International Journal of Applied Information Systems and Cybersecurity
Vol. 1 No. 1 (2026) pp. 82-103
Online ISSN: XXXX-XXXX

Original Research

Behavior-Aware Static Malware Detection: Benhmarking Classical and Modern Machine Learning Models on the EMBER Dataset

Intan Permata Sari[1]✉, Rita Hamdani[1]
[1] STMIK Eresha, Indonesia
✉ intanpermata.sari@gmail.com

10.65840/ijaisc.vxix.xx

Article Info
Submitted:
10/19/2025
Revised:
11/15/2025
Accepted:
12/25/2025
Available Online:
01/15/2026

Abstract. *Malware attacks continue to evolve rapidly in sophistication, scale, and obfuscation capability, posing significant challenges to traditional signature-based cybersecurity systems. The growing diversity of malicious software behaviors has increased the demand for scalable and intelligent malware detection frameworks capable of maintaining strong predictive performance while remaining interpretable and computationally practical. Despite substantial advances in machine learning-based malware detection, many existing studies primarily emphasize classification accuracy while providing limited discussion regarding explainability, computational tradeoffs, and fair benchmarking between classical machine learning approaches and modern deep learning architectures. In particular, the effectiveness of lightweight behavior-aware feature engineering within large-scale static malware analysis remains insufficiently explored. This study presents a comprehensive benchmarking framework for static malware detection using the EMBER dataset. The proposed framework evaluates several classical and modern learning approaches, including XGBoost, Random Forest, Multi-Layer Perceptron (MLP), and TabNet, while incorporating behavior-aware import-based features extracted from Portable Executable (PE) files. In addition, SHAP explainability analysis is employed to investigate feature contribution and model decision behavior. Experimental results demonstrate that optimized classical ensemble-based approaches consistently outperform several modern deep learning models across multiple evaluation metrics. The tuned XGBoost configuration achieved the best overall performance, reaching 94.43% classification accuracy and a ROC-AUC score of 0.9871. SHAP analysis further revealed that entropy characteristics, import statistics, suspicious API indicators, and string-related features were among the most influential contributors to malware classification decisions. This work contributes a large-scale benchmarking framework integrating behavior-aware feature engineering, explainable AI analysis, and computational tradeoff evaluation for practical malware detection systems. Future research may extend this framework through dynamic malware analysis, transformer-based architectures, and adversarially robust detection strategies.*

Keywords: *static malware detection; behavior-aware feature engineering; EMBER dataset; explainable machine learning; XGBoost; SHAP analysis; cybersecurity benchmarking; portable executable analysis*

1. Introduction

The rapid evolution of malware has created significant challenges for conventional cybersecurity systems. Modern malware increasingly employs sophisticated evasion techniques, including obfuscation, polymorphism, packing, and adversarial manipulation, which reduce the effectiveness of traditional signature-based detection methods (Balasubramanian et al., 2023). As malware variants continue to grow in volume and complexity, cybersecurity solutions require detection mechanisms that are not only accurate but also scalable and capable of adapting to previously unseen threats.

To address these challenges, static malware detection has emerged as one of the most practical approaches for large-scale cybersecurity applications. Unlike dynamic analysis, which relies on runtime execution and sandbox environments, static analysis examines executable files without execution, enabling faster, lightweight, and more scalable malware inspection (Ucci et al., 2019a). Static detection commonly utilizes structural information extracted from Portable Executable (PE) files, including PE headers, imported libraries, opcode sequences, byte histograms, entropy values, and printable strings. These features provide valuable information for distinguishing malicious software from benign applications while reducing computational overhead, making static analysis particularly suitable for operational environments that require rapid detection.

The availability of benchmark datasets, particularly EMBER, has further accelerated research in static malware detection by providing standardized datasets for evaluating different machine learning algorithms under consistent experimental settings. Recent studies have demonstrated that PE-based feature representations can achieve high classification performance when combined with appropriate feature engineering and learning algorithms (El Neel et al., 2022; Oyama et al., 2019). Consequently, static analysis remains an important research direction for developing efficient malware detection systems capable of handling continuously evolving cyber threats.

Machine learning has become a dominant approach for static malware detection because it enables automatic identification of complex patterns that cannot be captured effectively by rule-based techniques. Both classical machine learning and deep learning methods have shown promising performance using PE-derived features, API representations, and byte-level characteristics (Kacem & Tossou, 2025; Karat et al., 2024; M. & Sethuraman, 2023; Maniriho et al., 2022). While recent research has increasingly focused on deep neural architectures, evidence suggests that classification performance depends not only on model complexity but also on feature quality, feature selection strategies, and optimization procedures (Imran, 2025; Lai et al., 2025; Yousuf et al., 2023). These findings indicate that carefully optimized machine learning models remain competitive despite the rapid development of more sophisticated deep learning techniques.

Classical machine learning algorithms, particularly Random Forest and XGBoost, continue to demonstrate strong performance for structured cybersecurity datasets because of their ability to handle high-dimensional feature spaces while maintaining relatively low computational costs (Hasan et al., 2025). Previous studies

have shown that ensemble learning methods, especially gradient boosting algorithms, achieve robust generalization and stable malware classification performance across different datasets (Darmawan et al., 2025; Rathod et al., 2021). Compared with deep neural networks, these approaches generally require fewer computational resources for training and inference, making them attractive for practical cybersecurity deployment where efficiency and scalability are important considerations.

Deep learning has further advanced static malware detection by enabling automatic extraction of complex feature representations from large-scale cybersecurity data. Various architectures, including Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), CNN-BiLSTM hybrids, transformer-based models, and multi-input neural networks, have been developed to improve malware classification using Portable Executable features, byte-level representations, and API sequences (Kim & Kim, 2024; M. & Sethuraman, 2023). CNN-based models are particularly effective in learning spatial patterns from byte sequences or malware images, whereas hybrid and transformer architectures are capable of capturing sequential and contextual relationships that may not be fully represented by conventional machine learning methods (Ashawa et al., 2024; Kim & Kim, 2024; Maniriho et al., 2022). These advances have significantly improved detection capability, especially when trained on large-scale benchmark datasets.

Despite their strong predictive performance, deep learning approaches present several practical challenges for operational cybersecurity environments. Their training process typically requires substantial computational resources, longer training time, and larger datasets, while model interpretability remains limited because prediction decisions are often difficult to explain (Ling et al., 2023). Moreover, concerns regarding adversarial robustness, deployment scalability, and resource consumption continue to affect the adoption of deep learning models in real-world security systems. Although recent studies have proposed hybrid architectures and cloud-based deployment strategies to improve flexibility and classification accuracy (Alzu'bi et al., 2024), questions regarding computational efficiency and practical implementation remain open. Consequently, comparative evaluation between optimized classical machine learning models and modern deep learning approaches is still essential to determine the most appropriate solution for large-scale malware detection.

