightweight Go-based static heuristics to improve robustness, interpretability, and false-positive control. Algorithm 1 describes the complete end-to-end malware detection and classification process.

Algorithm 1: Hierarchical Malware Detection and Classification Pipeline with Go Heuristics

Input: Windows Portable Executable (PE) file
Output: Final verdict, calibrated risk score, and malware classification (if malicious)

Step 1: Static Feature Extraction
Static features are extracted from the input PE file using the EMBER feature schema. These features include PE header metadata, section statistics, imported functions, byte histograms, and entropy-based measurements. A unified feature vector consisting of 534 features is used across all machine learning stages.

Step 2: Binary Malware Detection (Ensemble Stage)
The trained binary classifiers—ExtraTrees, XGBoost, and LightGBM—are loaded. Each classifier independently computes a malware probability score:

- p_1: LightGBM prediction
- p_2: XGBoost prediction
- p_3: ExtraTrees prediction

Step 3: Confidence-Weighted Ensemble Aggregation
The individual probability scores are normalized to compute dynamic, per-sample weights. The ensemble malware score is calculated as a weighted aggregation of the individual classifier outputs.

Step 4: Risk Calibration and Preliminary Decision
A post-hoc calibration mechanism is applied to the ensemble score to suppress mid-confidence false positives. Based on the calibrated score, the sample is assigned to one of the predefined risk bands: Likely Benign, Low-Confidence Suspicious, Suspicious Win32, or High-Risk Malware. Samples classified as Likely Benign terminate the analysis and return the verdict.

Step 5: Go-Based Static Heuristic Validation
For samples classified as malicious or suspicious, a lightweight static heuristic engine implemented in Go is applied. The heuristic module evaluates rule-based indicators such as:

- Abnormal PE section counts and sizes
- Suspicious section entropy patterns
- Known malicious import combinations
- Invalid or uncommon header field values
- Go-optimized signature and rule checks

The heuristic output is used to adjust the calibrated risk score and flag samples for increased scrutiny without overriding high-confidence machine learning decisions.

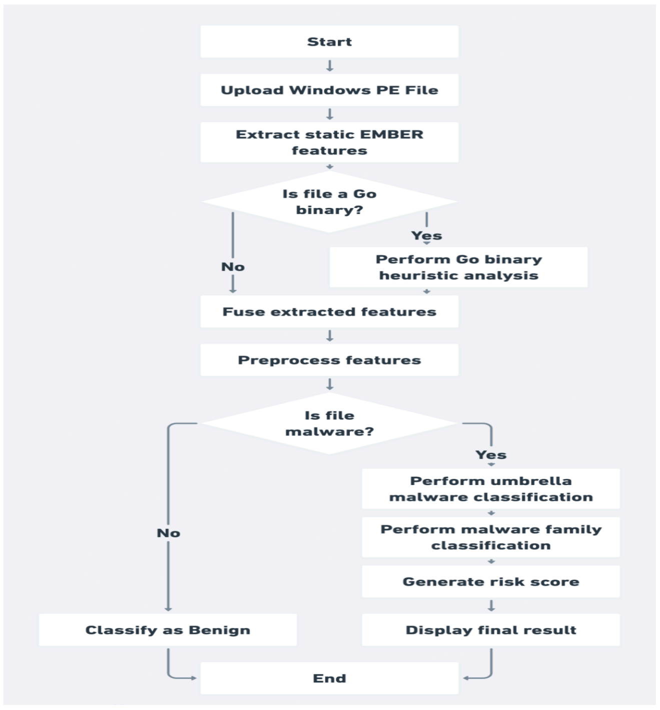
Step 6: Confidence Rating Assignment
A confidence rating is derived based on the calibrated ensemble score, heuristic indicators, and inter-model agreement. Confidence levels include High Confidence, Medium Confidence, and Low Confidence, and are used for reporting and analyst prioritization rather than decision override.

