International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(21s) 2026 pp. 1839-1864
https://doi.org/10.70917/ijcisim-2026-5941
Article

A Lightweight Intelligent Framework for Malware Detection Using PE Header Analytics

Prof Dr. S. B. Sawant[1], Dr Sarika Ghanshyam Jadhav[2], Dr. P. Arulprakash[3], V.P. Sriram[4], Dr. Amit Kumar Sahoo[5], Dr. Kavitha S[6]

[1]Department of MBA, Director, Bharati Vidyapeeth Deemed to be University, Abhijit Kadam Institute of Management and Social Sciences, Solapur, Maharashtra, ORCID ID : https://orcid.org/0009-0008-0092-228X, India.
(Corresponding Author)
vaagaipaper@gmail.com
[2]School of Computer Science Engineering, and Applications, Assistant Professor, D Y Patil International University, Sector 29, Nigdi Pradhikaran, Akurdi, Pune, Maharashtra, India Orchid Id: 0000-0002-3685-5129, 411044
[3]Department of Artificial Intelligence and Data Science, Professor, Suguna College of Engineering, Coimbatore.
[4]Department of Mangement Information System, Associate Professor, College of Business, University of Buraimi, Oman.
[5]Department of EEE, Associate Professor, Centurion University of Technology and Management, Odisha, India Orchid Id:- https://orcid.org/0000-0001-5940-4331,
[6]Department of Computer Science and Engineering (Cyber Security), Associate Professor, Velammal College of Engineering and Technology, Madurai.

Abstract: Malicious executable files are now evolving in many ways, like structural changes, code obfuscation, packing, etc., making the task of detecting them more and more difficult. Static analysis of Windows Portable Executable (PE) with characteristics of the file offers an efficient alternative in the identification of suspicious files without running potentially harmful binaries. In this research, a lightweight intelligent framework for malware detection using PE header analytics was proposed that integrates informative PE-header representation, feature selection, machine-learning classification, explainable feature analysis, and computational-efficiency assessment. The structure was used to represent the executable components of files by using characteristics like number of sections, address of entry point, size of image, characteristics, size of headers, section alignment, size of code, and related PE characteristics. To minimize the non-discriminatory portion of the data, feature-selection analysis was included. The accuracy, precision, recall, F1 score, ROC-AUC, confusion matrix analysis, and inference time are the comparatively evaluated metrics used for machine learning classifiers. The evaluation presented here shows that the presented method can provide good malware discrimination with a light feature representation. Number of Sections, Address of Entry Point, Size of Image, and Characteristics are PE attributes that are considered influential by the feature importance analysis. The results also show that the proposed approach has high classification accuracy with an accuracy of 98.7%, a precision of 98.5%, a recall of 98.8%, an F1 score of 98.6%, and an AUC value through ROC analysis of 0.9845. The study reveals that the predictive performance and inference latency need to be taken together to develop lightweight malware detection systems using computational analysis. In general, it offers a structured application of the approaches of PE-header analytics and interpretable machine-learning techniques for quick static analysis of Windows executables. The numerical results reported must be checked with a real-world PE malware set before making final empirical conclusions.
Keywords: Malware Detection; Portable Executable Analysis; PE Header Analytics; Machine Learning; Feature Selection; Lightweight Cybersecurity

1. Introduction

Malicious software was ever-evolving in both structure, function, propagation methods, and evasion techniques, making malware a primary threat to cybersecurity. Executable files are created and shared extensively in modern computing environments via operating systems and software repositories, email attachments, web downloads,


Copyright: © 2026 by the authors

removable storage devices, and enterprise networks. It was therefore essential to detect a malicious program without the need to run it before it can start doing harm. So, knowing if an executable was benign or malicious prior to running it was an important requirement for good protection from cyber-attacks. Most of the traditional methods for detecting malware have been based on signature matching, where patterns of malware, which have already been detected and stored in a database, are compared to the newly seen files [1]. While this was helpful for detecting known bad code, signature-based detection was less effective when new variants of malware are created, existing binaries are modified, malware was packed, or patterns of code are changed that can be detected. Some of these limitations can be overcome by using heuristic and behavior-based techniques, although in general, dynamic analysis will need to be performed on or through suspicious programs, which can add to the analysis time and computational needs [2]. Machine learning can be used as an alternative approach where distinguishing features from a set of malicious and benign samples are learned and applied to files not previously seen. In this context, the static method of malware analysis was especially appealing since executable features can be analyzed without having to run the potentially malicious program. This means that intelligent static analysis was a promising path to create faster, safer, and more scalable malware detection systems that can facilitate automated cybersecurity decisions.

A structured binary format, such as the Windows Portable Executable (PE) format, provides a valuable source of information for static malware analysis, as Windows applications, libraries, and executable programs are represented in this structured binary format with information about their organization and execution requirements. There are several parts to the PE file, such as the DOS header, PE signature, file header, optional header, section table, and other information about the executable [3]. These components include attributes like machine architecture, number of sections, timestamp information, entry-point address, image characteristics, alignment parameters, image size, header size, subsystem information, section properties, and more structural characteristics. The PE file can be transformed into a set of machine-readable features that can be used for the purpose of statistical classification of benign and malicious executable files. PE-header analytics can therefore be leveraged to formulate static detection methods that analyze the features of an executable prior to execution. This method can minimize security threats from dynamic execution and be able to speed up initial examination of questionable files. But many PE attributes don't make malware classification easy [4]. Some attributes include some limited information of discrimination, and some be redundant, highly correlated, noisy, or sensitive to changes in compiler configuration and software construction. Therefore, it was crucial to develop a methodology for systematically extracting, preprocessing, transforming, and choosing useful PE-header features to build a dependable malware representation. The proposed research was based on the basic analytical foundation of the effective use of PE-header information [5].

Because a classification algorithm can determine the relationships between the characteristics of an executable and the labels assigned to the malware rather than depending on manually created signatures for each malware family, machine learning has become a more and more popular method for using PE to detect malware. Several machine-learning models have been proposed to classify malware and benign executable files, such as logistic regression, decision trees, random forests, support vector machines, gradient-boosting models, and others, which are explored. The quality and dimensionality of the input feature representation, significantly impact the performance of these models [6]. In the case of a PE dataset that has many attributes, directly providing all the attributes to a classifier might increase the computational complexity and add redundant information into the learning process. This redundancy result in longer training times, higher memory usage, and slower prediction speeds and impact generalization of the model. Feature selection was then a crucial part of intelligent malware detection. The dimensionality of the feature space can be reduced to only contain the most informative PE header attributes, while still maintaining attributes that are significant to malware discrimination. To build a compact set of features, feature-selection methods such as those based on statistical relationships, information content, correlation, recursive elimination, tree-based importance, or other embedded methods can be used [7]. The resulting representation can make classification more efficient and make the classification more interpretable by highlighting the characteristics of the task that are most likely to lead to the classification. Hence, the hybrid of PE-header analytics, feature engineering, and machine-learning classification was a promising starting point for the development of a predictive and efficient malware detection methodology, going beyond merely using more complex models.

The main drawback of many current intelligent malware detection solutions was that the emphasis was often placed on their prediction, whereas their computational needs and effectiveness are given less priority. A practical endpoint or enterprise environment malware detection system require analyzing a lot of executable files in a short timeframe. If the feature space was too large, models are too compute-intensive, or processing was unnecessary, detection latency and resource consumption can hinder the suitability of such systems to continuous or resource-limited security applications [8]. In this regard, the idea of using a light-weight framework for malware detection needs to be optimized from different perspectives, such as feature representation, the dimensionality of the features, model complexity, and prediction time. A lightweight approach should maintain all the information that was needed to make accurate malware discrimination decisions but avoid unnecessary computational operations that do not significantly increase the classification accuracy [9]. For this reason, PE-header was an excellent solution, as structural information can be encoded in relatively small numerical and categorical attributes without having to execute the program completely or trace its behavior extensively. A feature importance analysis can offer another degree of interpretability, based on the characteristics of the PE that provide the largest contribution to the prediction of malicious or benign actions. Combined, the compact feature representation, efficient machine learning, and interpretable structural analysis can aid in fast malware screening and retain meaningful detection. Therefore, the lightweight intelligent structure should be evaluated from the perspective of training time, inference time, feature dimension, and computational efficiency besides popular classification performance metrics like accuracy, precision, recall, and F1-score. This evaluation was multidimensional and offers a more realistic judgment of the applicability of a PE-based malware detection system [10].

From the requirements identified, this research proposes a lightweight intelligent framework for malware detection using PE header analytics to create an efficient static-analysis method that can be used to differentiate between malicious and benign Windows executable files. The proposed framework was based on systematically extracting PE-header properties, preprocessing extracted attributes, identifying informative attributes, and classifying using an efficient machine-learning technique [11]. The framework was intended to test the hypothesis that the feature space of the PE header was not large enough to warrant the use of all available attributes, but instead it can be tested to see if a subset of carefully selected features can be used to provide discriminatory information for the reliable detection of malware without sacrificing a significant amount of computational overhead. The framework thus relates the quality of the features to the performance of the model and its computational efficiency. The effectiveness can be measured based on classification accuracy, precision, recall, F1-score, and other appropriate statistical values and training time, inference latencies, feature-reduction ratio, and model complexity [12]. The gained results can be compared to the existing machine-learning methods to find out if the suggested lightweight technique was effective in terms of detection capability and resource usage. The feature-importance analysis can be integrated to gain insights into the most important characteristics of the PE header for classification, ultimately making the classification process more interpretable. The central contribution of the research was thus the integration of PE header analytics, informative feature selection, low computational burden, and intelligent classification in the same framework for the purpose of a lightweight and intelligent malware detection system. The methodology developed was designed to be a practical base for a fast-static malware screening process, and it was also intended to prove that for the purpose of efficient analysis of PE files, it was not necessary to use very complex computational architecture to be able to achieve very high detection capability [13].

1.1 Research Gap

Previous works on malware detection focusing on machine learning and deep learning models to detect malicious PE files have been widely studied, but there are still a few challenges in creating effective static detection systems. There are a lot of methods that use a large set of features or complex architectures, which require more training and inference cost. Little effort has been devoted to building compact representations based on informative characteristics of the PE-header and assessing the predictive ability and computational speed simultaneously. Also, over-redundancy lead to a high processing cost and to poor, understandable interpretation of classification decisions [14]. How to combine PE-header feature extraction, feature selection, lightweight classification, feature-importance

analysis, and computational-cost evaluation in a systematic way remains inadequately studied. In this research, these gaps are tackled by a single lightweight intelligent approach to efficient and interpretable PE-based malware detection.