Alongside improvements in predictive performance, the growing use of artificial intelligence in cybersecurity has increased the need for transparent and interpretable decision-making. Malware detection models are frequently deployed in high-risk environments where analysts must understand the reasons behind a classification result before taking defensive actions. Black-box prediction models therefore limit analyst confidence and reduce operational accountability. Explainable Artificial Intelligence (XAI) has emerged as an important research direction to address these concerns by providing meaningful explanations for machine learning predictions (Saqib et al., 2024a).

Among existing explainability techniques, SHAP (SHapley Additive exPlanations) has become one of the most widely adopted methods because it quantifies the contribution of individual features to model predictions. In malware

detection, SHAP enables analysts to identify how characteristics such as entropy values, imported APIs, DLL dependencies, byte distributions, and other Portable Executable features influence classification outcomes (Alenezi & Ludwig, 2021a). Beyond improving model transparency, explainability also supports behavioral interpretation of malware and assists security analysts in identifying meaningful indicators associated with malicious activities. Although several recent studies have incorporated explainable AI into malware detection and intrusion detection systems (Alani et al., 2023a; Saqib et al., 2024a), interpretability is often treated as a complementary analysis rather than an integral component of model evaluation. Furthermore, the relationship between explainability, behavior-aware feature engineering, computational efficiency, and fair benchmarking has received relatively limited attention, particularly in large-scale malware classification studies.

Although substantial progress has been achieved in malware detection, several important research gaps remain. Existing studies predominantly emphasize classification accuracy while giving less attention to computational efficiency, scalability, and deployment feasibility, all of which are critical for operational cybersecurity applications (Rathod et al., 2021). Comparative evaluations also frequently assess sophisticated deep learning architectures against minimally optimized classical machine learning baselines, potentially leading to biased conclusions regarding model superiority (Galen & Steele, 2020). In addition, behavior-aware import features associated with suspicious APIs, DLL usage, persistence mechanisms, and process manipulation remain insufficiently explored within benchmark-oriented EMBER studies (Şandor et al., 2023). Despite increasing interest in explainable AI, comprehensive investigations that jointly examine feature attribution, behavior-aware feature engineering, model optimization, and computational tradeoffs are still limited (Hafiz et al., 2024; Ling et al., 2023; Saqib et al., 2024a; Zhang et al., 2022).

Motivated by these limitations, this study proposes a comprehensive benchmark framework for large-scale static malware detection using the EMBER dataset. The proposed framework compares optimized classical machine learning models, including XGBoost and Random Forest, with modern deep learning approaches represented by Multi-Layer Perceptron (MLP) and TabNet under consistent experimental settings. In addition, the study integrates behavior-aware import feature engineering, SHAP-based explainability analysis, and computational efficiency evaluation covering training time, scalability, and inference performance. Through this integrated evaluation, the study aims to provide a balanced assessment of predictive performance, interpretability, and deployment efficiency while offering practical insights into the suitability of optimized machine learning approaches for real-world cybersecurity applications.

2. Method

2.1. Experimental Workfflow

This study adopts a benchmark-oriented experimental framework designed to evaluate both classical machine learning and modern deep learning models under a unified and reproducible setting. The overall workflow consists of dataset

preparation, feature extraction, behavior-aware feature engineering, model training, hyperparameter optimization, explainability analysis, and comparative benchmarking evaluation. Rather than focusing solely on predictive accuracy, the proposed methodology also emphasizes interpretability, robustness, and computational efficiency in order to better reflect real-world cybersecurity deployment requirements and evolving malware landscapes affected by concept drift and adversarial behavior (Barbero et al., 2022; Ling et al., 2023).

The experimental process begins with preprocessing the EMBER dataset and constructing feature representations derived from Portable Executable (PE) characteristics. In addition to baseline structural features, the framework incorporates behavior-aware import indicators intended to capture suspicious executable activities associated with malware behavior. Such behavior-oriented feature engineering has become increasingly important in modern malware research due to the growing diversity of attack strategies across desktop, Android, IoT, and cloud-based environments. The engineered features are subsequently used to train and evaluate multiple classification models representing different learning paradigms, including approaches commonly applied in multi-platform malware analysis studies (Ucci et al., 2019b).

Following model training, hyperparameter optimization is applied to reduce bias introduced by suboptimal default configurations and to ensure fair comparison across architectures. The optimized models are then analyzed using SHAP-based explainability techniques to identify influential structural and behavioral features contributing to malware classification decisions. Finally, benchmarking evaluation is conducted using both predictive and computational performance metrics, while also considering scalability, adaptability, and robustness issues that increasingly affect modern malware detection systems operating in dynamic threat environments (Nazim et al., 2025).

Although the proposed framework primarily focuses on static malware analysis, the methodological design is informed by recent developments in dynamic malware analysis and hybrid behavioral intelligence systems. Prior studies demonstrated that runtime monitoring, sandbox analysis, network traffic inspection, and behavioral API tracing can provide complementary semantic information for malware characterization (Guven, 2024). In addition, recent survey studies emphasized the importance of combining explainability, benchmarking fairness, adversarial robustness, and adaptive learning strategies within future malware detection frameworks (Maniriho et al., 2022).

The complete end-to-end pipeline employed in this study is illustrated in Figure 1. As shown in the figure, the proposed framework integrates feature engineering, model optimization, explainability analysis, and benchmarking evaluation into a unified malware detection workflow.

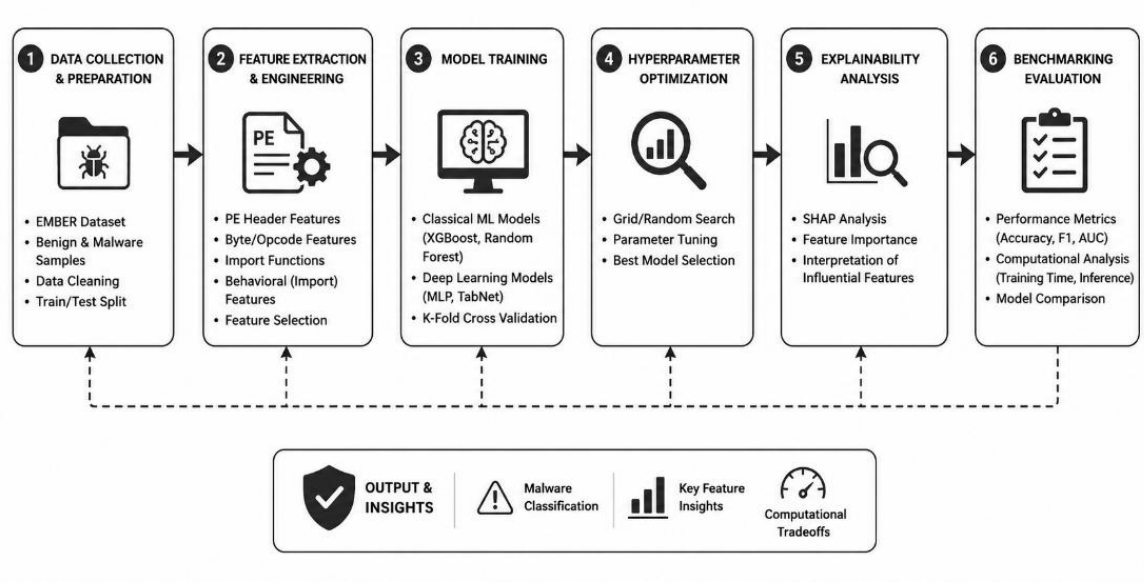

Figure 1. Proposed Experimental Workflow

2.2. Classical Machine Learning Models

2.2.1. XGBoost

XGBoost was selected as one of the primary baseline models because of its strong performance on structured and high-dimensional tabular datasets. The model employs gradient boosting optimization with regularization mechanisms that improve generalization capability while reducing overfitting risk. In malware detection tasks involving Portable Executable features, XGBoost has consistently demonstrated competitive classification performance due to its ability to capture nonlinear feature interactions efficiently (Darmawan et al., 2025; Imran, 2025). Recent studies further reported that gradient boosting approaches remain highly effective for PE-based malware classification, particularly when combined with feature selection and optimized tabular representations (Hashim, 2024)