Step 7: Umbrella Classification
The umbrella classifier assigns a high-level malware category, including Trojan, Stealer, Ransomware, Miner, Backdoor, Virus, or Worm.

Step 8: Trojan Sub-Umbrella Classification (Conditional)
If the assigned umbrella category is Trojan, a sub-umbrella classifier further categorizes the sample into Loader or Generic Trojan variants.

Step 9: Malware Family Classification
The appropriate family classification model is selected based on the assigned umbrella and sub-umbrella labels. Final malware family prediction is performed using a LightGBM-based multi-class classifier trained on 75 malware families with sufficient sample representation.

Step 10: Output Generation
The system returns the final verdict, calibrated risk score, umbrella category, optional Trojan sub-category, malware family label, and heuristic flags, if applicable.



Testing Strategy

The proposed framework is evaluated using progressive and large-scale experimental setups to assess detection accuracy, scalability, and robustness. Initial benchmarking is performed on 10% and 20% subsets of the combined EMBER 2018 and EMBER 2024 datasets to compare candidate models and identify optimal ensemble components.

Final evaluation is conducted on the complete dataset comprising over 3.1 million samples. Binary malware detection and umbrella classification performance are assessed using standard metrics, including accuracy, precision, recall, F1-score, and ROC-AUC. Malware family classification performance is evaluated using both macro and weighted F1-scores to account for significant inter-family class imbalance.

All experiments are carried out under realistic data distributions, emphasizing temporal generalization, computational efficiency, and robustness against evolving malware threats.

IV. Results and Discussions

A. Experimental Setup and Evaluation Metrics

The performance of the proposed framework is evaluated on a combined dataset constructed from EMBER 2018 and EMBER 2024, comprising over 3.1 million Windows Portable Executable (PE) samples with realistic benign-to-malware class distributions. Experiments are conducted in a staged manner, beginning with partial dataset subsets (10% and 20%) for initial model benchmarking and ensemble component selection, followed by full-scale evaluation on the complete dataset.

For binary malware detection and umbrella classification tasks, performance is assessed using standard evaluation metrics, including accuracy, precision, recall, F1-score, and the area under the receiver operating characteristic curve (ROC-AUC), enabling balanced analysis under severe class imbalance. For malware family classification, both macro F1-score and weighted F1-score are reported to capture per-family classification effectiveness while accounting for dominance effects introduced by large malware families.

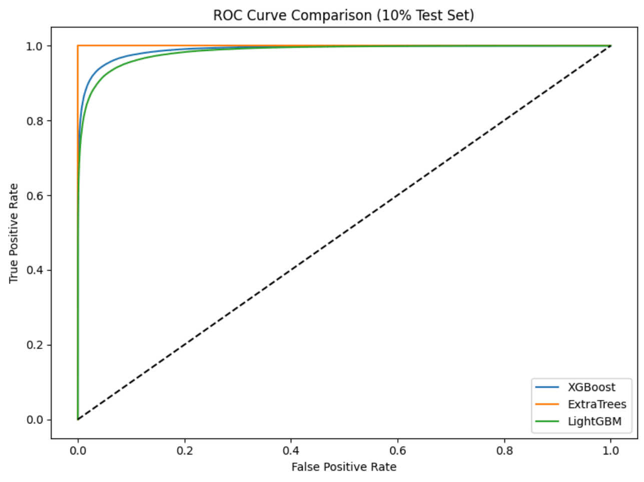


Fig. 2. ROC curve comparison for binary malware detection using XGBoost, ExtraTrees, and LightGBM.