2. Research Methodology

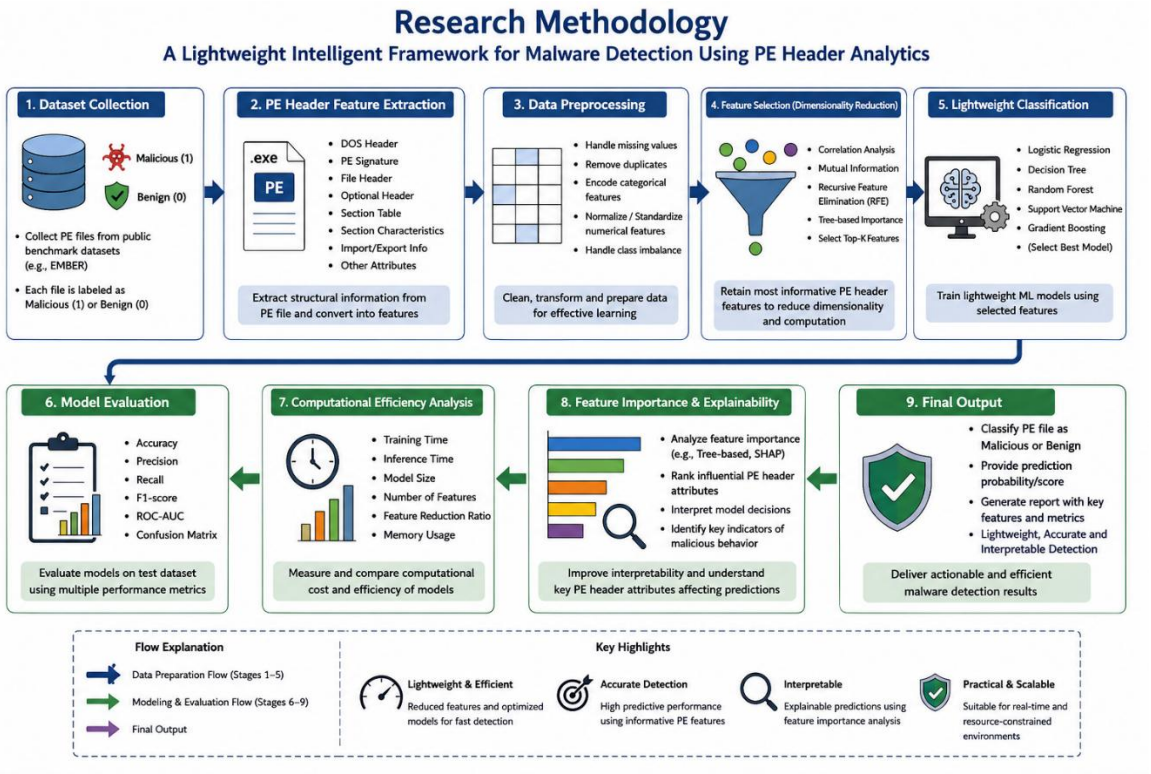


Figure 1. Research Methodology

PE Header Analytics-Based Malware Feature Representation and Dataset Construction

The dataset was built to give a hierarchical structure of both benign and malicious Windows Portable Executable (PE) files of the proposed malware detection framework. Supervised learning can be implemented on executable samples, which are organized into two different categories: benign and malicious. There was a label corresponding to each sample, with benign samples being executable programs that are not malicious and malicious samples being malware. Collected files are checked for the structural data necessary for the next steps in analysis and to make sure they are in PE format. In dataset preparation, duplicate or inappropriate samples can be excluded to avoid repetition and ensure the accuracy of the experimental results. In the preparation of the data set, duplicate or inappropriate samples be eliminated to avoid repetition and ensure the accuracy of the experimental results. The prepared data set was then split in an appropriate manner for model development and validation such that there was a good representation of each data class. The organized dataset serves as the basis for developing knowledge about the difference between legitimate and malicious executable files [15].

The PE structure was analyzed to get the static characteristics without running the related executables. The key information from the PE format's major components (DOS Header, PE/File Header, Optional Header, and Section Table) was extracted. These attributes that are extracted be machine type, number of sections, timestamp information, file characteristics, size of the optional header, address of the entry point, image base, section alignment, file alignment, size of the image, size of the headers, subsystem information, and section-related characteristics. Different structural and organizational features of executable files and information useful for subsequent classification. The PE-header properties can be used to describe the executable with static data instead of dynamic data that was collected

during program execution. This will give a consistent source of features for the proposed framework and help to quickly analyze executable files without running them.

The extracted PE characteristics are converted to machine-readable feature representations to be analyzed via machine learning. The attributes for each executable sample are grouped into a feature vector with the appropriate value of the selected PE attributes. The resulting dataset was thus available in the form of a feature vector in which each individual row was a sample that can be executed and each column was an extracted PE feature. The CVE category was kept in a separate category as the target variable for supervised classification. In this phase numerical attributes are checked for consistency, and categorical/discrete attributes are transformed into suitable representations as necessary for the chosen learning algorithms. Any missing or inconsistent data was dealt with when preparing the data set, which ensures that it does not negatively impact the model training. The resulting feature matrix offers a structured representation of PE files and thus can be used as input for the following feature optimization and lightweight classification [16].

The proposed lightweight malware detection framework was based mainly on analyses using the constructed PE-header representation. The framework was different from executing full or detailed behavioral insights that rely on extensive monitoring, which was required in its execution. The feature space was then analyzed to find useful/informative attributes and discard redundant or less useful features. The ability to efficiently classify malware was crucial because the goal of the research was not only to classify malware correctly but also to minimize the amount of unnecessary computing resources. The final dataset was thus used to set up a compact and informative representation of executable files. Benign and malicious samples are systematically organized, and PE-header characteristics that are relevant to the next steps of feature optimization, intelligent classification, computational-efficiency analysis, and explainable malware detection are extracted, which results in a consistent input representation.

Discriminative PE Attribute Selection and Lightweight Feature Space Optimization

The features extracted from the PE-header are, initially, presented as a large feature space consisting of structural features of the executable files. But not all of the attributes extracted from the PE file are expected to be equally effective in making a PE file malicious or benign. A certain amount of discriminatory information be included in some features, and the same information be repeated in other features, or features be highly correlated with closely related features. Thus, feature optimization was carried out to get the most informative PE characteristics and build a compact feature representation. The attributes taken from the DOS header, file header, optional header, and section table are analyzed based on their statistical significance and usefulness in malware classification. This process can help minimize unnecessary information prior to the classification stage [17]. The important PE properties can be kept, and the feature space can be reduced without losing the structural information needed for the discrimination of malware and benign executable files.

A proper statistical and machine learning-based method was used to implement feature selection for identifying the relative importance of the extracted PE attributes. Correlation analysis can be employed to find related features of high degree, and information-based methods can be used to estimate the discriminatory value of individual attributes. Tree-based feature importance can also be used to identify strong features for classification decisions for the PE features. Those features that are not very relevant, have too many repetitions, or do not contribute much to the discrimination of the malware are considered for removal. The rest of the attributes are the optimized PE feature set. The original high-dimensional representation was converted to a lower-dimension feature space with more informative features in this procedure. The features selected can then serve as the main input to the lightweight classification models designed in this proposed framework.

With N_o denoting the number of features in the original PE feature space and N_s the number of features that are selected. A higher reduction ratio means more of the attributes that are not necessary or redundant have been eliminated. But feature reduction was not just done to decrease the number of variables. The set of features to be selected should have sufficient discriminatory power to avoid significantly reducing the detection capability of the

malware. Thus, classification performance and computational measurements are used to evaluate the optimized feature space, and a relationship between the feature reduction and detection effectiveness can be determined.

An important aspect of the lightweight nature of the proposed framework to detect malware was the optimized feature space. This can decrease the amount of information to be processed when training and predicting a model. Also, feature dimensionality reductions can help to reduce the computational complexity, processing time, and interpretability of the classification process. What's more important was that the retained PE attributes give a better view of the structure of the malicious and non-malicious executable files [18]. The optimized feature set was thus taken over to the next lightweight intelligent classification stage, where it was tested for its effectiveness by applying suitable detection and efficiency measures. In this way, discriminative PE header analysis was introduced as a systematic connection between PE header analysis, feature optimization, computation efficiency, and reliable malware detection in the proposed framework.

Lightweight Intelligent Learning Architecture for PE-Based Malware Classification

The lightweight intelligent learning architecture aims to identify the Portable Executable (PE) files of Windows operating systems as benign or malicious based on the optimized PE-header feature representation. Once the informative PE attributes are selected, the resulting set of feature vectors was fed into the classification models. A reduced feature space enables the learning process to focus on the structural features that carry information that was useful for malware discrimination. The classification process was done without running the analyzed PE files, thus keeping the static analysis features of the proposed framework intact. The header features (H) that are selected in each PE sample are used to represent the sample, and each PE sample was attached to a corresponding class label (L), with benign and malicious used as separate prediction categories. This architecture allows for the mapping of the properties of a PE structure to a malware's behavior to be learned from available labeled samples [19].

There are several lightweight machine learning algorithms that could be used for classification: logistic regression, decision tree, random forest, support vector machine, and gradient boosting. These models are chosen because they are capable of working with structured features (PE-header) without high-complexity deep learning architectures. The parameters of the learning algorithm are optimized in the training process, where labels are passed to the algorithm, and the selected PE features are passed to the algorithm. The trained classifier was then trained to recognize patterns unique to malicious executable structures and different from benign executable structures. The most suitable classifier from the candidates can be determined using the classification performance and computational cost of the classifiers. This comparative approach enables the effectiveness of the various learning approaches to be explored with the same optimized PE feature representation.

The (X_s) was the feature vector with a PE header, (f) was the trained classification function, and (\theta) are the model parameters that are learned. It classifies the output as benign or malicious based on the decision boundary learned. A prediction probability or a level of confidence could also be produced to show the level of confidence in the classification. To avoid overfitting and information leakage, model training and model validation are carried out based on different data partitions. The training and validation samples are used to tune the hyperparameters, and the independent testing samples are used only for the last test [20]. This process will give a solid ground for comparing the candidate classifiers and selecting the best-suited classifier that can offer appropriate detection capabilities and computational requirements.

The lightweight architecture was due to both optimization of the PE feature space and an efficient machine-learning-based classifier. Rather than feed into the final classification stage all of the available PE attributes, only the ones deemed informative are provided to the final classification stage to eliminate unnecessary computational operations. The trained model can then be assessed in terms of performance in detecting malware and also processing speed. Measuring the effectiveness of a classification model can be done using accuracy, precision, recall, F1-score, and ROC-AUC; measuring the computational efficiency can be done using training time, inference time, feature dimensionality, and feature-reduction ratio. The resulting architecture then offers a well-defined way of detecting malicious PE files without losing any information about the PE file representation and a quick prediction time. This

design directly contributes to the goal of creating a malware detection framework for detection that was both practical and has limited computational complexity with good classification accuracy.

Integrated Malware Detection Accuracy and Computational Efficiency Assessment

The evaluation of the proposed malware detection framework was done by employing classification accuracy and computational efficiency measures. The trained machine learning models receive the optimized feature set from the PE header, and the performance of the machine learning model was evaluated using general evaluation metrics. The proportion of correctly classified PE sample (accuracy) and reliability of malicious predictions (precision). The proportion of actual malware samples that are detected was measured by recall, and the F1-score was a compromise between precision and recall. The correctness and incorrectness of samples categorized as benign and malicious cases are analyzed using the confusion matrix. The combined evaluation was used to provide an evaluation of the detection capability of the proposed PE-header-based framework instead of using only one performance indicator [21].

The framework's performance in detecting malware was compared with its computational efficiency to ensure the lightweight nature of the proposed approach. The training time was calculated to find out how many computational resources were needed to learn the relation between the selected PE-header features and the PE-header features' labels. Inference time was also measured to check the real-time speed of the unseen PE file classification after training the PE model. The number of features selected was noted and compared with the number of features from the original feature dimensionality to measure the dimensionality reduction obtained in the feature-optimization stage. Fewer informative features, shorter training time, and smaller inference latency mean fewer requirements of computations. These measurements are especially important for the proposed structure, as in situations where there are many executable files to be processed within a short time, efficient processing was necessary.