Another important advantage of XGBoost lies in its computational scalability and compatibility with explainability frameworks such as SHAP TreeExplainer. Since the EMBER dataset primarily consists of structured numerical representations, gradient boosting approaches remain highly suitable for this classification setting. Moreover, XGBoost enables efficient training even when the feature space is extended through behavior-aware engineering and adaptive feature-centric optimization strategies (Imran, 2025).

2.2.2. Random Forest

Random Forest was included as a representative ensemble-based classical machine learning approach due to its robustness, interpretability, and stability across diverse feature distributions. The model constructs multiple decision trees using randomized feature selection and bootstrap aggregation, thereby reducing overfitting while improving generalization performance. Ensemble-based malware detection frameworks utilizing Random Forest have consistently demonstrated reliable classification capability across structured malware datasets and Portable Executable feature representations (Yousuf et al., 2023). Several studies also reported that

Random Forest remains highly competitive for static malware analysis because of its ability to handle complex feature interactions while maintaining stable predictive behavior under high-dimensional cybersecurity data distributions (Darmawan et al., 2025; Imran, 2025).

In malware classification scenarios, Random Forest often performs effectively because PE-based structural features naturally align with tree-based decision boundaries. Compared to more computationally intensive deep learning architectures, Random Forest also offers lower training complexity and relatively efficient inference capability, making it attractive for operational cybersecurity environments requiring scalable deployment. Recent benchmark and survey studies further highlighted that ensemble-based learning approaches continue to provide strong tradeoffs between classification accuracy, interpretability, and computational efficiency in practical malware detection systems (Ucci et al., 2019b).

2.3. Modern Deep Learning Models

2.3.1. Multi-Layer Perceptron (MLP)

The Multi-Layer Perceptron (MLP) model was employed to evaluate the effectiveness of dense neural architectures for malware classification using structured feature representations. Unlike tree-based ensemble methods, MLP learns hierarchical nonlinear relationships through stacked fully connected layers and activation functions. This capability allows the network to model complex interactions among malware-related features that may not be captured explicitly through manual feature engineering alone. Deep learning-based malware classification frameworks utilizing CNN, BiLSTM, transformer, and API-driven architectures have demonstrated that neural models are capable of learning highly discriminative behavioral and structural representations from malware-related data (Kim & Kim, 2024).

Recent malware detection studies also highlighted the growing effectiveness of neural architectures for processing image-transformed malware representations and Portable Executable feature sequences. CNN and transformer-based approaches, for example, have shown promising capability in capturing spatial and contextual malware characteristics across large-scale classification tasks (Ashawa et al., 2024). These findings motivate the inclusion of MLP within the proposed benchmark framework as a representative dense neural baseline for structured malware classification.

Despite its flexibility, MLP performance is often highly dependent on training configuration, regularization strategy, and optimization stability. Consequently, careful tuning was required to reduce overfitting and improve convergence behavior throughout the training process. Similar observations were reported in recent deep learning malware detection studies where training sensitivity, hyperparameter instability, and computational overhead significantly influenced model performance and scalability.

2.3.2. TabNet

TabNet was selected as a modern deep learning architecture specifically optimized for tabular learning tasks. Unlike conventional neural networks that process all features simultaneously, TabNet employs sequential attention mechanisms to dynamically select relevant features during training. This design enables more

interpretable feature selection while maintaining the representational power of deep learning architectures. Attention-driven tabular frameworks have recently demonstrated promising capability for malware detection and explainable cybersecurity analysis, particularly in environments involving structured feature representations and large-scale classification tasks (Naseer et al., 2025).

The inclusion of TabNet in this study serves an important benchmarking purpose because it represents a newer generation of neural approaches designed explicitly for structured tabular datasets such as EMBER. Recent studies integrating TabNet with explainable AI and ensemble learning strategies further reported improved interpretability and feature attribution capability in malware-related classification problems. In addition, attention-based architectures combining tabular learning and explainability mechanisms have demonstrated strong potential for Android malware analysis and secure web classification tasks (Ambekar et al., 2025).

Nevertheless, TabNet also introduces additional computational complexity and training sensitivity, making comparative evaluation against optimized classical approaches particularly relevant in malware detection research. As deep learning architectures become increasingly sophisticated, understanding the tradeoff between interpretability, scalability, and predictive performance remains an important challenge for practical cybersecurity deployment.

2.4. Hyperparameter Optimization

Hyperparameter optimization was conducted to ensure fair comparison among all evaluated models and to minimize performance bias caused by default parameter configurations. Since malware classification performance can vary substantially depending on tuning strategy, each model underwent optimization procedures involving learning rate adjustment, architecture configuration, batch size selection, and regularization parameter refinement. Recent malware detection studies consistently emphasized that model optimization and feature selection strategies significantly influence classification effectiveness, generalization capability, and robustness across large-scale malware datasets (Ucci et al., 2019a).

For classical machine learning models, optimization primarily focused on tree depth, learning rate, and ensemble-related parameters. Previous studies involving XGBoost and ensemble-based malware classifiers reported that careful tuning of boosting iterations, feature selection mechanisms, and regularization parameters substantially improves PE malware classification performance while reducing overfitting risk. Meanwhile, deep learning architectures required additional tuning involving epoch configuration, optimizer selection, regularization strategies, and training stability control. Similar observations were reported in recent CNN, transformer, and tabular deep learning studies where optimization sensitivity strongly affected convergence stability, scalability, and malware detection capability (Alzu'bi et al., 2024).

The optimized hyperparameter settings used throughout the experiments are summarized in Table 1. The table highlights several representative configurations employed for benchmark evaluation across classical and deep learning models.