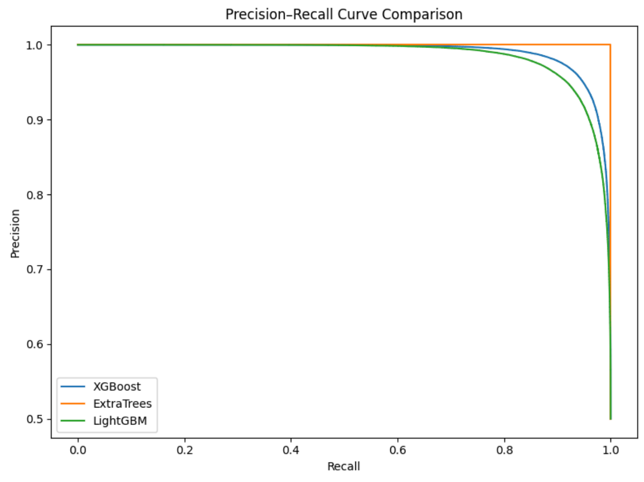


Fig. 3. Precision–Recall curve comparison highlighting classifier behavior under class imbalance.

B. Binary Malware Detection Performance

The binary malware detection stage demonstrates strong and stable performance across all experimental settings. Among individual classifiers, tree-based models consistently outperform simpler learners, with ExtraTrees achieving the highest standalone ROC-AUC, followed closely by XGBoost and LightGBM. This trend highlights the effectiveness of ensemble-based tree methods in modeling complex static feature interactions present in malware data.

The heterogeneous ensemble surpasses all individual models across accuracy, F1-score, and ROC-AUC metrics. This performance gain can be attributed to classifier diversity, where ExtraTrees contribute variance reduction through randomized feature selection, XGBoost effectively captures complex non-linear relationships, and LightGBM provides efficient gradient-based optimization on large-scale data. As a result, the ensemble exhibits enhanced robustness near decision boundaries, particularly under realistic class imbalance conditions.

Furthermore, the application of post-hoc risk calibration leads to a noticeable reduction in mid-confidence false positives without degrading high-confidence malware detection performance. This observation reinforces the importance of confidence-aware decision mechanisms in practical malware detection systems, where minimizing false alarms is critical for real-world deployment.

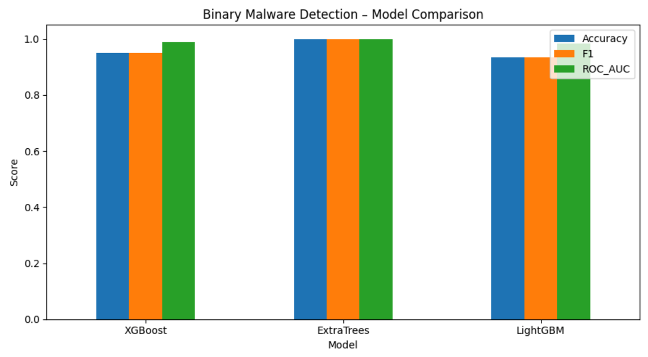


Fig. 4. Comparison of Accuracy, F1-score, and ROC-AUC for binary malware detection.

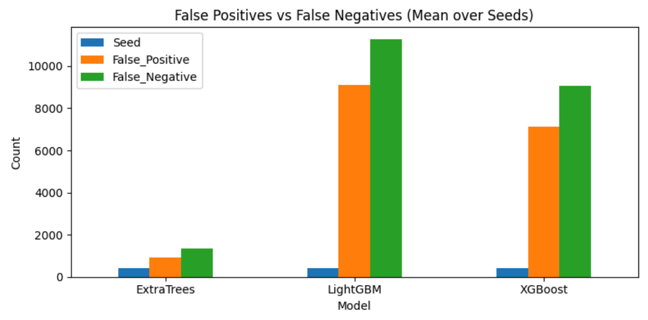


Fig. 5. Mean false positives and false negatives across multiple random seeds

C. Umbrella and Sub-Umbrella Classification Results

Umbrella classification results indicate that grouping malware into high-level semantic categories significantly improves downstream classification stability. Categories such as Trojan, Stealer, and Ransomware achieve consistently higher precision, which can be attributed to clearer separation in static feature representations. In contrast, smaller or semantically overlapping categories, including Virus and Worm, exhibit comparatively lower recall due to shared structural characteristics.

