International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(12s) 2026 pp. 115-124
https://doi.org/10.70917/ijcisim-2026-3883
Article

AI-Based Adaptive Malware Detection Using Portable Executable (PE) Header Analysis and Hybrid Machine Learning

V Padmapriya[1*], S Uma[2], S Sumathi[3], Abhay John Mathais[4]
[1*,2,4]Department of Computer Applications, B. M. S. College of Engineering, Bengaluru, Karnataka, India.
[3]Department of Electrical and Electronics Engineering, RNS Institute of Technology, Bengaluru, Karnataka, India.
Email: padmapriyav.mca@bmsce.ac.in , uma.mca@bmsce.ac.in, sumathi.s@rnsit.ac.in , abhay.mca24@bmsce.ac.in
Corresponding Author: V Padmapriya (padmapriyav.mca@bmsce.ac.in)

Abstract: With the rapid evolution of malware and sophisticated evasion techniques used, the efficiency of traditional detection approaches will be increasingly compromised since malware creators frequently leverage polymorphic, metamorphic, packing, encryption, and code obfuscation techniques to circumvent traditional security methods. This work includes an AI-based adaptive malware detection system that utilizes the Portable Executable (PE) headers along with hybrid machine learning models. The system conducts static analysis of malware without the need to run malware in sandboxed environments to gather information regarding the structure of executable files. In total, the system trains and tests three machine learning models - Random Forest, eXtreme Gradient Boosting (XGBoost), and Artificial Neural Network (ANN). Training and testing are conducted on a dataset of 19,611 executable files described with 79 characteristics extracted from Portable Executable headers. To enhance predictive accuracy, a hybrid ensemble model is used, while SHapley Additive exPlanations (SHAP) enable predictability. The experiments conducted yielded an accuracy of 99.13% and an Area Under the Curve (AUC) value of 0.9984 with an extremely low false negative rate. The proposed framework combines a hybrid ensemble of Random Forest and XGBoost classifiers with SHapley Additive exPlanations (SHAP) to provide accurate and interpretable malware detection using static PE-header analysis. Experimental results demonstrate that the proposed approach achieves high classification performance while improving transparency in malware detection decisions, making it suitable for practical cybersecurity applications.

Keywords: Adaptive Malware Detection, Artificial Neural Network, Cybersecurity, Explainable Artificial Intelligence, Hybrid Ensemble Learning, Machine Learning, Portable Executable (PE) Analysis, Random Forest, SHAP, XGBoost.

1. Introduction

The rapid development of digital technologies, cloud computing solutions, enterprise networks, and the Internet has increased the number of cyberattacks. Malware is among the major security risks companies face and can cause significant losses, data theft, process disruptions, and unauthorized access to sensitive information. Traditional solutions for detecting malware use signatures for detecting malicious executables in relation to patterns stored in a database of threats. This kind of approach is effective when dealing with known threats but cannot detect new categories of malware, which can adapt by changing their code to avoid detection. The current generation of malware uses sophisticated evasion strategies such as code obfuscation, encryption, packing, and changes in behavior patterns [1]. This means that malware can alter its appearance without losing its functionality, and hence it makes it difficult to detect malware using conventional signature-based methods. With the rise in sophisticated cyber threats, there arises a great need for smart malware detection tools.

AI-based learning methods are considered among the most successful ones in detecting malware because they can recognize hidden patterns in large data sets as well as generalize to never-before-seen instances of malicious software. The efficiency of various machine learning models in discriminating between malicious and benign programs using either the static or dynamic attributes has been proven by numerous research works [2].

In particular, random forests, gradient boosting techniques, and artificial neural networks outperform traditional antivirus solutions at detecting malware.

Several methods can be used for the analysis of malware; however, static analysis has been widely utilized due to its ability to categorize malware quickly without executing potentially damaging code. The information obtained by using static analysis is structural information or metadata that can be obtained from executable files. For software written for the Microsoft operating system, the Portable Executable (PE) headers provide important information including header characteristics, section properties, subsystem characteristics, linker info, and memory allocation information [3].