Table 1. Hyperparameter Configuration

Models	Hyperparameter	Value
XGBoost (Tuned)	n_estimators	300
	max_depth	10
	learning_rate	0.05
	subsample	0.8
	colsample_bytree	0.8
	min_child_weight	3
	gamma	0.1
	reg_lambda	1.0
	eval_metric	logloss
Random Forest	n_estimators	100
	max_depth	None
	min_samples_split	2
	min_samples_leaf	1
	bootstrap	True
MLP	Hidden Layers	512, 256, 128
	Activation	ReLU
	Dropout	0.3
	Optimizer	Adam
	Learning Rate	0.001
	Batch Size	1024
	Epochs	20
	Scheduler	StepLR
	Weight Decay	1e-4
TabNet	n_d	64
	n_a	64
	n_steps	5
	gamma	1.3
	lambda_sparse	1e-4
	Optimizer	Adam
	Learning Rate	0.02
	Batch Size	1024
	Epochs	50

2.5. Explainability Framework

To improve interpretability and analyst-oriented transparency, this study incorporates SHAP (SHapley Additive exPlanations) as the primary explainability framework. Specifically, SHAP TreeExplainer was utilized to quantify feature contributions and analyze how individual structural and behavioral attributes influence malware classification decisions. The increasing adoption of explainable AI within cybersecurity research has demonstrated that transparent model interpretation is essential for improving analyst trust, operational accountability, and security decision support systems (Saqib et al., 2024a). Recent studies further highlighted that SHAP-based explainability techniques are highly effective for analyzing cybersecurity threat data, intrusion detection systems, and malware classification frameworks involving complex machine learning architectures (Alenezi & Ludwig, 2021a).

The explainability analysis serves several important objectives. First, it enables identification of highly influential malware-related features such as entropy

distributions, imported APIs, DLL usage patterns, and behavior-aware indicators. Second, SHAP analysis provides insight into how different models interpret malware characteristics, thereby improving understanding of model behavior beyond raw predictive metrics alone. Similar explainability-driven malware analysis frameworks demonstrated that interpretable feature attribution can substantially improve analyst understanding of malicious behavior patterns and classification reasoning processes (Alani et al., 2023a). In addition, explainable malware detection systems have increasingly been integrated with lightweight security architectures and memory-based analysis frameworks to improve operational transparency in practical cybersecurity environments.

Finally, interpretability analysis contributes to operational trustworthiness, which is particularly important in cybersecurity environments where analysts require transparent reasoning for malicious executable classification. By integrating explainability directly into the benchmarking framework, the proposed methodology moves beyond conventional black-box evaluation and provides a more comprehensive assessment of malware detection performance. This approach aligns with recent research trends emphasizing that explainability should function not merely as a supplementary visualization tool, but as a central component of trustworthy and deployable cybersecurity intelligence systems (Saqib et al., 2024a).

3. Result and Discussion

3.1. Experimental Benchmark Results

Table 2 summarizes the progressive benchmark experiments conducted throughout this study. The evaluation included baseline ensemble methods, behavior-enhanced configurations, hyperparameter-tuned models, and several neural architectures designed for tabular learning.

Table 2. Experimental progression and benchmark performance across all evaluated models.

Experiment	Model	Accuracy	ROC-AUC	Training Time
EXP-01	XGBoost Baseline	92.31%	0.9777	2481 s
EXP-02	XGBoost + Behavioral Import Features	92.63%	0.9780	2863 s
EXP-03	Tuned XGBoost	94.43%	0.9871	8335 s
EXP-04	Random Forest	92.07%	-	-
EXP-05	Baseline MLP	84.56%	0.9274	269 s
EXP-06	Optimized MLP	71.71%	0.8835	938 s
EXP-07	Regularized MLP	76.06%	0.9147	572 s
EXP-08	TabNet	-	0.7606	1842 s

As shown in Table 2, tree-based ensemble methods consistently outperformed the neural architectures across nearly all evaluation settings. Even the baseline XGBoost configuration achieved strong discrimination capability, reaching 92.31%

accuracy with a ROC-AUC score of 0.9777. After integrating additional behavior-aware import features, the model exhibited a measurable improvement in both accuracy and ranking performance.

The best overall result was obtained by the tuned XGBoost model, which achieved 94.43% accuracy and a ROC-AUC score of 0.9871. This improvement did not emerge from feature scaling or architectural complexity alone. Instead, it appears to result from a more effective balance between feature interaction learning, regularization, and sampling control during gradient boosting optimization

Interestingly, modern neural approaches did not demonstrate superior performance despite their higher representational capacity. The baseline MLP produced moderate results, whereas several optimization attempts led to noticeable performance degradation. TabNet also underperformed relative to ensemble-based methods, suggesting that the EMBER feature space remains particularly favorable for gradient boosting approaches rather than deep neural representations.

Taken together, these findings reinforce an important observation: for structured PE malware features, carefully optimized classical machine learning models continue to provide highly competitive—and often superior—performance compared with more complex deep learning alternatives.

3.2. Impact of Behavior-Aware Feature Engineering

To investigate whether lightweight behavioral indicators could improve malware discrimination capability, three additional import-oriented features were introduced into the original EMBER representation: DLL count, total imported functions, and suspicious API statistics.

Table 3. Performance impact of behavior-aware feature engineering

Experiment	Feature Configuration	Accuracy	ROC-AUC
EXP-01	Original EMBER Features (615)	92.31%	0.9777
EXP-02	Enhanced Behavioral Features (618)	92.63%	0.9780

The results presented in Table 3 indicate that the additional behavioral descriptors produced a consistent performance improvement, despite increasing the feature space by only three dimensions. Accuracy improved from 92.31% to 92.63%, while the ROC-AUC score also increased slightly.

Although the numerical gain may appear relatively modest, the result remains meaningful given the already high baseline performance. In large-scale malware classification settings, even small improvements can substantially reduce false detections across hundreds of thousands of executable samples.

More importantly, the added features introduced semantically meaningful behavioral information into the learning process without requiring expensive dynamic execution analysis. This observation suggests that lightweight import-based indicators can serve as an effective middle ground between purely structural static analysis and fully behavioral malware inspection pipelines.

3.3. Impact of Hyperparameter Optimization

Hyperparameter optimization was performed to examine whether additional tuning could further improve the performance of the behavior-enhanced XGBoost configuration.

Table 4. Performance improvement obtained through XGBoost hyperparameter optimization

Experiment	Optimization Strategy	Accuracy	ROC-AUC
EXP-02	Default XGBoost Parameters	92.63%	0.9780
EXP-03	Tuned XGBoost Parameters	94.43%	0.9871

As illustrated in Table 4, systematic optimization substantially improved classification capability. The tuned model increased accuracy by approximately 1.8 percentage points while simultaneously improving the ROC-AUC score to 0.9871.

The improvement appears to stem from several interacting factors. Increasing the number of estimators enabled the model to learn more refined decision boundaries, while deeper trees allowed more complex feature interactions to emerge. Additional regularization and sampling constraints further stabilized the learning process and reduced overfitting tendencies.

However, these gains came with a considerable computational cost. Training time increased from 2863 seconds to more than 8300 seconds, indicating a clear tradeoff between predictive performance and computational scalability. In practical deployment scenarios, such tradeoffs become highly relevant, particularly for organizations operating under constrained infrastructure or real-time detection requirements.

3.4. ROC Curve Analysis

The receiver operating characteristic analysis provides additional insight into the separability achieved by the optimized classifier.