The Trojan sub-umbrella classification stage further refines detection by isolating structurally distinct Trojan subclasses.

Subcategories such as loaders and remote access trojans (RATs) demonstrate strong separability, while generic Trojan variants remain more challenging due to increased feature overlap. Despite these challenges, the hierarchical design effectively limits error propagation by constraining the label space at earlier stages, thereby reducing misclassification at the malware family level.

Overall, these results support the effectiveness of hierarchical classification strategies over flat multi-class approaches, particularly when applied to large and heterogeneous malware taxonomies.

Symbol	Precision	Recall	F1-Score
generic	0.9604	0.9796	0.9699
loader	0.9522	0.9096	0.9304
accuracy			0.9500
Macro avg	0.9563	0.9446	0.9501
Weighted avg	0.9579	0.9580	0.9577

TABLE I - Trojan Sub-Umbrella Classification Performance

D. Malware Family Classification Performance

The malware family classification stage demonstrates reliable performance across 75 malware families that satisfy the minimum sample size threshold. Dominant families achieve high precision and recall, while smaller families benefit from reduced label competition due to the hierarchical filtering applied in earlier classification stages. This progressive narrowing of the label space improves stability and reduces confusion among structurally similar families.

As expected, macro F1-scores are slightly lower than weighted F1-scores, reflecting the inherent imbalance in malware family distributions. This behavior is consistent with real-world malware datasets, where a small number of families account for a large proportion of samples. Importantly, the proposed framework avoids aggressive oversampling techniques, prioritizing generalization and predictive stability over artificially inflated performance metrics.

The use of LightGBM proves effective for large-scale multi-class classification, offering a favorable balance between training efficiency and predictive accuracy. Its ability to handle high-dimensional feature spaces and skewed class distributions makes it well suited for large malware taxonomies.

Table II summarizes the malware family classification performance across 75 families, reporting accuracy, macro-averaged metrics to capture per-family behavior, and weighted metrics to reflect overall system performance.

Metric	Value
Number of Families	75
Total Samples	736,508
Accuracy	0.9671
Macro Precision	0.9664
Macro Recall	0.9663
Macro F1-score	0.9662
Weighted Precision	0.9670
Weighted Recall	0.9671
Weighted F1-score	0.9670

TABLE II - Malware Family Classification Performance Across 75 Families

E. Scalability and Robustness Analysis

The proposed framework demonstrates strong scalability when transitioning from partial dataset experiments to full-scale evaluation. Training time increases approximately linearly with dataset size, while inference remains computationally efficient due to early termination of benign samples at the binary detection stage. This design significantly reduces unnecessary computation in downstream classification stages.

The combined use of EMBER 2018 and EMBER 2024 improves robustness against temporal concept drift. Models trained on the temporally diverse combined dataset exhibit improved generalization compared to those trained on a single temporal snapshot, highlighting the importance of time-aware training for long-term malware detection reliability.

Additionally, the hierarchical architecture limits error propagation by constraining predictions at each stage, ensuring that misclassifications do not disproportionately affect downstream classification outcomes. This contributes to improved stability and reliability in large-scale deployment scenarios.

F. Discussion and Key Observations

The experimental results demonstrate that the proposed framework effectively balances accuracy, scalability, and robustness through its multi-stage ensemble-based design. The system benefits from the integration of (i) high-quality static feature representations, (ii) heterogeneous ensemble learning, (iii) confidence-aware probability calibration, and (iv) hierarchical classification. Together, these components enable reliable malware detection and fine-grained classification under realistic, large-scale, and imbalanced data conditions.