Several studies have indicated that PE header features possess good discrimination capabilities when it comes to detecting malware [4]. Despite significant progress in malware detection, many existing approaches primarily focus on improving classification accuracy while offering limited interpretability. Explainable Artificial Intelligence (XAI), particularly SHapley Additive exPlanations (SHAP), improves transparency by explaining how individual features influence malware classification decisions. Many existing malware detection systems rely on a single machine learning model or computationally intensive dynamic analysis, limiting their scalability and generalization. These limitations motivate the development of an efficient static malware detection framework based on Portable Executable (PE) header analysis. In this regard, an AI-powered malware detection framework that employs the combination of PE header analysis and hybrid machine learning is introduced in this paper.

The suggested framework applies three machine learning methods, namely Random Forest, XGBoost, and Artificial Neural Network (ANN), in a hybrid ensemble framework. Besides, SHapley Additive exPlanations (SHAP) are used to ensure interpretability in the malware classification task. The evaluation process of the framework involved 19,611 executable files with 79 PE-header attributes as the training data. As shown in the experimental results, the presented approach has achieved 99.13% accuracy and an AUC of 0.9984.

The key highlights of this work are given below:

- Propose a hybrid machine learning approach that integrates Random Forest, XGBoost, and Artificial Neural Network models for malware detection.
- Adoption of features in PE headers to enable static malware analysis.
- Application of SHapley Additive exPlanations (SHAP) for explaining malware detection decisions.
- Implementation of a comparison between individual models and the hybrid machine learning model using PE header data.
- Ensured malware detection with low levels of false negatives. Demonstrated the potential of hybrid machine learning and XAI for adaptive malware detection.

2. Related Work

Malware detection has evolved from traditional signature-based techniques to machine learning-based approaches capable of identifying previously unseen threats. Schultz et al. demonstrated that features extracted from executable files could be used for malware classification without relying solely on signatures [1]. Whereas the authors Kolter and Maloof further established the effectiveness of machine learning using executable-based features for malware detection [2].

Among machine learning approaches, Random Forest has been widely adopted because of its ability to handle high-dimensional data and provide strong generalization performance [3]. Recent studies have also reported its effectiveness for malware detection tasks [4]. Gradient boosting methods such as XGBoost further improve prediction accuracy through regularization and efficient handling of structured data [6]. Deep learning techniques including CNNs, RNNs, and LSTMs have also shown promising performance; however, they often require extensive computational resources and provide limited interpretability [7, 8]. Quantum AI is one such recently developed methodology which can facilaitate fast and efficient way to overcome complex computations in fasted possible time. [9]

Recent research has also focused on practical challenges such as malware obfuscation, executable packing, and concept drift. Hasan and Dhakal investigated malware obfuscation using memory analysis, while Gibert et al. studied the impact of executable packing on machine learning models [10, 11]. Alam et al. proposed the ADAPT framework to address concept drift using pseudo-labeling techniques [12]. To improve model transparency, SHapley Additive exPla-nations (SHAP) have been widely adopted to explain feature contributions in machine learning predictions [13].

Although existing studies have demonstrated the effectiveness of individual machine learning models, deep learning techniques, and concept drift adaptation, no single approach consistently performs well across different malware scenarios [14]. Consequently, recent research has increasingly explored hybrid and ensemble learning strategies that combine the strengths of multiple classifiers to improve malware detection performance.

The comparison presented in Table 1 indicates that existing malware detection approaches either rely on individual machine learning models that are computationally intensive dynamic analysis, or lack interpretability. Although several studies have reported high classification accuracy, limited attention has been given to combining hybrid machine learning with explainable artificial intelligence using Portable Executable (PE) header features. To address this gap, the proposed framework integrates a hybrid ensemble of Random Forest and XGBoost classifiers with SHAP-based explainability to improve classification robustness while providing transparent and interpretable malware detection using static Portable Executable (PE) header analysis. Unlike existing approaches that rely on individual classifiers or computationally intensive dynamic analysis, the proposed framework combines efficient static analysis with explainable machine learning to support practical cybersecurity applications.