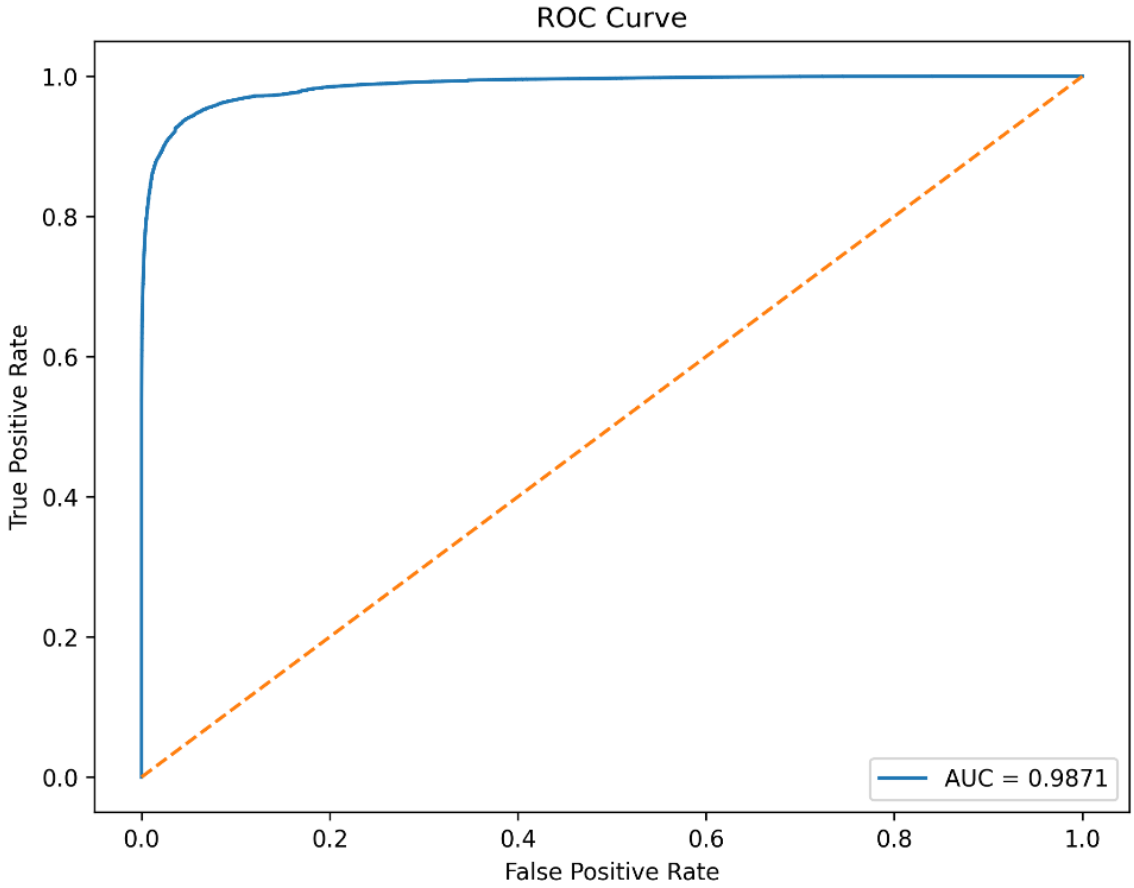

Figure 2. ROC curve of the tuned XGBoost malware classifier

As shown in Figure 2, the ROC curve rapidly approaches the upper-left corner of the feature space, indicating strong discrimination capability across varying classification thresholds. The resulting ROC-AUC score of 0.9871 confirms that the optimized model maintained highly reliable separation between benign and malicious samples.

This behavior is particularly desirable in operational cybersecurity environments. Malware detection systems must minimize false negatives without generating excessive false positives that could overwhelm analysts or disrupt legitimate activity. The ROC profile observed in this study suggests that the tuned XGBoost model achieves a favorable balance between these competing objectives.

The smooth curve progression also indicates stable probabilistic ranking behavior, which is essential for deployment scenarios involving adaptive threshold selection or risk-based malware prioritization.

3.5. Confusion Matrix Analysis

To further evaluate prediction behavior at the class level, a confusion matrix analysis was performed using the tuned XGBoost configuration.

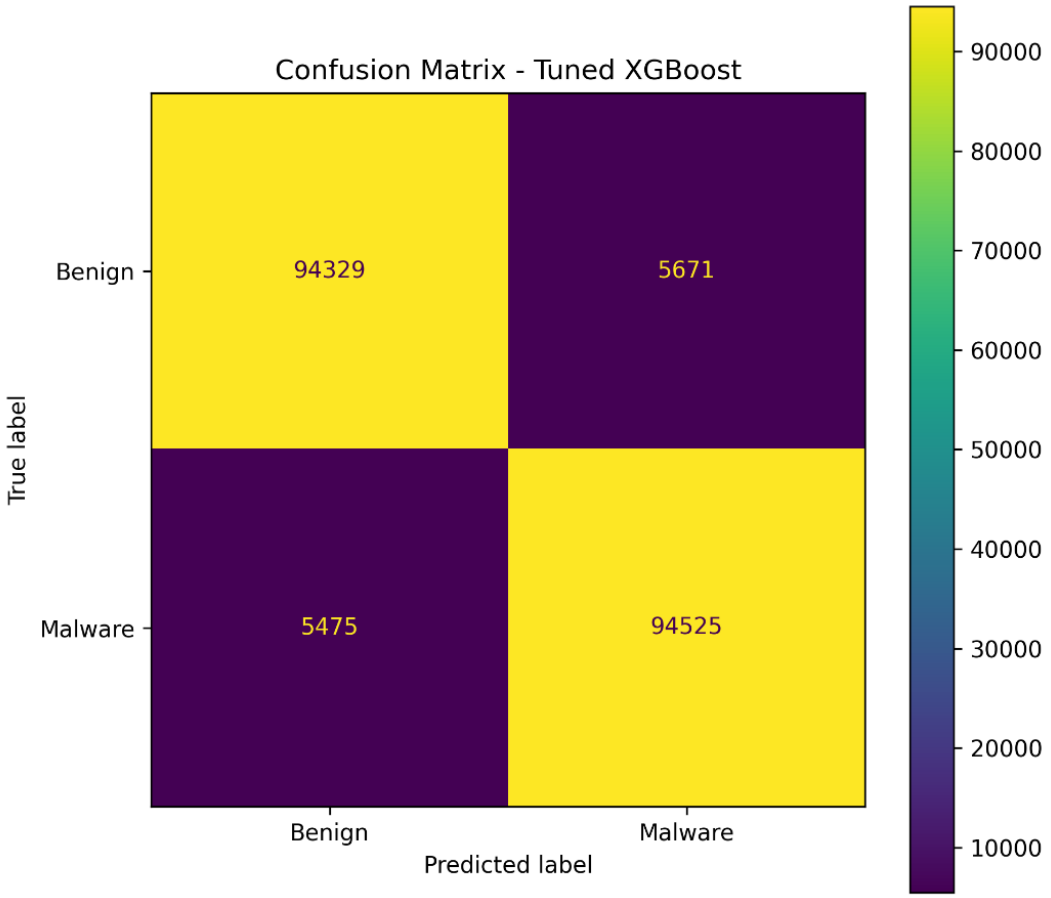

Figure 3. Confusion matrix of the tuned XGBoost model

Figure 3 demonstrates balanced classification performance across both malware and benign categories. The model correctly identified 94,329 benign samples and 94,525 malware samples, while maintaining relatively low false positive and false negative counts.