Compared to flat classification pipelines, the hierarchical approach provides improved stability, reduced false-positive rates, and enhanced interpretability by progressively constraining the decision space. The use of ensemble learning further improves robustness near decision boundaries, while confidence-aware calibration enhances reliability in practical deployment scenarios. Although the framework is limited to static analysis, its scalability, computational efficiency, and resistance to sandbox evasion make it well suited for large-scale real-world malware detection systems.

Despite these strengths, certain limitations remain. The framework relies exclusively on static feature analysis and may not detect malware whose malicious behavior is observable only at runtime. Additionally, heavily obfuscated or encrypted binaries can reduce feature visibility, potentially impacting classification performance. These limitations motivate future work focused on integrating dynamic and hybrid analysis techniques, as well as extending the framework to support additional file formats and execution-aware features.

V. Future Scope

While the proposed framework demonstrates strong performance in static malware detection, several extensions can further enhance its capabilities. First, integrating dynamic and behavioral features alongside static analysis can improve the detection of heavily obfuscated or packed malware that evades purely static signatures. Such hybrid approaches would enable deeper visibility into runtime behaviors that are not observable through static features alone.

Second, incorporating continual and incremental learning mechanisms would allow the system to adapt to emerging malware families and evolving threat patterns without requiring complete retraining. This capability is particularly important for mitigating long-term concept drift and maintaining detection effectiveness in rapidly changing threat landscapes.

From a deployment perspective, future work may focus on optimizing real-time inference pipelines through model compression, pruning, and hardware-aware acceleration techniques to support high-throughput environments such as enterprise gateways and cloud-based malware scanning services. Additionally, extending the framework to support cross-platform malware analysis beyond Windows Portable Executable files represents a promising direction for broadening applicability and real-world impact.

VI. Conclusion

This paper presented a multi-stage ensemble-based static malware detection and classification framework designed for large-scale analysis of Windows Portable Executable files. Unlike flat classification pipelines, the proposed approach integrates hierarchical decision-making with confidence-aware ensemble learning to enhance robustness, scalability, and interpretability.

Experimental evaluation on a combined EMBER 2018 and EMBER 2024 dataset comprising over 3.1 million samples demonstrates that the framework achieves high detection accuracy and strong generalization under realistic class imbalance conditions. The heterogeneous ensemble consistently outperforms individual classifiers in binary malware detection, while the hierarchical umbrella, sub-umbrella, and family-level classification stages effectively reduce error propagation and improve downstream precision.

Overall, the results confirm that static feature–based analysis, when combined with ensemble learning and hierarchical taxonomy, remains a practical and effective approach for modern malware detection. The proposed framework is particularly well suited for large-scale deployment scenarios where computational efficiency, reliability, and resistance to sandbox evasion are critical requirements.

VII . Acknowledgment

The authors would like to thank their project guide and faculty mentors for their valuable guidance and technical feedback throughout the development of this work. The authors also acknowledge the use of publicly available datasets and open-source machine learning frameworks that made this research possible.

References

[1] H. S. Anderson and P. Roth, "EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models," *arXiv preprint arXiv:1804.04637*, 2018.

[2] H. S. Anderson, A. Kharkar, B. Filar, and D. Evans, "EMBER 2024: A Modern Benchmark for Static Malware Detection," *arXiv preprint*, 2024.

[3] J. Saxe and K. Berlin, "Deep Neural Network Based Malware Detection Using Two-Dimensional Binary Program Features," in *Proc. 10th Int. Conf. on Malicious and Unwanted Software (MALWARE)*, Oct. 2015, pp. 11–20.

[4] Y. Ye, T. Li, D. Adjeroh, and S. Iyengar, "A Survey on Malware Detection Using Data Mining Techniques," *ACM Computing Surveys*, vol. 50, no. 3, pp. 1–40, 2017.

[5] F. Baldangombo, M. H. D. C. Nascimento, and A. Zúquete, "An Ensemble-Based Approach for Malware Detection," in *Proc. IEEE Symp. on Computers and Communications*, 2018, pp. 1–6.