Table 1: Comparison of Existing Malware Detection Approaches

Reference	**Method**	**Analysis Type**	**Limitation**
Schultz et al. [1]	Data Mining	Static	Limited capability for modern malware variants.
Hasan and Dhakal [10]	Memory Analysis	Dynamic	High computational overhead and execution required.
Alam et al. [12]	Pseudo-labeling (ADAPT)	Static	Focuses on concept drift without explainable AI.
Vijayalakshmi et al. [4]	Comparative ML Models	Static	Evaluates individual models without ensemble explainability.
Alshoulie and Mehmood [9]	Deep Learning	Static/Dynamic	High computational complexity and limited interpretability.
Proposed Framework	**Random Forest + XGBoost + SHAP**	**Static (PE Header)**	**Provides hybrid classification with interpretable predictions.**

3. System Design and Implementation

The proposed system analyzes Portable Executable (PE) headers and applies machine learning and ensemble learning along with XAI algorithms in detecting malware. The system does not execute malicious programs but uses the internal structure of Windows executables and uses static analysis to detect possible threats. Various types of machine learning classifiers are used together to ensure more accurate results. The process of prediction becomes easier to interpret using SHapley Additive exPlanations (SHAP) and helps security experts understand which specific features played an important role in classification decisions.

3.1 Architectural Design

The proposed framework consists of five processing layers designed to work collectively to detect malware and provide interpretable classification results as described below:

1. Data Acquisition Layer*:* Retrieves PE header information from executable files. A total of 79 features are retrieved per sample, including subsystem information, linker information, memory allocation details, executable information, and code-related information.

2. Data Preprocessing Layer: Cleaned the collected data by discarding irrelevant attributes, treating any possible missing values, and normalizing numeric features. This step ensures that the data remains consistent before use for model training.

3. Feature Engineering Layer: Numerical representation of the collected PE header information.

4. Hybrid Classification Layer*:* Parallel execution of the Random Forest and eXtreme Gradient Boosting (XGBoost) classifiers. The prediction probabilities generated by both classifiers are combined using a soft-voting ensemble strategy to produce the final classification result. The Artificial Neural Network (ANN) model is trained and evaluated independently for performance comparison.

5. Explainability and Decision Layer: Utilization of SHapley Additive exPlanations to determine the attributes responsible for classification decisions. Based on an aggregated prediction, executable files will be classified into malware and benign categories.

The multi-layer architecture ensures efficiency in malware detection and provides explainability in the process.

3.2 Data Set Extraction and Preprocessing

The dataset utilized in this research includes 19,611 Windows executable files, represented by their 79 Portable Executable (PE) header features. In total, 14,599 samples belong to the class of malicious programs while 5,012 samples correspond to the class of benign executables. Although the dataset contains a larger number of malicious samples than benign samples, stratified sampling was employed during the train-test split to preserve the original class distribution in both subsets. This approach helps ensure that the machine learning models are trained and evaluated on representative data while reducing potential bias caused by class imbalance.

The collected PE header attributes are *TimeDateStamp, MajorSubsystemVersion, Characteristics, ImageBase, MajorLinkerVersion, SizeOfCode,* among others, which can provide information about the structural aspects of executables. To improve the models' generalization, non-informative features, such as the sample file name, file path, hashes, and unique identifiers, were removed from the initial dataset. Furthermore, the dataset was divided using stratified sampling, with 80% of the samples allocated for training and the remaining 20% reserved for testing. A fixed random state of 42 was used during the train-test split to ensure reproducibility while preserving the original class distribution. The reported performance metrics are based on this stratified 80:20 train-test split, and cross-validation was not performed in the present study.

Before the modeling stage, standardization of the selected features was performed according to equation 1 shown below:

$$z_i = \frac{x_i - \mu_i}{\sigma_i} \quad \textbf{(1)}$$

where xi, μi, and σi denote the initial value of the feature, its mean value, and its standard deviation, respectively.

3.3 Machine Learning Methodology

The suggested framework uses a mixture of three types of machine learning algorithms, such as Random Forest, eXtreme Gradient Boosting (XGBoost), and Artificial Neural Network (ANN).

3.3.1 Random Forest

The Random Forest algorithm is one where several decision trees are constructed using bootstrap sampling and feature randomness. In other words, instead of a single decision tree, the final output classification is produced using a vote from all the trees in the forest. In this work, the Random Forest algorithm consisted of 100 decision trees. Using feature randomness while splitting nodes of a tree leads to greater diversity within the forest. The number of decision trees was selected based on empirical experimentation to achieve a balance between classification performance and computational efficiency. A fixed random state of 42 was used to ensure reproducibility of the experimental results.