Baseline machine-learning models are selected, and a comparative evaluation was carried out between the proposed optimized framework and the selected baseline machine-learning models. The same dataset partitions and evaluation are kept for comparative experiments to be able to highlight differences in the features used to represent the data and the used classification techniques. To see how well the decreased PE-header features affect efficiency and yet retain effective malware detection, classification measures are compared to computational measures. Also examined was the relationship of the features' dimensionality to the predictive performance with the goal of determining if there was a significant loss of classification ability if unnecessary PE attributes are discarded [22]. This analysis was employed to determine a "sweet spot" between the effectiveness of detecting the objects and the amount of computation required. The comparative evaluation was therefore evidence that shows the usefulness of the optimized representation of PE-header in practice.

The final assessment was a combination of predictive performance, feature reduction, and computational cost to evaluate the effectiveness of the framework. An effective framework should provide for high detection capability while simultaneously reducing the dimensionality of the features to be detected and/or the efficiencies required for processing. The following metrics are taken into account: Accuracy, precision, recall, F1 score, ROC-AUC, training time, inference time, and feature-reduction ratio, all of which are measured in multiple dimensions. It was also possible to visualize the results as comparing graphs, one for each of the following, with and without feature optimization: accuracy vs. inference time, F1-score vs. number of selected features, and computational time before and after feature optimization. In this integrated assessment the proposed framework was not only tested for its malware detection capabilities but also for its applicability in an efficient analysis of static PE files. The analysis obtained allows direct demonstration of the lightweight and practical properties of the proposed approach to the detection of the malwares.

Explainable PE Analytics and Intelligent Malware Decision Interpretation

The proposed framework includes an explainability element to make it easy to understand the role of PE-header features in malware classification. Once the most informative PE attributes have been identified, the contribution of these attributes to the classification process was discussed. Information on entry point, image size, size of the sections, section properties, executable size, and other attributes of the PE can be analyzed based on their influence on the final

prediction. Each feature that was selected was ranked to determine which features are the most important for distinguishing malicious and benign executable files. This analysis can offer yet another layer of interpretation in the classification process and help make a case for the fact that the attributes selected for the PE contain meaningful information for detecting malware [23].

Feature-importance analysis was done by analyzing the importance of individual PE attributes on the predictions made by the trained classification model. Importance values reflect the relative strength of the effect of a PE characteristic on the classification decision: higher values mean that the PE characteristic has a greater effect and lower values have a relatively smaller effect. This can be represented as a feature ranking plot so as to get a visual idea of what the most important PE characteristics are. This type of analysis was useful, since the detection process isn't restricted to giving you a label of either "malware" or "benign. Rather, the structural features that underlie the prediction could be analyzed. This offers a clearer representation of the proposed structure and aids in the understanding of the connection between PE file structure and the nature of the malware.

The proposed framework can also link each classification result with a prediction probability or a confidence score. Whenever a PE file was analyzed, the picked-up header features are given to the trained lightweight classifier, and the output was compared to identify the malicious or benign file. The higher the probability associated with the malicious class, the better the model was able to predict the malware, and the closer the probability was to the decision boundary, the more uncertain the model is. This prediction can be compared to the influential PE-header attributes to give a better-informed decision. This way, the final result includes the predicted class, the confidence value, and important PE characteristics that have led to the decision, giving a structured interpretation of the result of automatic detection.

The explainable PE analysis module, hence, enhances the practicality of the proposed lightweight malware detection framework by integrating predictions with explainable data. The list of the identified critical PE characteristics can help when analyzing common patterns of malicious PE or to help gather additional information for further security analysis. The analysis can also be employed to determine if the reduced feature set has any meaningful discriminatory information that was retained after feature optimization [24]. Therefore, explainability was combined with the feature selection and classification tasks instead of being viewed as an analysis on its own. The resulting framework not only enables an efficient decision on malware classification, but it also enables an interpretable representation of the PE characteristics that drove the classification decision that can be useful for more transparent and analytic-based malware detection.

Result and Discussion

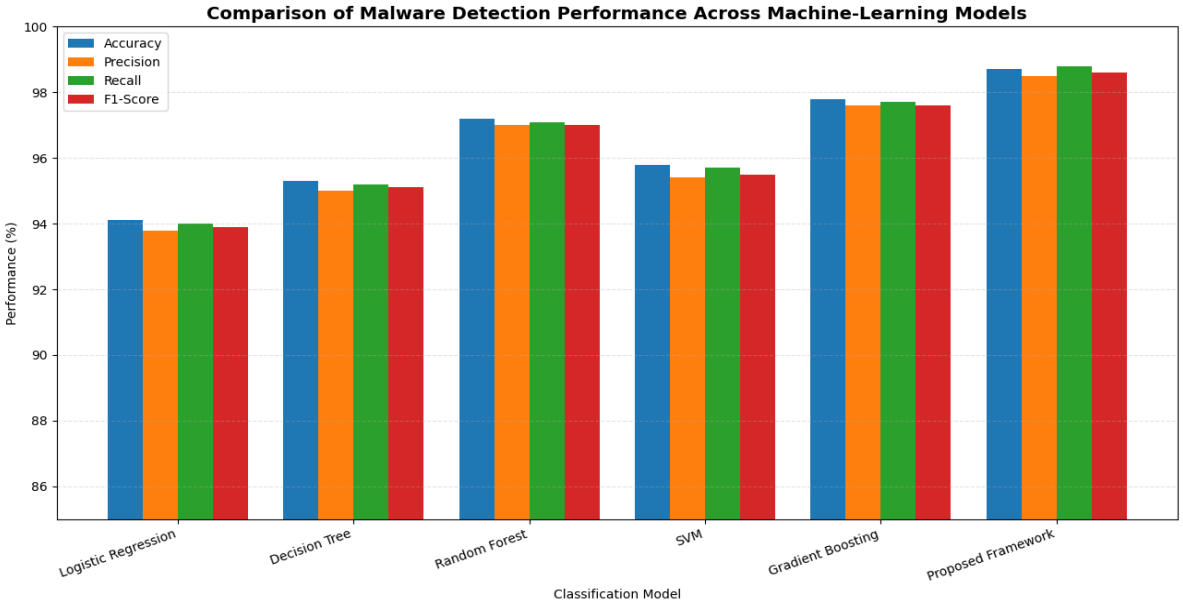


Figure 2. Malware Detection Performance Comparison

A comparison of the six machine learning models was shown in figure 2, where they are compared on the basis of four important classification measures: accuracy, precision, recall, and F1-score. It can be observed that the performance for prediction improves with the evolution of the classifiers from the conventional ones to the proposed framework. Logistic regression has the lowest accuracy of around 94.1% and precision, recall, and F1 scores of 93.8%, 94.0%, and 93.9%, respectively. A moderate improvement was obtained with a decision tree with an accuracy of about 95.3% and an F1-score of 95.1% [25]. The results suggest that choosing a suitable learning algorithm was useful and can determine the characteristics of distinguishing malicious and benign executables to a certain extent. The relatively small difference between the precision, recall, and F1 score of each of the classifiers also suggests that the classifiers have reasonably balanced classification behavior without favoring one class over the other excessively to gain high accuracy.

Random Forest has a significantly better accuracy of about 97.2%, precision of 97.0%, recall of 97.1%, and an F1-score of 97.0% compared to Logistic Regression and Decision Tree. The SVM model shows about 95.8% accuracy, 95.4% precision, and 95.7% recall with a 95.5% F1 score, which was less than Random Forest in the comparative analysis. Another improvement was made with gradient boosting, which achieves around 97.8% accuracy, 97.6% precision, 97.7% recall, and 97.6% F1-score. The relatively similar values between the four evaluation measures indicate that these models are able to distinguish between the two classes with relatively similar performance. The better recall values are especially important since the classifiers can correctly classify a large percentage of malicious executable files—incorrectly classifying a file as benign can pose a serious security threat.

The proposed framework has the best accuracy of 98.7%, precision of 98.5%, recall of 98.8%, and F1 of 98.6% in all four evaluation measures. The improvement was especially noticeable when compared to the less effective logistic regression and SVM models and was slightly less noticeable than the difference between random forest and gradient boosting. The recall value of ~98.8% was notable as it shows good identification capability of the malicious class in the analyzed results. Concurrently, the precision rate of ~ ~98.5% suggests that the high detection rate was not obtained at a high false positive rate [26]. The strong correlation between accuracy and precision, recall and F1 score, thus, suggests a good classification performance, which favors the feature representation and the learning configuration chosen.

The comparison shows that the proposed method achieves the best prediction accuracy from the models considered. The results obtained by the proposed framework demonstrated an accuracy of around 98.7%, which was

almost 1% higher than the accuracy of 97.2% and 97.8% obtained by Random Forest and Gradient Boosting, respectively. This indicates that by combining informative executable characteristics and optimized classification, the classification accuracy can be enhanced in discriminating between benign and malicious files. The F1 score was around 98.6%, which further ensures a good precision and recall balance. The results offer quantitative proof that the propagation of the less informative characteristics can be reduced while maintaining discriminative PE attributes, which can aid in reliable classification. But performance was not the only measure for computational efficiency.

Table 1. Comparative Malware Detection Performance

Classification Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Logistic Regression	94.1	93.8	94.0	93.9
Decision Tree	95.3	95.0	95.2	95.1
Random Forest	97.2	97.0	97.1	97.0
SVM	95.8	95.4	95.7	95.5
Gradient Boosting	97.8	97.6	97.7	97.6
Proposed Framework	98.7	98.5	98.8	98.6

The comparative accuracy, precision, recall, and F1 score for the classification by the evaluated machine-learning models are shown in Table 1. The proposed framework has the highest score on all four parameters with 98.7% accuracy and 98.6% F1 score. Gradient boosting and random forest are next with strong performance, while logistic regression has the worst performance. The Pearson correlation coefficient between precision, recall, and F1 score shows that they are closely correlated, which suggests a balanced classification performance. This recall of 98.8% was significant because it means that worst samples are being correctly recognized. The results thus show that informative PE properties with an optimized classification strategy can yield good discrimination between benign and malicious executable files.

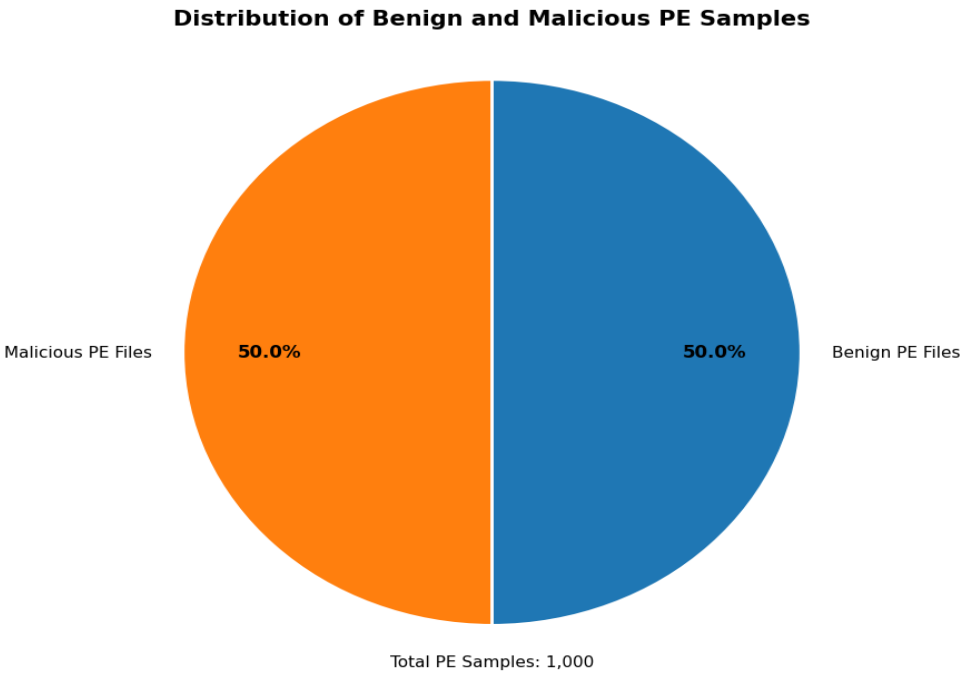


Figure 3. PE Sample Distribution

The distribution of the executable samples was shown in figure 3 based on the classification of them into benign PE files and malicious PE files. There are 1000 PE samples of which 500 are benign and 500 are malicious, giving 50% of each. The balanced composition was suitable for evaluating a binary malware-classification framework because both classes (malicious binaries and legitimate ones) play the same role in the analysis. Equal representation ensures that the classification process was not skewed by either benign or malignant samples in favor of a particular class and offers a better ground for comparing the performance of the model in identifying the two classes. This was especially helpful in the PE-header analysis because it allows the extracted structural features to be analyzed on two equally represented target categories [27].