An interesting observation emerges from the error distribution itself. The number of false negatives remained slightly lower than false positives, which is generally preferable in malware detection scenarios because missed malware samples often pose greater operational risk than excessive alert generation.

The confusion matrix also suggests that the classifier did not exhibit severe class bias despite the substantial variability contained within the EMBER malware corpus. This stability further supports the suitability of ensemble-based learning for large-scale PE malware classification tasks.

3.6. SHAP Explanability Analysis

Beyond predictive performance alone, explainability remains an important requirement for modern cybersecurity systems. To investigate the internal decision behavior of the optimized classifier, SHAP analysis was conducted using the tuned XGBoost model.

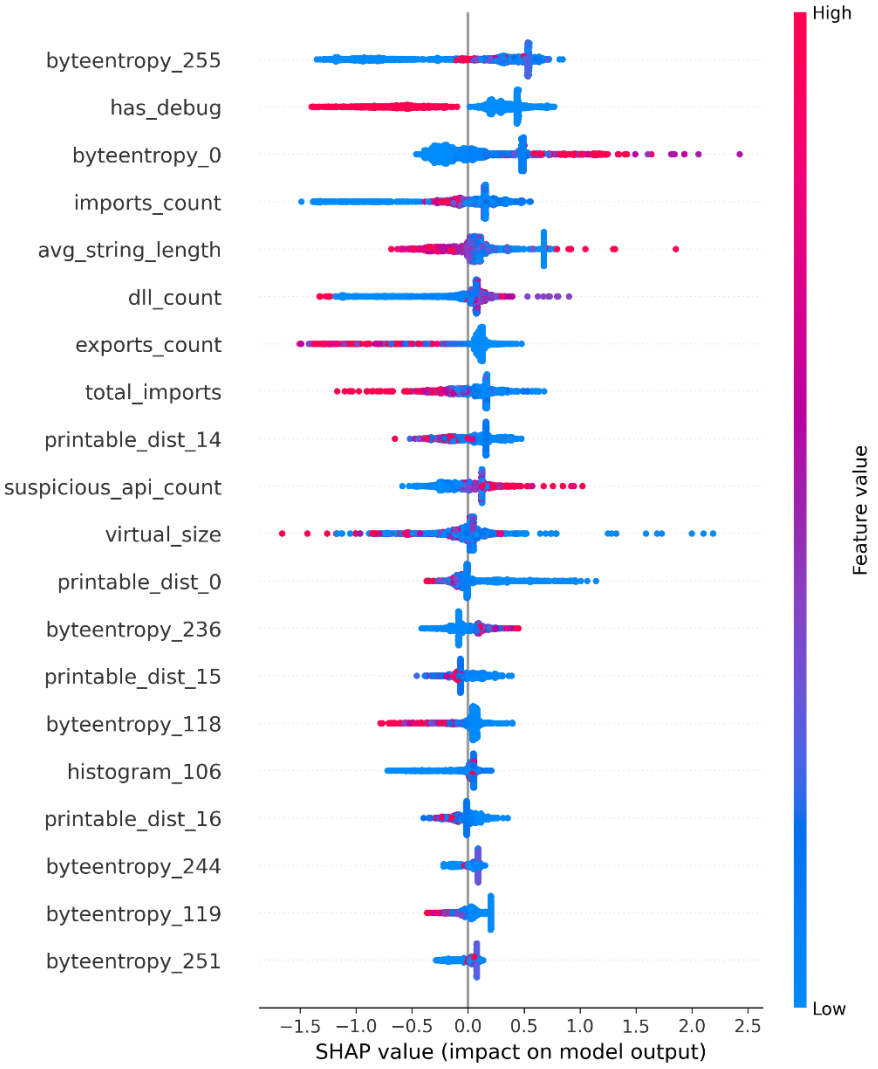

Figure 4. SHAP summary plot highlighting the most influential malware detection features

The SHAP summary plot shown in Figure 4 reveals that entropy-related features, import statistics, string characteristics, and behavioral indicators collectively contributed to the final classification decisions.

Several byte enthropy features consistently exhibited strong positive SHAP contributions toward malware predictions. This finding aligns with prior observations in malware analysis, where entropy irregularities often indicate obfuscation, compression, or packing behavior commonly associated with malicious executables.

Behavior-aware features also emerged as influential contributors despite representing only a small subset of the overall feature space. In particular, suspicious API counts, DLL-related statistics, and import distributions demonstrated meaningful impact on classification outcomes. The persistence of these features among the top SHAP contributors after hyperparameter optimization suggests that the additional

behavioral descriptors capture genuinely informative malware characteristics rather than merely introducing statistical noise.

Another notable observation concerns interpretability consistency. The optimized XGBoost model maintained relatively coherent feature attribution patterns across samples, indicating that the classifier relied on stable structural and behavioral indicators rather than isolated spurious correlations.

Overall, the explainability analysis strengthens the argument that combining structural PE representations with lightweight behavioral indicators can improve both predictive capability and model transparency in practical malware detection systems.

3.7. Computational Tradeoff Analysis

Although predictive performance remains important, computational efficiency also plays a critical role in practical cybersecurity deployment.

Table 5. Computational tradeoff comparison across evaluated malware detection models

Experiment	Model	Accuracy (%)	ROC-AUC	Training Time (s)
EXP-01	XGBoost Baseline	92.31	0.9777	2481
EXP-02	XGBoost + Behavioral Features	92.63	0.9780	2863
EXP-03	Tuned XGBoost	94.43	0.9871	8335
EXP-04	Random Forest	92.07	-	-
EXP-05	Baseline MLP	84.56	0.9274	269
EXP-06	Optimized MLP	71.71	0.8835	938
EXP-07	Regularized MLP	76.06	0.9147	572
EXP-08	TabNet	-	0.7606	1842

As presented in Table 5, the tuned XGBoost configuration delivered the strongest predictive performance but also required the highest computational cost. In contrast, neural models such as MLP trained significantly faster, although their classification performance remained considerably lower. This tradeoff highlights an important practical consideration. Higher architectural complexity does not automatically translate into superior malware classification capability, particularly for structured tabular feature spaces such as EMBER.

From a deployment perspective, the results suggest that classical ensemble methods currently offer a more favorable balance between predictive accuracy, interpretability, robustness, and operational feasibility for large-scale static malware detection pipelines.