3.3.2 eXtreme Gradient Boosting (XGBoost)

In the XGBoost algorithm, decision trees are created sequentially, and each subsequent tree corrects the errors made by previous trees. This work uses the XGBoost classifier that is designed with 200 estimators, tree depth set at 8, and a learning rate of 0.1. Some regularization mechanisms were used to enhance the ability of this model to generalize and avoid overfitting. The selected hyperparameters were determined through empirical experimentation

by evaluating different parameter settings and choosing those that provided stable classification performance while minimizing overfitting. A fixed random state of 42 was used to ensure reproducibility.

3.3.3 Artificial Neural Network (ANN)

Artificial Neural Network (ANN) model is applied to capture non-linear dependencies among features in the Portable Executable header. ANN model used in this research contains an input layer, two hidden layers, consisting of 128 neurons and 64 neurons, respectively, as well as a sigmoid output layer. ReLU activation functions have been used in the two hidden layers to help facilitate learning of feature representations. To minimize overfitting effects, the dropout regularization technique has been implemented. Adam optimization technique and binary cross-entropy have been used as loss functions in the ANN. The ANN was trained for 20 epochs using a batch size of 64 with a validation split of 0.2. The network architecture and training parameters were selected through empirical experimentation to achieve stable convergence while reducing the risk of overfitting.

3.3.4 Ensemble Learning of Hybrid Models

The proposed framework employs a hybrid ensemble consisting of the Random Forest and eX-treme Gradient Boosting (XGBoost) classifiers. Instead of relying on a single classifier for malware detection, the prediction probabilities generated by both models are combined using a soft-voting ensemble strategy to obtain the final classification result. The Artificial Neural Network (ANN) model is trained and evaluated independently to compare its classification performance with that of the hybrid ensemble. The performance of the ensemble is given by Equation 2:

$$P_{ensemble} = w1P_{RF} + w2P_{XGB} \quad (2)$$

where PRF and PXGB denote the prediction probabilities generated by the Random Forest and XGBoost classifiers, respectively. Equal weights were assigned to both classifiers during the soft-voting process (w1 = w2 = 0.5), allowing each classifier to contribute equally to the final prediction without introducing additional weight optimization. By combining the complementary strengths of Random Forest and XGBoost, the hybrid ensemble improves the robustness and stability of malware classification, while the Artificial Neural Network (ANN) serves as an independent model for performance comparison. The proposed hybrid ensemble is computationally efficient because it relies on static Portable Executable (PE) header features, eliminating the need to execute suspicious files in sandboxed environments. Although combining two classifiers introduces a small computational overhead compared to using a single model, the soft-voting strategy requires only probability aggregation, making the overall prediction process suitable for practical malware detection applications.

3.4 Implementation and Data Flow

The implementation process starts with the PE header extraction of the executable files. After that, these features are transformed into numerical feature vectors, standardized, and passed to the machine learning models as input data for further classification. These standardized feature vectors are then processed in parallel by the Random Forest and eXtreme Gradient Boosting (XGBoost) classifiers. The prediction probabilities generated by both classifiers are combined using a soft-voting ensemble strategy to obtain the final malware classification decision. The Artificial Neural Network (ANN) model is trained and evaluated separately to compare its performance with that of the hybrid ensemble. To facilitate better interpretability of the results, SHapley Additive exPlanations (SHAP) are performed on the final prediction. Using SHAP, it becomes possible to identify feature contributions and detect the most influential PE attributes in the malware classification decision process. Using the aggregated prediction of the ensemble, it is possible to classify each executable as malware or non-malware.

4. Results and Discussion

The effectiveness of the suggested framework was examined based on the performance of the framework using the test part of the dataset. The following measures were used for performance evaluation: Accuracy, Precision, Recall, F1-Score, and Area Under the Curve (AUC). Overall, the performance of the hybrid ensemble framework was quite good, as the achieved accuracy was 99.13% while the Area Under the Curve (AUC) score was as high as 0.9984. The Random Forest and XGBoost classifiers also produced performance metrics comparable to those of the hybrid ensemble. This similarity indicates that the selected Portable Executable (PE) header features provide strong discriminatory capability for malware classification. The hybrid ensemble combines the prediction probabilities of both classifiers to improve the robustness and stability of the prediction process, even though the overall evaluation metrics remain similar. The ROC curve of the hybrid ensemble framework is shown in Fig. 1. It is worth noting that

the proposed framework's ROC curve does not diverge much from the upper left part of the graph, thus having very good discrimination capabilities. Moreover, the area under the proposed framework's curve is 0.9984.