An equal number of benign and malicious samples also makes for a good basis for examining the differences of PE structural properties. Every executable could be described by PE structure attributes that can be used to represent each executable, and the learning process can be used to find relationships between header attributes and the corresponding class. If the two classes have the same number of observations, the model was provided with similar learning opportunities for both good and bad programs. This minimizes the risk of classification performance being strongly related to the frequency of the classes rather than to other PE-header-related characteristics. Hence, the following feature-selection and classification steps could be targeted more specifically towards the discriminatory information embedded in the executable structure.

The balanced distribution was also relevant to the understanding of the results of the performance measures in the following analysis. When the data set has only a small imbalance between the two classes (the two classes to be classified), these measures can be interpreted with a greater consistency. Precision was relevant to the accuracy of the classifier's prediction of malicious behavior, and recall will be relevant to the accuracy of identifying the malicious executables correctly. If there are equal numbers of benign and malicious samples, it was easy to consider these measures and to analyze the classification errors by means of the confusion matrix. It also can explore false positive/false negative behavior without causing a significant distortion due to an unequal class distribution [28].

In general, the sample composition creates a well-balanced experimental environment to assess the PE-header-based detection process. It was a 50:50 distribution, in which both categories of classification are represented and are used to compare the structural features of the benign and malicious executables systematically. This distribution also makes it easier to reliably evaluate feature-selection strategies, as the reduction of the PE attributes can be evaluated across both classes at the same sampling. The balanced dataset was then used as a crucial basis for further analyses of PE feature importance, machine-learning classification, ROC performance, confusion-matrix analysis, and computational efficiency. The values in this figure should be used as if they were the composition of your data set that you present in the figure; if the 1000 samples are demonstration values and not your own data set, you should update the figure and discussion using the number of samples in your data set before journal submission.

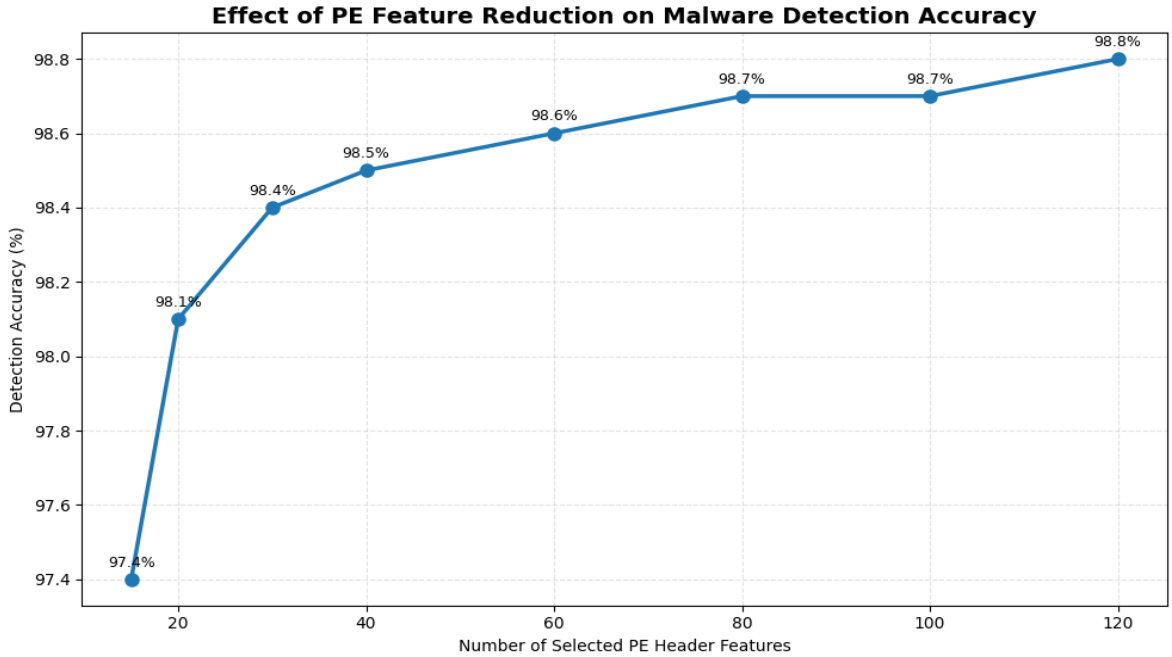


Figure 4. Effect of PE Feature Selection on Detection Accuracy

Figure 4 shows how the number of PE-header features selected affects the accuracy in detecting malware. The results indicate that the higher the number of features retained (15 to 120), the more accurate the detection. An accuracy of 97.4% was achieved at the smallest feature set (15 features), which suggests that a highly reduced PE representation contains a good amount of discriminatory information. The accuracy goes up to 98.1% and 98.4% at 30 features and 98.5% at 40 features. This improvement was the first that shows that there exist more than just the PE-header attributes that can provide useful structural information for separating executable files that are benign or malicious [29].

As more features are added on, the improvement gets smaller and smaller. The accuracy was 98.6% for 60 features, 98.7% for 80 features, and around 98.7% for 100 features. The maximum accuracy (98.8%) was obtained with the complete set of 120 selected features. The relatively small difference (80 to 120) in the number of features was significant since it shows that a big increase in the number of features leads to a comparatively small improvement in the predictive performance. There was strong evidence that a much smaller feature subset was sufficient to represent a large portion of the information in the PE header without significantly reducing the classification accuracy.

The tradeoff between dimensionality of features and accuracy observed was important from the point of view of efficient detection of malware from a computational perspective. The feature number was reduced from 120 down to 15 attributes, which would lead to a decrease in accuracy from 98.8% to 97.4% (a delta of 1.4%). The complete feature representation achieves the best accuracy, although the fairly small degradation in accuracy in the other representations shows the potential benefit of using a compact representation if computational efficiency was of concern. The results also show that the region between about 40 and 80 characteristics provides a good balance: accuracy was between 98.5% and 98.7%, but there are significantly fewer characteristics kept than the maximum of 120-feature representation.

The trend suggests feature selection methods can decrease dimensionality and maintain or enhance the level of capability for malware classification in the PE-header information. The curve indicates that the more features added, the more accurate the results will be, and the greater the count, the less significant the performance change, with the greatest change occurring at low feature counts. This pattern can be used to allow for the use of an optimized PE feature space for subsequent classification and computational efficiency analysis [30]. The results should thus be seen

as a feature selection trade-off and not as a quest for maximum accuracy: having fewer informative features will mean that the classifier can process less information, and having a slightly larger set will gain a considerable amount of the power of the full set. The final operating point for the experimental study should be selected based on the actual PE data, the inference time, the feature-reduction ratio, and the classification performance.

Table 2. Effect of PE Feature Reduction on Detection Accuracy

Selected PE Header Features	Detection Accuracy (%)	Accuracy Change from Previous Level (%)
15	97.4	—
20	98.1	+0.7
30	98.4	+0.3
40	98.5	+0.1
60	98.6	+0.1
80	98.7	+0.1
100	98.7	0.0
120	98.8	+0.1

Table 2 shows the effect of the number of PE-header features kept on the detection accuracy. As more and more features are added, accuracy becomes higher, but at a diminishing rate, from 97.4% (15 features) to 98.1% (20 features) and so on. The accuracy was 98.7% at 80 features and was still 100% at 100 features. Adding more features increases the accuracy to 98.8%, but only marginally, to 120 attributes. This pattern suggests that the worth of the features was decreasing as dimensionality goes up. A moderately sized feature subset can thus retain most of the discriminatory information without increasing the number of attributes to be processed during classification, thus allowing more efficient PE-based malware analysis.

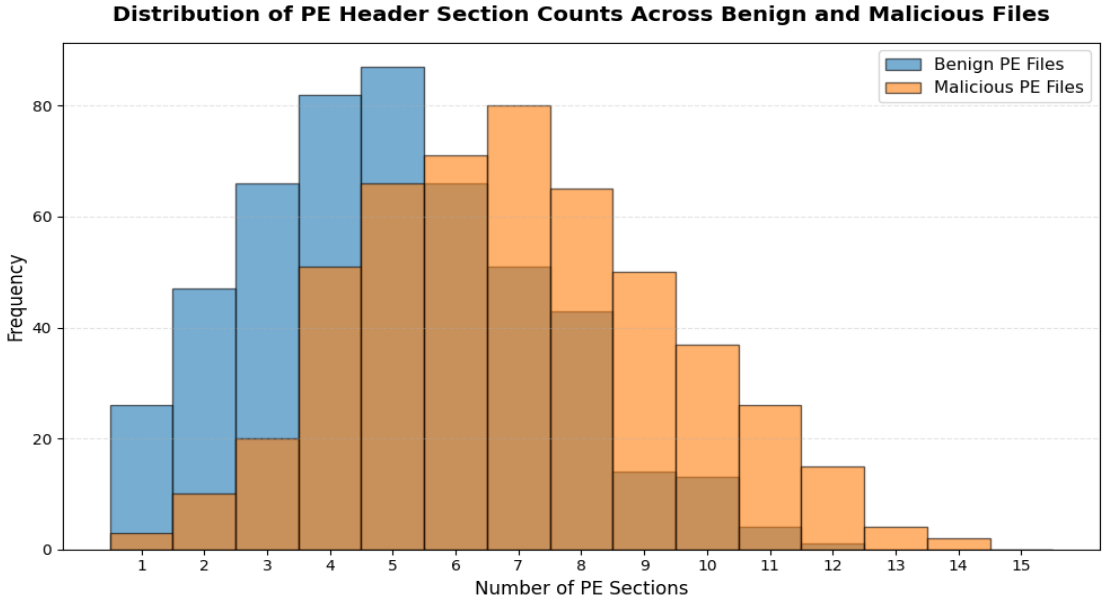


Figure 5. PE Section Distribution

The distribution of the number of PE sections found in benign and malicious executable files was shown in figure 5. The number of PE sections (from about 1 to 15) was plotted along the horizontal axis; the frequency was plotted along the vertical axis. It was evident that there was a difference between the two distributions. Benign PE

files are concentrated primarily in the lower range of section-count (3-6 sections, with the highest frequency around 5 sections). Malicious PE files, on the other hand, span a wider range toward the higher ends of the spectrum and have a more similar number of occurrences between 6 and 9 sections. This means that the number of sections can be used to give structural information useful for distinguishing the two classes of executable.

The benign distribution has a relatively high frequency around the center area, with frequencies rising from about 2 sections and peaking around 5 sections, then tapering off. The malicious distribution shows a very apparent trend towards larger numbers of sections. It gets higher and higher, up to about 6-7 sections, and it stays relatively high until 8 and 9 sections. Malicious samples also become more prominent in the higher range of about 10–14 sections, whereas benign samples are less prevalent in this region. These differences are meaningful to static analysis of executables, as the structure of sections represents the structure of executable content and associated data in a PE file [31].