4. Conclusion

This study proposed a benchmark-oriented framework for static malware detection that combines behavior-aware feature engineering, explainable artificial intelligence (XAI), and computational tradeoff analysis within a unified evaluation framework using the EMBER dataset. The results demonstrate that optimized classical machine learning models, particularly XGBoost and Random Forest, remain highly

effective for structured malware classification and can achieve competitive performance compared with modern deep learning models. The findings further indicate that feature engineering and model optimization contribute substantially to detection performance, highlighting that architectural complexity alone does not necessarily lead to superior classification results. The incorporation of behavior-aware import features improved malware characterization by capturing semantic information related to suspicious executable behavior, while SHAP-based explainability analysis enhanced model transparency by identifying the contribution of individual features to prediction outcomes. In addition, the evaluation of computational efficiency suggests that lightweight and interpretable machine learning models provide practical advantages for large-scale malware analysis, Security Operation Centers (SOCs), and resource-constrained cybersecurity environments.

This study proposed a benchmark-oriented framework for static malware detection that combines behavior-aware feature engineering, explainable artificial intelligence (XAI), and computational tradeoff analysis within a unified evaluation framework using the EMBER dataset. The results demonstrate that optimized classical machine learning models, particularly XGBoost and Random Forest, remain highly effective for structured malware classification and can achieve competitive performance compared with modern deep learning models. The findings further indicate that feature engineering and model optimization contribute substantially to detection performance, highlighting that architectural complexity alone does not necessarily lead to superior classification results. The incorporation of behavior-aware import features improved malware characterization by capturing semantic information related to suspicious executable behavior, while SHAP-based explainability analysis enhanced model transparency by identifying the contribution of individual features to prediction outcomes. In addition, the evaluation of computational efficiency suggests that lightweight and interpretable machine learning models provide practical advantages for large-scale malware analysis, Security Operation Centers (SOCs), and resource-constrained cybersecurity environments.

Although the proposed framework demonstrates promising performance, several limitations remain. This study is limited to static malware analysis using the EMBER dataset and does not consider runtime behavior or dynamic sandbox execution, which may provide complementary information for malware characterization. Future research may extend this work by integrating static and dynamic analysis, investigating transformer- and graph neural network-based architectures, exploring federated learning for privacy-preserving malware detection, and improving adversarial robustness against evasion attacks. Additional efforts are also needed to address concept drift through adaptive learning strategies and to evaluate model generalization across heterogeneous platforms, including Windows, Android, IoT, and cloud environments. These directions are expected to support the development of more scalable, interpretable, robust, and operationally applicable malware detection systems.

Acknowledgement

The authors would like to express their sincere appreciation to their respective institutions for providing academic support and access to research resources that contributed to the completion of this study. The authors are also grateful to the reviewers and editors for their valuable comments and constructive suggestions that helped improve the quality of this manuscript.

Author Contributions

Intan Permata Sari: Conceptualization, Methodology, Data collection, Formal analysis, Writing—original draft, Writing—review and editing.
Rita Hamdani: Conceptualization, Methodology, Visualization, Writing—review and editing, Supervision.

Funding

This research received no external funding.

Declaration on the Use of Artificial Intelligence

Artificial intelligence (AI) support was used in a limited capacity through ChatGPT (OpenAI), based on the GPT-5 family model, solely for paraphrasing and language editing, including improvements to grammar, sentence structure, clarity, and overall readability of the manuscript. The AI tool was not used for conceptual development, research design, data collection, data analysis, interpretation of results, generation of findings, literature selection, citation management, or any scholarly decision-making. All scientific content, methodological design, data interpretation, intellectual contributions, and the final review and approval of the manuscript were carried out entirely by the authors. The authors accept full responsibility for the accuracy, originality, integrity, and scientific validity of the published work.

References

Alani, M. M., Mashatan, A., & Miri, A. (2023a). XMal: A lightweight memory-based explainable obfuscated-malware detector. *Computers & Security*, *133*, 103409. https://doi.org/10.1016/j.cose.2023.103409

Alani, M. M., Mashatan, A., & Miri, A. (2023b). XMal: A lightweight memory-based explainable obfuscated-malware detector. *Computers & Security*, *133*, 103409. https://doi.org/10.1016/j.cose.2023.103409

Alenezi, R., & Ludwig, S. A. (2021a). Explainability of Cybersecurity Threats Data Using SHAP. *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, 01–10. https://doi.org/10.1109/SSCI50451.2021.9659888

Alenezi, R., & Ludwig, S. A. (2021b). Explainability of Cybersecurity Threats Data Using SHAP. *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, 01–10. https://doi.org/10.1109/SSCI50451.2021.9659888

Al-Issa, A. (2005). The Role of English Language Culture in the Omani Language Education System: An Ideological Perspective. *Language, Culture and Curriculum*, *18*(3), 258–270. https://doi.org/10.1080/07908310508668746

Aliyah, I. (2016). The Roles of Traditional Markets as the Main Component of Javanese Culture Urban Space (Case Study: The City of Surakarta, Indonesia). *IAFOR Journal of Sustainability, Energy & the Environment*, *3*(1). https://doi.org/10.22492/ijsee.3.1.06

Alzu'bi, A., Abuarqoub, A., Abdullah, M., Agolah, R. A., & Al Ajlouni, M. (2024). *Malware Prediction Using Tabular Deep Learning Models* (pp. 379–389). https://doi.org/10.1007/978-3-031-47508-5_30

Ambekar, N. G., Devi, N. N., Thokchom, S., & Yogita. (2025). TabLSTMNet: enhancing android malware classification through integrated attention and explainable AI. *Microsystem Technologies*, *31*(3), 695–713. https://doi.org/10.1007/s00542-024-05615-0

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

Balasubramanian, K. M., Vasudevan, S. V., Thangavel, S. K., T, G. K., Srinivasan, K., Tibrewal, A., & Vajipayajula, S. (2023). Obfuscated Malware detection using Machine Learning models. *2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, 1–8. https://doi.org/10.1109/ICCCNT56998.2023.10307598

Barbero, F., Pendlebury, F., Pierazzi, F., & Cavallaro, L. (2022). Transcending TRANSCEND: Revisiting Malware Classification in the Presence of Concept Drift. *2022 IEEE Symposium on Security and Privacy (SP)*, 805–823. https://doi.org/10.1109/SP46214.2022.9833659

Darmawan, I., Setiaji, H., Sutriani, L., Supriyadi, A., Rizal, R., & Rahmatulloh, A. (2025). Ensemble Learning for Malware Classification: A Performance Evaluation Using Random Forest and Hist Gradient Boosting. *2025 International Conference on Information and Communication Technology (ICoICT)*, 1–6. https://doi.org/10.1109/ICoICT66265.2025.11193116

El Neel, L., Copiaco, A., Obaid, W., & Mukhtar, H. (2022). Comparison of Feature Extraction and Classification Techniques of PE Malware. *2022 5th International Conference on Signal Processing and Information Security (ICSPIS)*, 26–31. https://doi.org/10.1109/ICSPIS57063.2022.10002693

Galen, C., & Steele, R. (2020). Evaluating Performance Maintenance and Deterioration Over Time of Machine Learning-based Malware Detection Models on the EMBER PE Dataset. *2020 Seventh International Conference on Social Networks Analysis, Management and Security (SNAMS)*, 1–7. https://doi.org/10.1109/SNAMS52053.2020.9336538