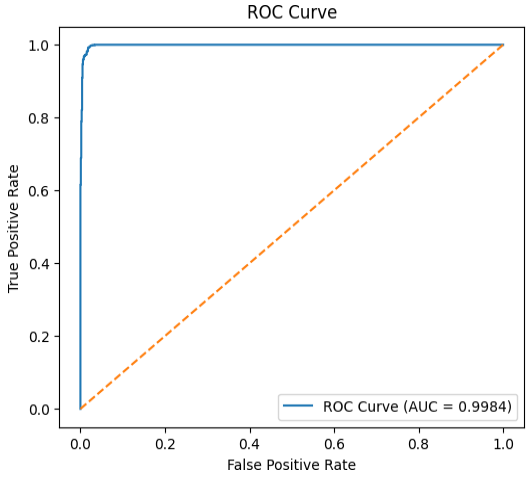

Figure 1: Receiver Operating Characteristic (ROC) curve of the proposed hybrid malware detection framework.

The confusion matrix derived as a result of testing is listed below:

- *True Positive (TP): 2917*
- *True Negative (TN): 972*
- *False Positive (FP): 31*
- *False Negative (FN): 3*

Figure 2 depicts the confusion matrix resulting from testing of the proposed framework. In total, the framework has successfully classified *2917 samples as malicious and 972 samples as benign*, while only making 31 false positive and 3 false negative predictions. Such a low number of errors shows the effectiveness of using the hybrid ensemble technique.

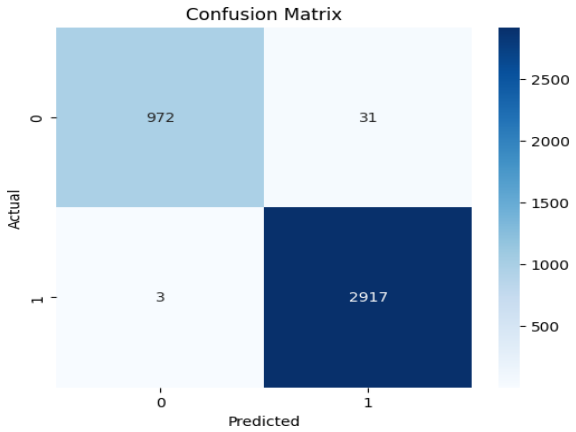

Figure 2: The confusion matrix for the hybrid malware detection system

Out of all the malware, only three samples were misclassified as benign programs. The low number of false negatives is especially critical when implementing a solution in the realm of cybersecurity because malware that was not detected could become a potential danger. The importance of these features in relation to their contribution to malware detection decisions was analyzed, and TimeDateStamp, MajorSubsystemVersion, Characteristics, MajorLinkerVersion, and ImageBase were some of the significant PE header characteristics for this purpose. Table 2 depicts the summary of the models' performance.

Table 2: Model Performance Metrics Summary

Model	Accuracy	Precision	Recall	F1-Score
Hybrid Ensemble	99.13%	0.9895	0.9990	0.9942
Random Forest	99.13%	0.9895	0.9990	0.9942
XGBoost	99.13%	0.9895	0.9990	0.9942
Artificial Neural Network	98.87%	0.9871	0.9976	0.9923

To further evaluate the effectiveness of the proposed framework, its performance was compared with representative malware detection approaches reported in the literature. The proposed framework achieves competitive classification performance while providing interpretable predictions through SHAP-based explainability. Unlike several existing approaches that rely on individual classifiers or computationally intensive dynamic analysis, the proposed framework combines hybrid machine learning with static Portable Executable (PE) header analysis for efficient malware detection.