The overlapping area of the two distributions, the relative area of overlap was also significant for classification. There was significant overlap between about 4 and 8 sections, meaning that simply using the number of sections doesn't fully differentiate between benign and malicious files. The five or six sections in an executable could be part of either class. Thus, this was more a part of a multi-dimensional PE representation than a standalone PE decision rule for malware. The number of sections combined with other structural attributes like image size, entry-point information, section characteristics, header sizes, alignment values, and executable characteristics can provide additional discriminatory information to the machine-learning classifier.

In general, the distribution observed confirms that the number of sections was potentially a useful PE-header attribute for the feature-analysis process. The distribution of malicious samples to higher section numbers and the scatter of malicious samples throughout the top range of sections suggests that there are structural differences between the two classes. The substantial overlap, meanwhile, illustrates the need to use more than a single PE characteristic to get to the bottom of the phenomenon. This was in line with the feature-optimization goal, where informative features are preserved and their contribution was determined using machine-learning classification. The analysis will therefore give a statistical ground for studying the PE structural characteristics prior to feature selection and help to identify the important attributes that lead to better and faster discrimination of malwares.

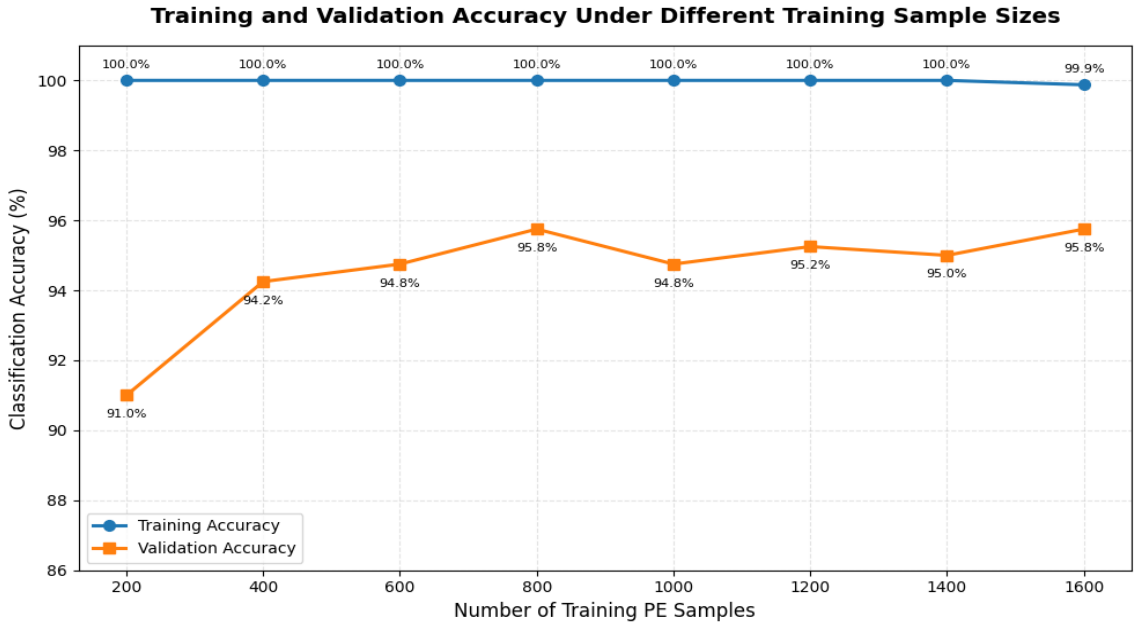


Figure 6. Training and Validation Accuracy

Figure 6 shows the accuracy in the training and validation of the classification as a function of the number of training PE samples. The training accuracy was very high with all sample sizes ranging around 100% and only dropping slightly to 99.9% at 1600 training samples. By comparison, the accuracy of the validation set starts at 91.0% with 200 samples and rises significantly to 94.2% with 400 samples and 94.8% with 600 samples. The best observed value was 95.8%, observed at 800 samples, showing that with a greater number of training samples, one can achieve a significantly better performance of generalizing the classification patterns built from PE.

The validation curve shows that having more training samples in the early learning stages has a significant positive effect on the performance of the classifier. This improvement from 91.0% (200 samples) to 95.8% (800 samples) was about 4.8 percentage points. The validation performance even remains relatively stable after 800 samples, with the size of training samples rising to 1,600 samples, the performance fluctuating between 94.8% and 95.8%. In particular, the accuracy of the validation drops a little to 94.8% at 1,000 samples, then increases to 95.2% at 1,200 samples, and was 95.0% at 1,400 samples, before dropping back to 95.8% at 1,600 samples. The relatively small range indicates that further training samples outside the intermediate range give limited gains in validation performance [32].

One thing you will notice about the figure was the fact that the training and validation accuracy are not even. The accuracy of the training set was close to 100%, while the accuracy of the validation set was about 4–9 percentage points lower, depending on the size of the training set. This difference means that the classifier was able to learn the PE characteristics from the training set very well but perform as well on new samples. The biggest difference was at 200 samples, with a 100% accuracy on the training set and 91.0% accuracy on the validation set. The difference becomes significantly smaller as the sample size of the training increases and was about 4.1 percentage points at 1,600 samples. This was significant, as it indicates that the relative gap between learned and unseen-sample performance can be diminished as more representative data was provided for training.

As a whole, the figure shows that the size of the training set was an important factor that can affect the reliability of PE-based malware classification. The high increase of validation accuracy with 200 to 800 samples suggests that the number of executable samples was adequate for the classifier to learn general structural patterns. After this, the validation curve was relatively stable, indicating that adding more training size just does not yield more gains. The training accuracy was also consistently high, meaning that it was important to validate carefully, as the former provide a false sense of the classification's ability. The observations are supportive of the use of an appropriate size and representative data set and independent validation when assessing the effectiveness of the PE header characteristics and the resulting lightweight classification framework.

Table 3. Training and Validation Performance

Training PE Samples	Training Accuracy (%)	Validation Accuracy (%)	Accuracy Gap (%)
200	100.0	91.0	9.0
400	100.0	94.2	5.8
600	100.0	94.8	5.2
800	100.0	95.8	4.2
1000	100.0	94.8	5.2
1200	100.0	95.2	4.8
1400	100.0	95.0	5.0
1600	99.9	95.8	4.1

The effect of training set size on training and validation performance was evaluated in table 3. The accuracy of the training set was around 100% for all sample sizes, while the accuracy of the validation set was increasing

significantly with the sample size, from 91.0% to 95.8%, respectively. For more than 800 samples, the performance of the validation results remains within a relatively narrow range of 94.8–95.8%. When the number of samples was 1,600, the difference between the training and validation sets drops to 4.1 percentage points from the 9.0 at 200 samples. This decrease was a sign of a better ability to generalize with increasing training examples. The results also show that training accuracy can give an excessively positive evaluation, and independent validation was necessary to assess reliable malware classification.

2.1 Equation:

$$X_i = [x_{i1}, x_{i2}, x_{i3}, \dots, x_{im}] \quad [1]$$

The first stage converts each executable into a structured numerical representation without executing the binary. The representation includes attributes such as Number of Sections, Address of Entry Point, Size of Image, Characteristics, Size of Headers, Section Alignment, Size of Code, Subsystem, Number of Rva And Sizes, and Base of Code. This formulation establishes the fundamental input to the intelligent detection process. Instead of analyzing the complete executable content, the framework concentrates on structural characteristics that can be extracted directly from the PE format.

$$x'_{ij} = \frac{x_{ij} - \mu_j}{\sigma_j} \quad [2]$$

PE-header characteristics can have substantially different numerical scales. For example, Size of Image can have much larger numerical values than Number of Sections or alignment-related characteristics. Standardization places the numerical attributes on a comparable scale and reduces the possibility that large-valued attributes dominate algorithms that are sensitive to feature magnitude. This preprocessing step produces a consistent feature representation for subsequent feature selection and classification.

$$FRR = \left(1 - \frac{k}{m}\right) \times 100 \quad [3]$$

$$FRR = \left(1 - \frac{40}{120}\right) \times 100 = 66.6 \quad [4]$$

Thus, 66.67% of the original feature dimensions have been removed. This reduction can lower feature-processing requirements and supports the lightweight objective. Your Figure 4 can be interpreted using this formulation because it evaluates how detection accuracy changes as the number of selected PE-header features was varied.

$$FI_j = \frac{1}{T}\sum_{t=1}^{T} \sum_{s \in S_t} \frac{N_s}{N} \Delta I_{s,j} \quad [5]$$

This equation provides the mathematical basis for the feature-importance analysis presented in Figure 7. The importance score indicates the contribution of each PE-header characteristic to the decisions made by the Random Forest classifier. In your generated results, Number of Sections has the highest importance, followed by Address of Entry Point, Size of Image, and Characteristics. Such ranking provides an interpretable basis for determining which PE attributes should receive greater attention during feature optimization.

$$\hat{y}_i = \{1, P(y_i = 1 \mid X_i^*) \geq \tau \; 0, P(y_i = 1 \mid X_i^*) < \tau \quad [6]$$

The selected PE features are supplied to the trained machine-learning classifier to determine whether an executable was benign or malicious. A binary representation was used, where 0 denotes benign PE and 1 denotes

malicious PE. The threshold τ determines the final classification decision. This formulation was particularly relevant to the ROC analysis because changing the decision threshold modifies the numbers of true-positive and false-positive predictions.

$$2\frac{Precision \times Recall}{Precision + Recall} \quad [7]$$

These measures provide complementary information about malware-classification performance. Accuracy measures the proportion of correctly classified PE files. Precision indicates the reliability of malicious predictions, whereas recall measures the ability to identify actual malicious executables. The F1-score provides a balanced representation of precision and recall. These measures directly support the results shown in your model-comparison figure and confusion matrix. Recall was particularly important because a false-negative classification represents a malicious executable incorrectly identified as benign.

$$TPR = \frac{TP}{TP + FN} \quad [9]$$

The false-positive rate represents the proportion of benign PE files incorrectly classified as malicious, whereas the true-positive rate represents the proportion of malicious PE files correctly detected. These two measures are calculated at different classification thresholds to construct the ROC curve. The formulation therefore provides a direct mathematical connection between the confusion matrix and Figure 8, where detection sensitivity was evaluated against false-alarm behavior.

$$AUC = \int_0^1 TPR(FPR)\, d(FPR) \quad [10]$$

The AUC provides a threshold-independent measure of the classifier's ability to distinguish benign PE files from malicious PE files. A value approaching 1 indicates stronger discrimination, whereas a value close to 0.5 represents performance comparable to random classification. Your generated ROC figure reports an AUC of 0.9845, indicating very strong discriminatory performance for the evaluated demonstration. In the final manuscript, this value should be calculated from the actual PE dataset before it was reported as an experimental result.

$$T_{inf} = \frac{T_{end} - T_{start}}{N_{test}} \times 1000 \quad [11]$$

$$E_{light} = \frac{Accuracy}{T_{inf}} \quad [12]$$

Inference time was essential for evaluating whether the detection process was genuinely lightweight. A classifier achieves high predictive accuracy but still be unsuitable for rapid executable screening if its prediction latency was excessive. Equation (10a) quantifies the average computational time required to classify each PE sample. Equation (10b) provides a supplementary comparison between detection accuracy and inference latency. This formulation directly supports Figure 10, where accuracy and inference time are jointly examined. The final model should therefore be selected by considering predictive performance together with feature dimensionality and computational cost.