Guven, M. (2024). Dynamic Malware Analysis Using a Sandbox Environment, Network Traffic Logs, and Artificial Intelligence. *International Journal of Computational and Experimental Science and Engineering*, *10*(3). https://doi.org/10.22399/ijcesen.460

Hafiz, Md. F. Bin, Khan, N. A., Kamal, Z., Hossain, S., & Barman, S. (2024). A Robust Malware Classification Approach Leveraging Explainable AI. *2024 International Conference on Intelligent Systems for Cybersecurity (ISCS)*, 1–6. https://doi.org/10.1109/ISCS61804.2024.10581382

Hasan, R., Biswas, B., Samiun, M., Saleh, M. A., Prabha, M., Akter, J., Joya, F. H., & Abdullah, M. (2025). Enhancing malware detection with feature selection and scaling techniques using machine learning models. *Scientific Reports*, *15*(1), 9122. https://doi.org/10.1038/s41598-025-93447-x

Hashim, M. et al. (2024). *Enhancing XGBoost Performance in Malware Detection through Chi-Squared Feature Selection*. https://www.researchgate.net/publication/386096117

Imran, M. F. (2025). Malware classification using SVM and XGBoost: A study of the influence of features. *Journal of Al-Qadisiyah for Computer Science and Mathematics*, *17*(4). https://doi.org/10.29304/jqcsm.2025.17.42536

Kacem, T., & Tossou, S. (2025). Trandroid: An Android Mobile Threat Detection System Using Transformer Neural Networks. *Electronics*, *14*(6), 1230. https://doi.org/10.3390/electronics14061230

Karat, G., Kannimoola, J. M., Nair, N., Vazhayil, A., G, S. V, & Poornachandran, P. (2024). CNN-LSTM Hybrid Model for Enhanced Malware Analysis and Detection. *Procedia Computer Science*, *233*, 492–503. https://doi.org/10.1016/j.procs.2024.03.239

Kim, H., & Kim, M. (2024). Malware Detection and Classification System Based on CNN-BiLSTM. *Electronics*, *13*(13), 2539. https://doi.org/10.3390/electronics13132539

Lai, T.-H., Tsai, Y.-J., & Liu, C.-L. (2025). Improving the Performance of Static Malware Classification Using Deep Learning Models and Feature Reduction Strategies. *Mathematics*, *13*(23), 3753. https://doi.org/10.3390/math13233753

Ling, X., Wu, L., Zhang, J., Qu, Z., Deng, W., Chen, X., Qian, Y., Wu, C., Ji, S., Luo, T., Wu, J., & Wu, Y. (2023). Adversarial attacks against Windows PE malware detection: A survey of the state-of-the-art. *Computers & Security*, *128*, 103134. https://doi.org/10.1016/j.cose.2023.103134

M., G., & Sethuraman, S. C. (2023). A comprehensive survey on deep learning based malware detection techniques. *Computer Science Review*, *47*, 100529. https://doi.org/10.1016/j.cosrev.2022.100529

Maniriho, P., Mahmood, A. N., & Chowdhury, M. J. M. (2022). A study on malicious software behaviour analysis and detection techniques: Taxonomy, current trends

and challenges. *Future Generation Computer Systems*, *130*, 1–18. https://doi.org/10.1016/j.future.2021.11.030

Naseer, M., Ullah, F., Saeed, S., Algarni, F., & Zhao, Y. (2025). Explainable TabNet ensemble model for identification of obfuscated URLs with features selection to ensure secure web browsing. *Scientific Reports*, *15*(1), 9496. https://doi.org/10.1038/s41598-025-93286-w

Nazim, S., Alam, M. M., Rizvi, S. S., Mustapha, J. C., Hussain, S. S., & Suud, M. M. (2025). Advancing malware imagery classification with explainable deep learning: A state-of-the-art approach using SHAP, LIME and Grad-CAM. *PLOS One*, *20*(5), e0318542. https://doi.org/10.1371/journal.pone.0318542

Oyama, Y., Miyashita, T., & Kokubo, H. (2019). Identifying Useful Features for Malware Detection in the Ember Dataset. *2019 Seventh International Symposium on Computing and Networking Workshops (CANDARW)*, 360–366. https://doi.org/10.1109/CANDARW.2019.00069

Rathod, V., Parekh, C., & Dholariya, D. (2021). AI & ML Based Anamoly Detection and Response Using Ember Dataset. *2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, 1–5. https://doi.org/10.1109/ICRITO51393.2021.9596451

Şandor, M., Portase, R. M., & Coleşa, A. (2023). Ember Feature Dataset Analysis For Malware Detection. *2023 IEEE 19th International Conference on Intelligent Computer Communication and Processing (ICCP)*, 203–210. https://doi.org/10.1109/ICCP60212.2023.10398693

Saqib, M., Mahdavifar, S., Fung, B. C. M., & Charland, P. (2024a). A Comprehensive Analysis of Explainable AI for Malware Hunting. *ACM Computing Surveys*, *56*(12), 1–40. https://doi.org/10.1145/3677374

Saqib, M., Mahdavifar, S., Fung, B. C. M., & Charland, P. (2024b). A Comprehensive Analysis of Explainable AI for Malware Hunting. *ACM Computing Surveys*, *56*(12), 1–40. https://doi.org/10.1145/3677374

Tsuchiya, K. (2018). Javanology and the Age of Ranggawarsita: An Introduction to Nineteenth-Century Javanese Culture. In T. Shiraishi (Ed.), *Reading Southeast Asia* (pp. 75–108). Cornell University Press. https://doi.org/10.7591/9781501718922-005

Ucci, D., Aniello, L., & Baldoni, R. (2019a). Survey of machine learning techniques for malware analysis. *Computers & Security*, *81*, 123–147. https://doi.org/10.1016/j.cose.2018.11.001

Ucci, D., Aniello, L., & Baldoni, R. (2019b). Survey of machine learning techniques for malware analysis. *Computers & Security*, *81*, 123–147. https://doi.org/10.1016/j.cose.2018.11.001

Widodo, S. T. (2020). Norms and teachings in the art of lovemaking of kings in ancient Javanese manuscripts. *Rupkatha Journal on Interdisciplinary Studies in Humanities*, *12*(1), 1–12. https://doi.org/10.21659/rupkatha.v12n1.30

Wijaya, D. A., Djono, D., & Ediyono, S. (2018). Local Knowledge in Joglo Majapahit: Analysis of Local Wisdom Models Gemah Ripah Loh Jinawi in Rural Java. *International Journal of Multicultural and Multireligious Understanding*, *5*(3), 113. https://doi.org/10.18415/ijmmu.v5i3.235

Yousuf, M. I., Anwer, I., Riasat, A., Zia, K. T., & Kim, S. (2023). Windows malware detection based on static analysis with multiple features. *PeerJ Computer Science*, *9*, e1319. https://doi.org/10.7717/peerj-cs.1319

Zhang, Z., Hamadi, H. Al, Damiani, E., Yeun, C. Y., & Taher, F. (2022). Explainable Artificial Intelligence Applications in Cyber Security: State-of-the-Art in Research. *IEEE Access*, *10*, 93104–93139. https://doi.org/10.1109/ACCESS.2022.3204051