Table 3: Comparison of the Proposed Framework with Existing Malware Detection Approaches

Study	Technique	Analysis Type	Key Limitation
Hasan and Dhakal [10]	Memory Analysis	Dynamic	High computational overhead due to run-time analysis.
Alam et al. [12]	ADAPT (Pseudo-labeling)	Static	Primarily focuses on concept drift without model explainability.
Vijayalakshmi et. al. [4]	Individual Machine Learning Models	Static	Evaluates individual classifiers without hybrid ensemble learning.
Alshoulie and Mehmood [9]	Deep Learning	Static/Dynamic	Computationally expensive and less interpretable.
Kumar et. al. [5]	Machine Learning Classification	Static	Uses individual machine learning mod-els without explainable AI.
Proposed Framework	Hybrid Ensemble (Random Forest + XGBoost) with SHAP	Static PE Header	Provides interpretable malware detection using a hybrid ensemble with low computational overhead.

The comparison presented in Table 3 demonstrates that existing approaches primarily focus on either individual machine learning models, deep learning techniques, or dynamic malware analysis. In contrast, the proposed framework combines static Portable Executable (PE) header analysis with a hybrid ensemble of Random Forest and XGBoost classifiers while incorporating SHAP-based explainability. This combination provides accurate, computationally efficient, and interpretable malware detection, making it suitable for practical cybersecurity applications.

Finally, Figure 3 shows the importance of each of the PE header features considered in the framework of malware detection.

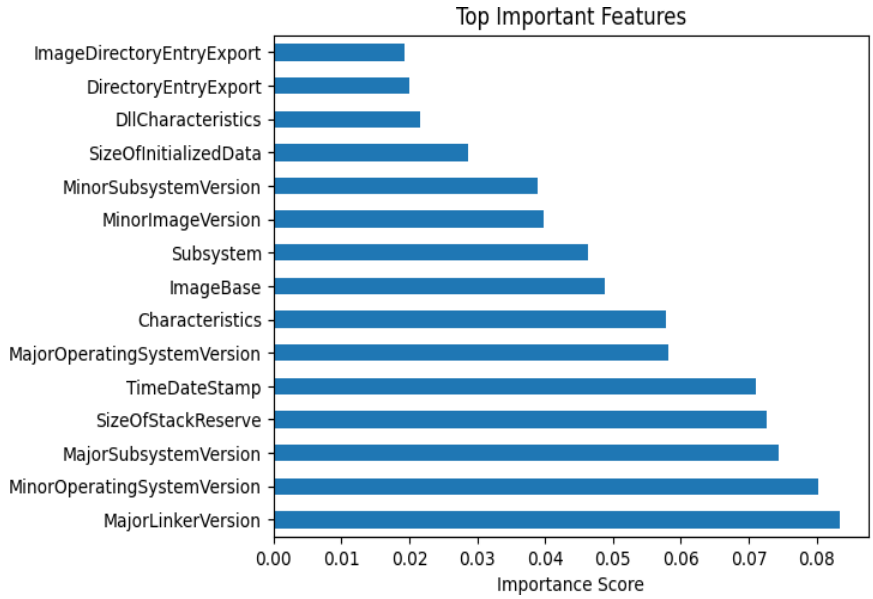

Figure 3: Significant Portable Executable (PE) header attributes in terms of malware detection

4.1 SHapley Additive exPlanations (SHAP) Analysis

To enhance the interpretability of the proposed algorithm, the use of SHapley Additive exPlanations (SHAP) was implemented during the process of malware detection. The SHAP values allow one to determine the contribution of specific features to the classification and give an explanation for the prediction of malware. According to the results of the SHAP analysis, it is clear that some of the attributes of the PE header significantly impacted the outcome of classification. SHAP values greater than zero positively affected the classification process leading to malware, while SHAP values less than zero had a positive effect on classification as benign. The implementation of SHAP enables cybersecurity specialists to easily comprehend the rationale behind model predictions and, therefore, be confident about the operation of automatic malware detection systems.