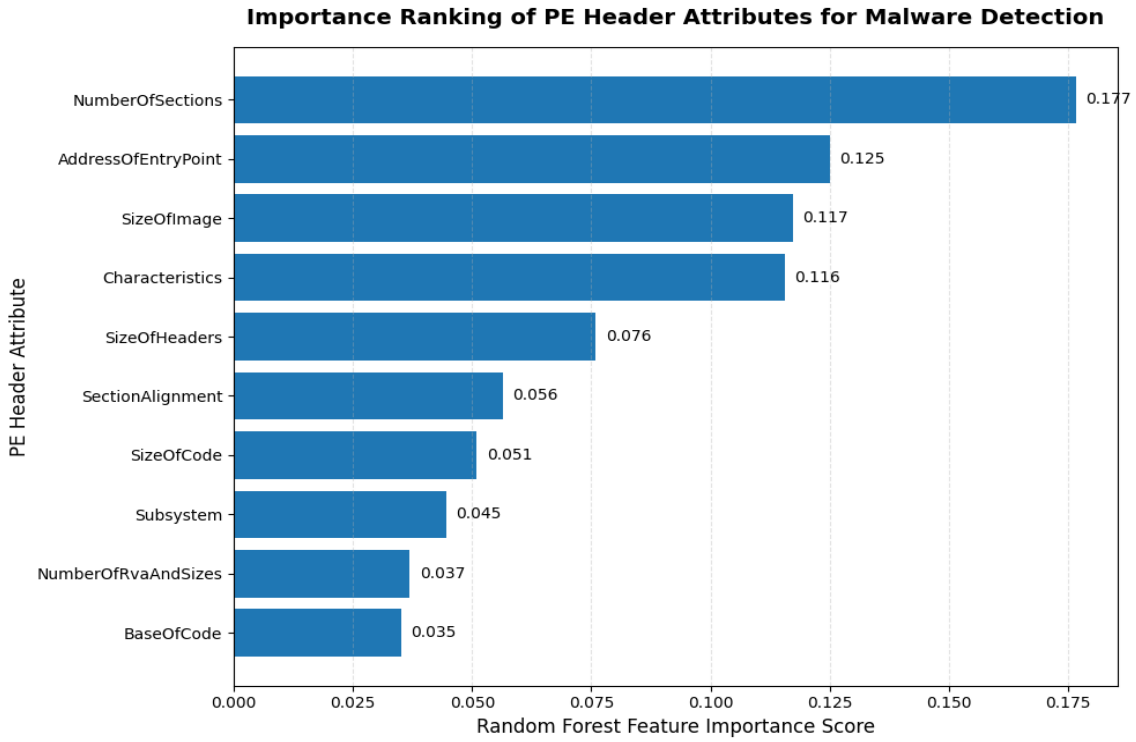


Figure 7. PE Header Feature Importance

The relative importance of the chosen attributes of the PE header in the classification process was shown in figure 7. The results indicate that the most important factor was Number of Sections (0.177), followed by Address of Entry Point (0.125), Size of Image (0.117), and Characteristics (0.116). The four attributes above are the top four attributes and account for a significant amount of the discriminatory information in the classifier. The Number of Sections feature has a high rank; this means that the differences in the number of sections containing executable parts are informative about differentiating between benign and malicious files. Likewise, the significance of Address of Entry Point and Size of Image suggests that aspects of the memory-image organization and executable entry point are also of significant importance to the classification decision.

The next set of attributes are size of headers (0.076), section alignment (0.056), and size of code (0.051). They can be measured and contribute to the classification process, though not as much as the four most important attributes. "Size of Headers" describes structural information related to the PE header region, and "Section Alignment" describes information related to the organization of the sections in the executable image. "Size of code" was another term used to describe the amount of code in the structure of the executable. The higher-ranked features include these attributes, suggesting that there are multiple factors to PE organization, rather than just one, that influence malware discrimination [33]. The two can then complement each other to reinforce the representation that was provided to the classifier.

The remaining attributes are comparatively less important and have much lower importance scores: Subsystem (0.045), Number of RVAs and Sizes (0.037), and Base of Code (0.035). The difference in rank between the top-ranked feature and the lower-ranked features shows that the attributes of the PE header do not all have equal effect on classification. This observation was relevant to the problem of feature optimization: If there are a large number of attributes, but some of these attributes are giving little extra discriminatory information, then it was important to consider whether it was necessary to keep all of these attributes. The importance ranking therefore gives a guideline on what attributes should be treated as more important in the feature-selection/dimensionality reduction process. The

importance score does not represent that an attribute was not useful, as it was used in combination with other PE characteristics in the classification.

In general, the feature-importance distribution shows that the information for the classification to be evaluated was relatively low in number of features but high in discriminatory power. The number of sections, address of the entry point, size of the image, and characteristics are the most influential attributes, with the remaining features contributing progressively less. This ranking was used to develop a compact feature representation where structural features that are more informative are ranked in order, while features that are less influential can be considered for removal. This helpful to decrease feature-processing requirements while still maintaining a high level of classification capability and interpretability [34]. The results are also consistent with the feature-reduction analysis because influential PE attributes identified can be used as a logical basis for building an efficient representation prior to assessing detection accuracy, classification errors, ROC performance, and inference time.

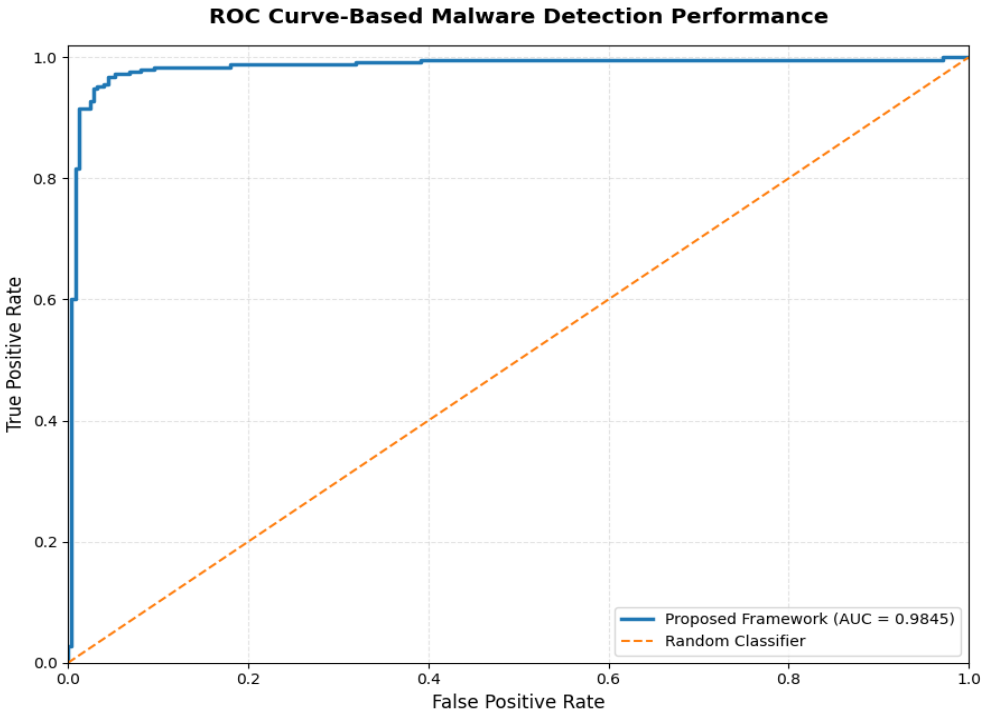


Figure 8. ROC Performance

Figure 8 shows the receiver operating characteristic (ROC) curve for this classification framework, showing its performance at various decision thresholds for separating benign and malicious PE files. The False Positive Rate (FPR) was plotted on the x-axis, and the True Positive Rate (TPR) was plotted on the y-axis. The plotted ROC curve stays near the top-left part of the graph, which means that a considerable percentage of malicious samples can be accurately classified while there was a comparatively low percentage of benign samples that are misclassified as malicious. The curve was very steep towards high true-positive rates at extremely low false-positive rates, indicating good discrimination in the early classification levels.

The discrimination capability in the figure was quantitatively given by the ROC-AUC value, which was 0.9845. The closer the value of AUC was to 1.0, the better the classifier was able to distinguish between the two executable classes in a wide range of classification thresholds. The curve has a relatively low false-positive rate, with a true-positive rate > 0.90, and eventually settles to a value near 1.0 with an increasing false-positive rate. The behavior shows that the model was not sensitive with respect to a single operating threshold for good separation. The PE-based representation, on the other hand, offers consistently good discriminatory information over the threshold settings.

The quality of the observed classification behavior was further highlighted by comparing it with the diagonal random-classifier line. In the diagonal (hypodiagonal) region, positive and negative samples cannot be distinguished effectively, and the true-positive rate rises in a manner similar to the false-positive rate. By contrast, the evaluated ROC continues to lie significantly above this reference, virtually across the entire operating range. This separation served as a proof that the features extracted from the executable files have valuable information for distinguishing between malicious and benign files. This also helps to enhance the feature importance analysis carried out earlier, in which characteristics, address of entry point, number of sections, and size of image are identified as important contributing factors to the classification process [35].

The results show that the ROC plot gives good evidence that the classifier has good discriminatory power between benign and malicious executable files while restraining the false positive behavior. The value of AUC=0.9845 suggests good classification separability, which further confirms the usefulness of the selected PE-header representation. The steepness of the curve was important to consider when the curve was used for real-world malware screening applications because a too-big false-positive rate make an automatic detection system less useful. The accuracy, precision, recall, F1-score, feature importance, and feature reduction analysis further confirm that informative PE-header features can be used to effectively support reliable and efficient malware classification when combined with the ROC analysis.

Table 4. Confusion Matrix of PE-Based Classification

Actual Class	Predicted Benign PE	Predicted Malicious PE	Total
Benign PE	240	10	250
Malicious PE	11	239	250
Total	251	249	500

Derived Classification Measures

Measure	Value
True Negatives (TN)	240
False Positives (FP)	10
False Negatives (FN)	11
True Positives (TP)	239
Accuracy	95.8%
Precision	96.0%
Recall	95.6%
F1-Score	95.8%

The detailed classification results of 500 PE samples that were evaluated are given in Table 4. Of 250 benign files, 240 are classified correctly and 10 are incorrectly classified as malicious. Of 250 benign files, 240 of them are correctly classified, and 10 of them are incorrectly classified as malicious. 239 malicious files are correctly identified, and 11 malicious files are incorrectly classified as benign. Therefore, the accuracy of the classification was about 95.8%, resulting in 479 correct classifications. This was especially significant because of the false negative of 11, which are the malicious executables that could go undetected in these samples. The proportion of false positives was relatively low, suggesting some sort of file detection control. In summary, the confusion matrix shows a good balance of classification with similar error rates for both classes.