[6] A. Kantchelian, J. D. Tygar, and A. Joseph, "Evasion and Hardening of Tree Ensemble Classifiers," in *Proc.*

34th Int. Conf. on Machine Learning (ICML), 2017, pp. 2387–2396.

[7] D. Gibert, C. Mateu, and J. Planes, "The Rise of Machine Learning for Detection and Classification of Malware: Research Developments, Trends and Challenges," *Journal of Network and Computer Applications*, vol. 153, pp. 102526, 2020.

[8] [8] T. G. Dietterich, "Ensemble Methods in Machine Learning," in *Proc. Int. Workshop on Multiple Classifier Systems*, 2000, pp. 1–15.

[9] Z.-H. Zhou, *Ensemble Methods: Foundations and Algorithms*. Boca Raton, FL, USA: CRC Press, 2012.

[10] A. Kantchelian *et al.*, "Adversarial Machine Learning for Malware Detection," *IEEE Security & Privacy*, vol. 17, no. 2, pp. 62–69, 2019.

[11] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On Calibration of Modern Neural Networks," in *Proc. 34th Int. Conf. on Machine Learning (ICML)*, 2017, pp. 1321–1330.

[12] A. Niculescu-Mizil and R. Caruana, "Predicting Good Probabilities with Supervised Learning," in *Proc. 22nd Int. Conf. on Machine Learning (ICML)*, 2005, pp. 625–632.

[13] S. Hou, Y. Ye, Y. Song, and M. Abdulhayoglu, "Hindering Evasion Attacks Against Malware Detection," *IEEE Trans. on Information Forensics and Security*, vol. 11, no. 5, pp. 1005–1018, 2016.

[14] M. G. Schultz, E. Eskin, F. Zadok, and S. J. Stolfo, "Data Mining Methods for Detection of New Malicious Executables," in *Proc. IEEE Symp. on Security and Privacy*, 2001, pp. 38–49.

[15] I. Santos, F. Brezo, J. Nieves, Y. K. Penya, B. Sanz, C. Laorden, and P. G. Bringas, "Idea: Opcode-Sequence-Based Malware Detection," in *Proc. Int. Symp. on Engineering Secure Software and Systems*, 2010, pp. 35–43.

[16] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proc. 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, 2016, pp. 785–794.

[17] G. Ke *et al.*, "LightGBM: A Highly Efficient Gradient Boosting Decision Tree," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.

[18] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[19] M. Z. Shafiq, S. M. Tabish, F. Mirza, and M. Farooq, "PeMiner: Mining Structural Information to Detect Malicious Executables in Real Time," in *Proc. Int. Symp. on Recent Advances in Intrusion Detection*, 2009, pp. 121–141.

[20] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-Sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[21] H. He and E. A. Garcia, "Learning from Imbalanced Data," *IEEE Trans. on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[22] R. Moskovitch, Y. Elovici, and A. Shabtai, "Hierarchical Malware Classification," *Journal in Computer Virology*, vol. 4, no. 2, pp. 71–82, 2008.

[23] J. B. D. Cabrera, L. Lewis, and R. K. Mehra, "Detection and Classification of Intrusions Using Hierarchical Models," *Computer Networks*, vol. 34, no. 4, pp. 579–596, 2000.

[24] S. Eskandari, M. Hashemi, and M. Khazaei, "Adaptive Malware Detection Based on Uncertainty-Aware Learning," *Computers & Security*, vol. 92, 2020.

[25] A. Raff *et al.*, "On the Limitations of Static Malware Classifiers," in *Proc. 17th Int. Symp. on Research in Attacks, Intrusions and Defenses (RAID)*, 2017, pp. 1–21.

[26] M. Wojnowicz, G. Choudhary, and S. Wolff, "Data Drift in Malware Detection," *IEEE Security & Privacy Workshops*, 2016, pp. 1–6.