4.2 Explainable Artificial Intelligence and Structural Analysis

Although machine learning models achieve high prediction accuracy, understanding the reasoning behind their decisions is equally important in cybersecurity applications. Therefore, the proposed framework incorporates SHapley Additive exPlanations (SHAP) to improve the transparency and interpretability of malware classification. SHAP quantifies the contribution of individual Portable Executable (PE) header attributes toward the final prediction. The analysis identified TimeDateStamp, MajorSubsystemVersion, Characteristics, MajorLinkerVersion, and ImageBase as the most influential features. Positive SHAP values increase the likelihood of malware classification, whereas negative values indicate benign predictions. Figure 4 represents the SHAP summary plot illustrating the contribution of the most significant PE-header attributes to malware classification.

The integration of SHAP improves the interpretability of the proposed framework by providing feature-level explanations for every prediction, thereby increasing transparency, supporting security investigations, and improving user confidence in automated malware detection. Combined with the hybrid machine learning framework, this enables accurate and explainable malware classification suitable for practical cybersecurity applications.

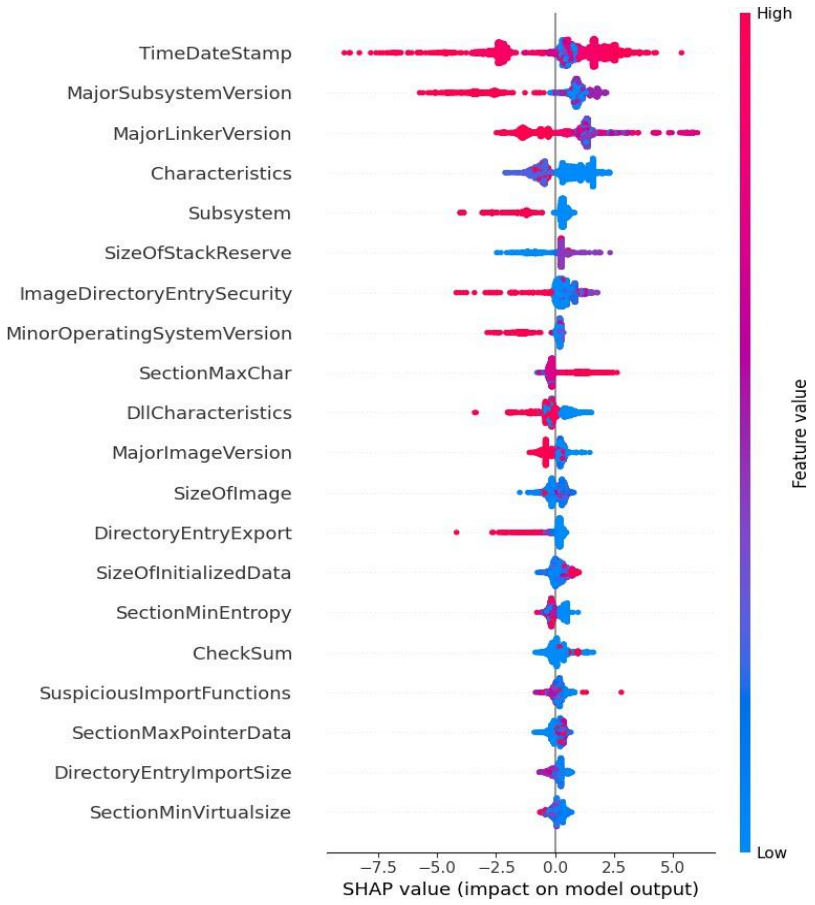

Figure 4: SHAP summary plot illustrating the contribution of Portable Executable (PE) header attributes toward malware classification decisions.

5. Conclusion and Future Work

In this work, an adaptive malware detection system that uses static analysis based on Portable Executable (PE) headers and artificial intelligence has been designed. This AI-based adaptive malware detection framework incorporates static PE-header information with a hybrid ensemble consisting of Random Forest and eXtreme Gradient Boosting (XGBoost) classifiers, while the Artificial Neural Network (ANN) model is evaluated independently for performance comparison. In other words, by using static analysis methods, efficient malware detection is possible while saving most of the costs of dynamic analyses.

For evaluation purposes, the proposed model was tested using the dataset including 19,611 executable samples and 79 attributes related to Portable Executable (PE) headers. According to the experimental results, an accuracy of 99.13% and an area under the curve score of 0.9984 were achieved. Furthermore, the confusion matrix analysis demonstrated 2917 true positives, 972 true negatives, 31 false positives, and 3 false negatives. As a result, it can be claimed that the proposed framework is effective against adaptive malware detection.