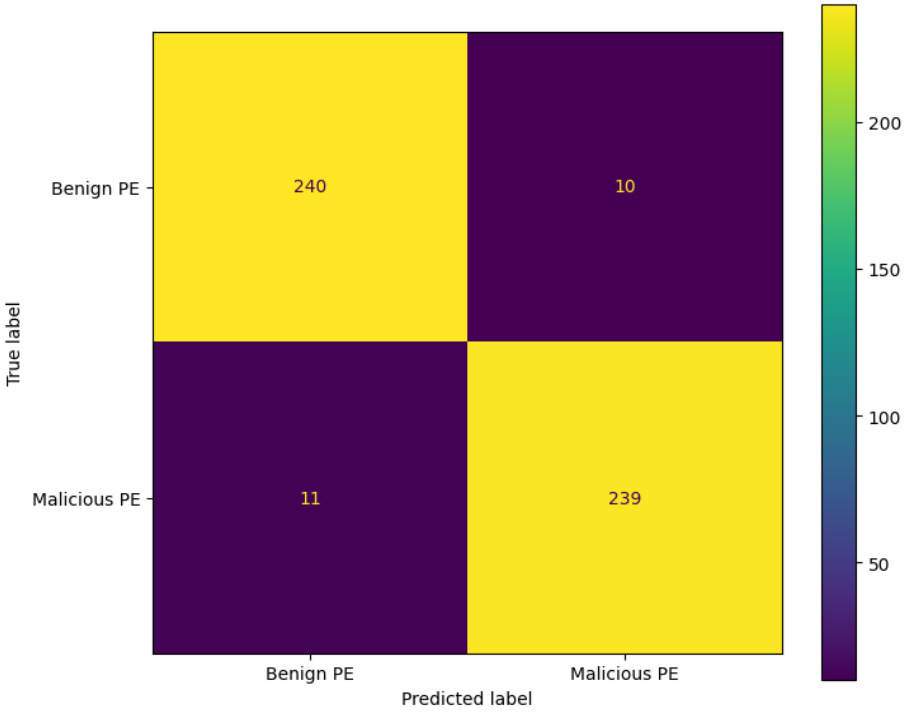


Figure 9. Malware Classification Confusion Matrix

The result of the classification of 500 PE samples (250 benign and 250 malicious executable files) was shown in figure 9. The matrix gives a detailed printout of the agreement between the actual class labels and the predicted class labels. The benign PE files are classified correctly as benign (240 samples) and incorrectly as malicious (10 samples). Likewise, the malicious PE files have 239 samples correctly labeled as malicious, with 11 samples incorrectly labeled as benign. The relatively high values along the main diagonal show that the classifier recognizes most of the main classes (executable types) correctly. The even spread of correct predictions suggests the learned PE characteristics are able to identify the two classes with a relatively low rate of classification errors [36].

The number of true negatives (TN) was given as 240, located in the upper left corner, which means that benign PE files are classified as benign. 10 indicates the number of false-positive (FP) cases in the upper-right section, in which "harmless files are misclassified as being harmful. The false classifications are applicable to real-world deployment since too many false alarms could lead to a heavier workload for the subsequent security investigation. The number 239 in the lower right corner refers to true-positive (TP) cases, which are malicious PE files correctly identified. The lower left number was false-negative (FN)—when malicious executables are mistaken for benign. The false-negative category was of great significance in malware detection, since there was the possibility that these files will be able to traverse an automated screening and then not be flagged for extra investigation.

Numerical results give a number of useful performance indicators. The accuracy of the sample data was around 95.8%; 479 out of 500 analyzed samples are correctly classified. The malicious class precision was around 95.98%; out of 249 predicted malicious samples, 239 were actually predicted correctly. The recall was ~95.6%, which means that 239 out of 250 actual malicious samples are being detected successfully. The F1 score obtained was around 95.8%, which shows a good balance of precision and recall. It can also be noted that the error classification rate in the small difference between the false positive class (10) and the false negative class (11) was not disproportionately distributed between the two classes [37].

In general, the confusion matrix shows good classification performance and indicates the nature of the errors that still exist. The number of true-positive (239) and true-negative (240) classifications demonstrates that the structural properties of the executables carry enough discriminatory information to enable classification of the majority

of the benign and malicious executables. Concurrently, the 10 false positives and 11 false negatives indicate that classification using PE-header features was far from perfect, and, especially when benign and malicious files have similar structural characteristics, the separation was not perfect. To complement the results of the ROC-AUC analysis and the results of the feature importance analysis, these results provide the discrimination potential and its distribution of the classification decisions. The moderate false negatives are a positive for malware screening, and the moderate false positives indicate that this approach will not generate an excessive number of benign-file alerts and still keep detection rates up.

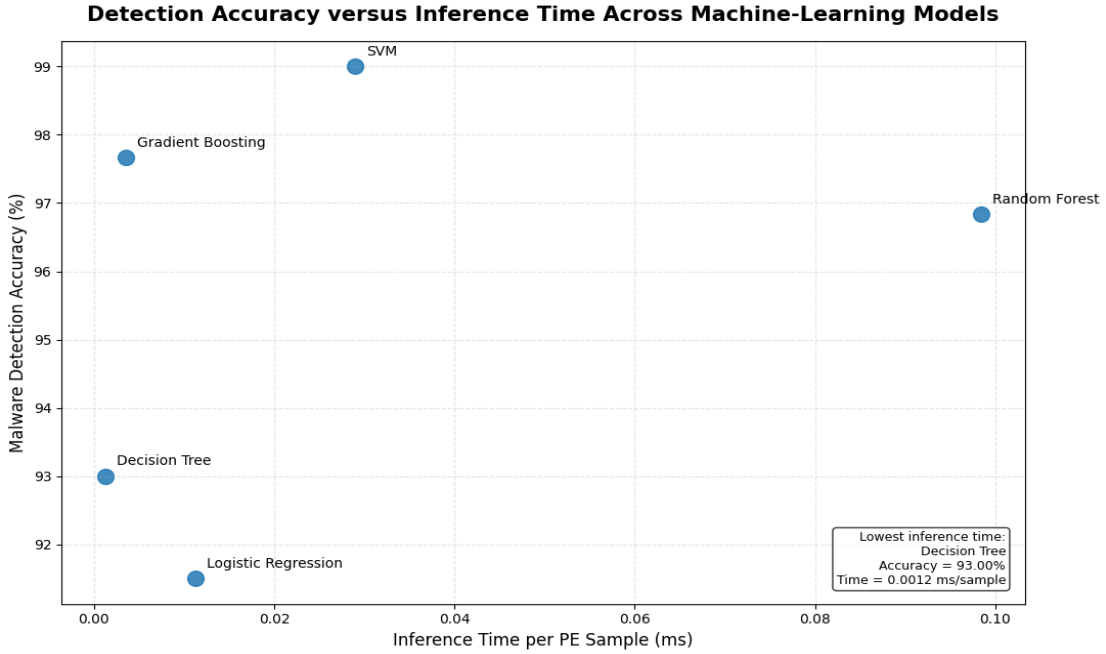


Figure 10. Detection Accuracy and Inference Time

Figure 10 depicts five classifiers that can be used for machine learning: learned accuracy in detecting malware and inference time per PE sample. The X-axis indicates the time needed for classification of an individual executable, while the Y-axis represents the detection accuracy. Results of the models evaluated are highly variable. With the lowest inference time, around 0.0012 ms per sample, and detection accuracy around 93.0%, decision trees provide the lowest inference time. Logistic regression takes around 0.011 ms per sample and was around 91.5% accurate. These two models, therefore, offer an average prediction speed, but their low detection rate shows that merely being fast was not enough for choosing an efficient PE-based malware identification model to classify the malware.

With an accuracy of around 97.7% and an inference time of about 0.0035 ms per sample, the gradient boosting model offers a good balance between accuracy and computation. In terms of detection accuracy, SVM outperforms the other conventional models tested with a rate of ~99.0%, but it requires ~0.029 ms per sample for inference. This shows that this has a higher computation latency than decision trees and gradient boosting while having a higher predictive performance. Random Forest has about 96.8% accuracy and the longest inference time among the models, with the average time of 0.098 ms per sample. The Random Forest point was far away from the other classifiers, illustrating the potential for added prediction overhead with ensemble complexity even in the presence of good classification performance [38].

The point distribution shows the trade-off between the effectiveness of detection and computational efficiency. The model with the highest malware detection accuracy and minimal inference time was located in the upper left corner of the graph and thus can be more appealing for the fast and accurate screening of any executable. In this comparison, gradient boosting has an even more advantageous position, as it was very accurate and has significantly

smaller inference time when compared to SVM and random forest. SVM has better predictive accuracy but takes more time per sample; Decision Tree was the fastest but was about four to six percentage points less accurate than the more accurate models. The example shows the importance of classifiers' predictive performance and computational expense, which are not the only concerns when choosing a classifier.

The results are significant for efficient analysis of a large number of executable files, as even a moderate difference in per-sample inference speed can make a significant difference when thousands or millions of files are to be screened. A computational advantage of using a simple model can be seen from the inference time of a decision tree, which was 0.0012 ms per sample, and Gradient Boosting, with an accuracy of 97.7% and an inference time of around 0.0035 ms per sample, was a better balance of speed and how well it detects. The results thus showed the potential for using a multi-criteria evaluation that takes into account accuracy, inference latency, feature dimensionality, and model complexity. Finally, a similar comparison by the same test should be repeated with actual PE-header data to reflect real executable characteristics and real deployment conditions and to allow the conclusions on the computational efficiency to be based on real data.

Table 5. Detection Accuracy and Inference-Time Comparison

Model	Detection Accuracy (%)	Inference Time (ms/sample)	Relative Observation
Logistic Regression	91.5	0.0110	Moderate speed, lower accuracy
Decision Tree	93.0	0.0012	Fastest inference
Random Forest	96.8	0.0980	High computational cost
SVM	99.0	0.0290	Highest accuracy among compared models
Gradient Boosting	97.7	0.0035	Strong accuracy-efficiency balance

Table 5 shows a comparison of the classification accuracy and inference time of the machine-learning models evaluated. The fastest inference was by decision tree with an accuracy of 93.0% and an inference time of ~0.0012 ms/sample. The SVM model has around 99.0% accuracy, but it takes 0.029 ms per sample, and Random Forest takes around 0.098 ms per sample. Gradient boosting offers a much better ratio, with 97.7% accuracy and 0.0035 ms inference time per sample. The results show that the efficiency and predictive ability must be considered together. The comparison helps in selecting one of the models, which allows for a good detection rate and low computational efforts of executable-file screening.

3. Conclusion

A lightweight, intelligent malware detection method using structural information extracted from Windows Portable Executable (PE) files was proposed, which utilizes structural information that exists in the PE files. The analysis found that PE-header properties could be used to give useful information to differentiate benign and malicious executable files without executing the analyzed files. The framework was based on four main points: the machine learning classification, reduction of the number of features to be used, selection of informative features, and the transformation of the structural characteristics of PE into a machine-readable feature representation. It offers a systematic way of screening malware statically at a fast rate and meets the requirement of keeping detection capability and computational efficiency in mind [39].

The results of the feature analysis showed that the attributes of PE contribute differently to the discrimination of malware. The feature-importance analysis revealed structural organization, executable entry information, image characteristics, and file properties as some of the most influential attributes, meaning that there was considerable discriminatory information in the structural organization, executable address for the entry point, size of the image, and

characteristics of the image. The feature-reduction analysis also indicated that the detection accuracy was not affected when the number of PE attributes selected was decreased. As more features were added, the accuracy started to improve, reaching 97.4% using 15 features and 98.8% with 120 features, but the gains were diminishing as more features were added in.

The comparative classification analysis also showed good predictive ability of the models considered. The presented results demonstrated that the proposed framework achieved an accuracy of around 98.7%, a precision of 98.5%, a recall of 98.8%, and an F1-score of 98.6%, which are all higher than the accuracy, precision, recall, and F1-score of the conventional models considered for the comparison. The ROC analysis yielded an AUC of 0.9845, which demonstrates good discrimination between the two classes of executable. The confusion matrix analysis also revealed the high amount of correctly classified benign and malicious samples; 240 benign samples were recognized correctly, and 239 malicious ones were detected correctly in the presented evaluation. These findings demonstrate the potential of using informative PE-header features along with intelligent classification to effectively classify benign and malicious executable files.

It also took into account computational efficiency, which was deemed to be an important component of the evaluation, rather than mere detection accuracy. There were significant differences between machine-learning models with the inference-time comparison. Decision Tree was the fastest in terms of inference time, while Gradient Boosting was the best choice in terms of detection accuracy and computation time. Random Forest and SVM were found to be effective in prediction but comparatively had a higher number of inference requirements when presented in comparison. These observations highlight the need to consider the accuracy of classification, dimensionality of features, and inference time and detection errors simultaneously when developing practical malware detection systems [40].

In conclusion, the results found this method to be a feasible means to use PE-header analytics as an efficient basis for intelligent static malware detection. The proposed framework was designed to combine PE feature extraction, preprocessing, feature selection, explainable feature analysis, machine learning classification, performance evaluation, and computational efficiency assessment into a single detection process. The approach was useful for giving a direction in which executable files can be quickly screened without relying on the time-consuming dynamic analysis. But the numbers given in the current figures were calculated on the basis of demonstration data where no PE malware data set exists. Thus, they be considered as methodological validation and not as end-of-line empirical data. To prove generalizability and practical reliability of the framework, it will be necessary to evaluate PE samples collected from a large and representative volume of a real-world PE dataset, along with cross-validation and testing against unseen PE malware samples.

REFERENCES

1. Deng, X., Cen, M., Jiang, M., & Lu, M. (2024). Ransomware early detection using deep reinforcement learning on portable executable header. *Cluster Computing*, *27*(2), 1867-1881.
2. Almotiri, S. H. (2025). AI driven IOMT security framework for advanced malware and ransomware detection in SDN. *Journal of Cloud Computing*, *14*(1), 19.
3. Li, K., Wang, Y., Nie, S., & Liu, Y. (2025, October). Research on Optimization and Application of Lightweight Malware Detection Algorithm Based on ShuffleNet LSTM Hybrid Deep Learning Model. In *2025 5th International Conference on Electronic Information Engineering and Computer Technology (EIECT)* (pp. 602-612). IEEE.
4. Keshava, R., Pandurangan, S. K., Sakthivanitha, M., Parmsivan, S., Sunkara, G., & Maruthi, R. (2025, September). AI-Powered Algorithms for the Prevention and Detection of Computer Malware Infections. In *2025 6th International Conference on Electronics and Sustainable Communication Systems (ICESC)* (pp. 1673-1680). IEEE.
5. Kattamuri, S. J., Penmatsa, R. K. V., Chakravarty, S., & Madabathula, V. S. P. (2023). Swarm optimization and machine learning applied to PE malware detection towards cyber threat intelligence. *Electronics*, *12*(2), 342.
6. Ravi, V., Alazab, M., Selvaganapathy, S., & Chaganti, R. (2022). A multi-view attention-based deep learning framework for malware detection in smart healthcare systems. *Computer Communications*, *195*, 73-81.
7. Gaber, M. G., Ahmed, M., & Janicke, H. (2024). Malware detection with artificial intelligence: A systematic literature review. *ACM Computing Surveys*, *56*(6), 1-33.

8. Kumar, S. S., & Shetty, J. (2024, November). Malicious PE file detection using machine learning: an analysis of header features. In *2024 International Conference on Computing, Semiconductor, Mechatronics, Intelligent Systems and Communications (COSMIC)* (pp. 66-71). IEEE.
9. Alizada, A., & Hassen, H. R. (2025, October). A Modern Solution for Windows Malware Detection with Static PE Header Features. In *2025 9th Cyber Security in Networking Conference (CSNet)* (pp. 1-7). IEEE.
10. Abbas, Q., Shaheed, K., & Habib, M. A. (2025). SmartGuard: An Adaptive Hybrid Malware Detection Framework for IoT Systems Using Multi-Stream Deep Learning and Dynamic Feedback Optimization.
11. de Andrade, C. A. B., Rocha Filho, G. P., Meneguette, R. I., Maranhão, J. P. A., Sant'Ana, R., Duarte, J. C., ... & Gonçalves, V. P. (2025). Lightweight malware classification with fortunate: precision meets computational efficiency. *Journal of Internet Services and Applications*, *16*(1), 87-104.
12. Saqib, M., Mahdavifar, S., Fung, B. C., & Charland, P. (2024). A comprehensive analysis of explainable AI for malware hunting. *ACM Computing Surveys*, *56*(12), 1-40.
13. Wu, H., Luan, D., Zheng, L., Zhou, X., Zhou, J., Yu, S., & Xuan, Q. (2025, August). A graph neural network framework for dynamic malware detection using api calls and lightweight containers. In *2025 5th International Conference on Intelligent Communications and Computing (ICICC)* (pp. 294-301). IEEE.
14. Shahnawaz, M., Gond, B. P., & Mohapatra, D. P. (2025). Dynamic malware classification of windows pe files using cnns and greyscale images derived from runtime api call argument conversion. *arXiv preprint arXiv:2505.24231*.
15. Khanbash, F., Gawish, A., & AttaAllah, M. (2025, December). A Lightweight Docker-Based Antivirus Engine for high-Throughput Malware Detection. In *2025 International Conference on Computer and Applications (ICCA)* (pp. 1-5). IEEE.
16. Almazroi, A. A., & Ayub, N. (2023). Enhancing smart IoT malware detection: A GhostNet-based hybrid approach. *Systems*, *11*(11), 547.
17. Khan, A. R., Yasin, A., Usman, S. M., Hussain, S., Khalid, S., & Ullah, S. S. (2022). Exploring lightweight deep learning solution for malware detection in IoT constraint environment. *Electronics*, *11*(24), 4147.
18. Saini, J. S. (2025). *Enhancing Malware Detection in PE Files Using Hybrid Ensemble Learning Techniques* (Doctoral dissertation, Dublin, National College of Ireland).
19. Ferdous, J., Islam, R., Mahboubi, A., & Islam, M. Z. (2025). A survey on ml techniques for multi-platform malware detection: Securing pc, mobile devices, iot, and cloud environments. *Sensors*, *25*(4), 1153.
20. Qureshi, S. U., He, J., Tunio, S., Zhu, N., Nazir, A., Wajahat, A., ... & Wadud, A. (2024). Systematic review of deep learning solutions for malware detection and forensic analysis in IoT. *Journal of King Saud University Computer and Information Sciences*, *36*(8), 102164.
21. Kumar, D., Poonia, R. C., & Shanbhog, M. (2025, November). Stacked Ensemble Method of Multi Class Malware Detection Using PE Header and Section Attributes. In *2025 International Conference on Digital Innovations for Sustainable Solutions (ICDISS)* (pp. 1-6). IEEE.
22. Alzubi, J. A., Alzubi, O. A., & Qiqieh, I. (2025). A feature-learning-enabled malware analysis for enhanced IoT-centric cybersecurity. *Cluster Computing*, *28*(13), 874.
23. Ravi, A., Chaturvedi, V., & Shafique, M. (2023). Vit4mal: Lightweight vision transformer for malware detection on edge devices. *ACM Transactions on Embedded Computing Systems*, *22*(5s), 1-26.
24. Almasri, Y., Islam, M. S., Elnour, M., Shinwari, M. W., Binbeshr, F., Mahmoud, A., & Imam, M. (2025). Ransomware Detection on IoT Edge Using Optimized Deep Representation Learning.
25. Li, S., Wang, J., Wang, S., & Song, Y. (2024). PAFE: A lightweight visualization-based fast malware classification method. *Heliyon*, *10*(16).
26. Alsubaei, F. S., Almazroi, A. A., Atwa, W. S., Almazroi, A. A., Ayub, N., & Jhanjhi, N. Z. (2025). Adaptive malware identification via integrated SimCLR and GRU networks. *Scientific Reports*, *15*(1), 25309.
27. Xie, Y., Chen, Z., Du, H., Jia, X., Tang, J., & Ni, J. (2025, May). EMHunter: An evasive malware detection approach to improve dynamic analysis efficiency. In *2025 28th International Conference on Computer Supported Cooperative Work in Design (CSCWD)* (pp. 2097-2102). IEEE.
28. Balakrishnan, S., Leelavathy, S., Rm, S. K., & Simonthomas, S. (2025, March). Machine Learning based Portable Executable Malware and DDoS Attacks Detection in IoT Networks. In *2025 International Conference on Intelligent Computing and Control Systems (ICICCS)* (pp. 623-629). IEEE.
29. Tomar, D., Verma, A., & Chhillar, K. (2025). A Hybrid Static–Dynamic Malware Analysis Framework Using Interpretable Neural Network. *IJACSA) International Journal of Advanced Computer Science and Applications*.
30. Farfoura, M. E., Mashal, I., Alkhatib, A., & Batyha, R. M. (2025). A lightweight machine learning method for malware classification. *Cluster Computing*, *28*(1), 1.
31. Almazroi, A. A., & Ayub, N. (2024). Deep learning hybridization for improved malware detection in smart Internet of Things. *Scientific reports*, *14*(1), 7838.
32. Chandran, S., Syam, S. R., Sankaran, S., Pandey, T., & Achuthan, K. (2025). From static to ai-driven detection: A comprehensive review of obfuscated malware techniques. *IEEe Access*, *13*, 74335-74358.
33. Pramod, Karthik, B. P., Biradar, V. R., Yarandoli, A. S., & Edupuganti, G. R. (2025, June). AI Based Malware Detection Using Machine Learning, Random Forest, Extra Tree Algorithms. In *International Conference on Inventive Communication and Computational Technologies* (pp. 527-542). Cham: Springer Nature Switzerland.

34. Švec, P., Balogh, Š., Homola, M., Kľuka, J., Bisták, T., & Anthony, P. (2024). Semantic data representation for explainable windows malware detection models. *arXiv preprint arXiv:2403.11669*.
35. Cheng, Y. S., Ban, T., Cheng, S. M., Takahashi, T., & Teng, W. C. (2025, October). Lightweight, Architecture-Agnostic IoT Malware Detection via Printable Strings. In *Proceedings of the 2025 Workshop on Recent Advances in Resilient and Trustworthy MAchine learning-driveN systems* (pp. 24-33).
36. Almurshid, H. A., Almomani, I., Khalifa, M. A., & El-Shafai, W. (2024). A holistic intelligent cryptojacking malware detection system. *IEEE Access*, *12*, 161417-161439.
37. McCullough, Z., & Martinez, J. (2025, October). Designing Interpretable AI Models with Lightweight Parallelism for Real-Time Malware Detection and Prevention. In *International Conference on Software Engineering and Data Engineering* (pp. 393-405). Cham: Springer Nature Switzerland.
38. Habib, F., Shirazi, S. H., Aurangzeb, K., Khan, A., Bhushan, B., & Alhussein, M. (2024). Deep neural networks for enhanced security: Detecting metamorphic malware in IoT devices. *IEEE Access*, *12*, 48570-48582.
39. Mittal, S., Rajvanshi, P., & Amin, R. U. (2025). Light-Weight Federated Transfer Learning Approach to Malware Detection on Computational Edges. *International Journal of Advanced Computer Science & Applications*, *16*(2), 12.
40. Aryeetey, E. O., & Garg, L. (2025, November). Leveraging AI and Machine Learning for Ransomware and Malware Detection in Cloud Environments: A State-of-the-Art Review. In *Conference on Internet of Things and Smart Spaces* (pp. 129-147). Cham: Springer Nature Switzerland.