To increase the levels of interpretability and transparency, SHapley Additive exPla-nations (SHAP) were implemented into the proposed framework. As a result of the explainability analysis, the attributes from PE headers that have had the largest impact on malware classification have been identified. With the help of Explainable Artificial Intelligence (XAI), the performance of machine learning models can be analyzed better and, thus, increase the overall effectiveness of the solution. In general, the obtained results show that the use of Portable Executable (PE) headers analysis in combination with the techniques of hybrid machine learning models is an effective way to detect new malware variants. In addition to the high level of predictive power, the framework demonstrates good interpretability as well.

For future research, the implementation of the framework will include the analysis of dynamic malware behavior and behavioral and network activity features. Moreover, further investigations can cover such areas as the application of deep learning architectures, concept drift approaches, and real-time detection of malware for enterprise-level environments.

Funding

This work is not funded by any agency.

Conflict of Interest

- Authors declare that they have no conflict of interest.
- Consent to participate: All the authors involved have agreed to participate in this submitted article.
- Consent to Publish: All the authors involved in this manuscript give full consent for publication of this submitted article.

References

1. M. G. Schultz, E. Eskin, F. Zadok, and S. J. Stolfo, “Data Mining Methods for Detection of New Malicious Executables,” in Proc. IEEE Symposium on Security and Privacy, 2001, pp. 38–49.
2. A. F. J. Alshammari, “Explainable AI-Based Malware Detection for Protecting Assistive Learning Systems Used by Students with Special Needs,” International Journal of Special Education, vol. 41, no. 8s, pp. 843–858, Jun. 2026.
3. R. Malge, S. Swamy, and S. Gupta, “Real-Time PM2.5 Forecasting Using Lightweight Ensemble Learning and Temporal Feature Fusion,” Journal of Intelligent Decision Making and Information Science, vol. 3, no. 3s, pp. 1894–1917, 2026.
4. V. K., E. B. Subhashini, S. Al Sabahi, R. Al Shanawi, and T. Al Harthy, “Comparative Evaluation of Machine Learning Algorithms for Efficient Malware Detection,” International Research Journal of Multidisciplinary Technovation, vol. 7, no. 6, pp. 266–282, 2025.
5. S. Kumar, Shersingh, S. Kumar, and K. Verma, “Malware Classification Using Machine Learning Models,” 2024.
6. T. Chen and C. Guestrin, “XGBoost: A Scalable Tree Boosting System,” in Proc. 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 785–794.
7. I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA, USA: MIT Press, 2016.
8. A. Bensaoud, J. Kalita, and M. Bensaoud, “A Survey of Malware Detection Using Deep Learning,” Machine Learning with Applications, vol. 16, p. 100546, 2024.
9. E. A. Elhadidi and A. Salah, “Quantum Artificial Intelligence: A Comprehensive Review of Architectures, Applications, Challenges, and Future Directions,” International Journal of Computational Intelligence in Engineering (IJCIE), vol. 1, no. 2, pp. 35–41, Apr. 2026.
10. S. M. R. Hasan and A. Dhakal, “Obfuscated Malware Detection: Investigating Real-World Scenarios Through Memory Analysis,” in 2023 IEEE International Conference on Telecommunications and Photonics (ICTP), 2023, pp. 1–5.
11. D. Gibert, N. Totosis, C. Patsakis, Q. Le, and G. Zizzo, “Assessing the Impact of Packing on Static Machine Learning-Based Malware Detection and Classification Systems,” Computers & Security, vol. 156, p. 104495, 2025.
12. M. T. Alam, A. Piplai, and N. Rastogi, “ADAPT: A Pseudo-labeling Approach to Combat Concept Drift in Malware Detection,” arXiv preprint arXiv:2507.08597, 2025.
13. S. M. Lundberg and S. I. Lee, “A Unified Approach to Interpreting Model Predictions,” in Advances in Neural Information Processing Systems, 2017, pp. 4765–4774.
14. M. Saadoon and S. Faisal, “Malware Detection Using Machine Learning Techniques: A Review,” Basrah Journal of Science, vol. 42, no. 2, pp. 219–247, 2024.