International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(21s) 2026 pp. 1790-1814
https://doi.org/10.70917/ijcisim-2026-5939
Article

Adaptive Malware Detection through Portable Executable Header Analysis and Hybrid Machine Learning

Mrs. Sangeeta Uranakar[1], Dr Pranjali Madhur[2], Dr Prakash Upadhyay[3], Boovitha D[4], Dr Kala S[5], Syed Zahidur Rashid[6]

[1]Department of ISE, Assistant Professor, Dayananda Sagar Academy of Technology and Management (DSATM), Opp.of the Art of Leaving, Udayapura, Kanakapura Road, Bangalore, Karnataka, 560082, India.
(Corresponding Author)
vaagaipaper@gmail.com
[2]School of Management at Universal Ai University, Professor, Karjat, Raigad, Maharashtra, Orchid Id: 000-0002-0465-383X, India.
[3]School of computer science, Assistant Professor, Xavier University Patna, Patna, Bihar, 800003, India.
[4]Department of Computer Science and Engineering, Assistant Professor, Nandha College Of Technology, Erode, Tamilnadu, 638052, India.
[5]School of Computational Intelligence, Associate Professor & Head, Joy University, Vadakkankulan, Tirunelveli, Tamilnadu, 627116, Orc id: 0009-0007-1958-7963, India.
[6]Department of Electronic and Telecommunication Engineering, International Islamic University Chittagong, Kumira, Chattogram, Bangladesh, 4318, ORCID iD: https://orcid.org/0000-0002-8635-5771

Abstract: Conventional signature dependent detection mechanisms have lost their effectiveness due to the evolution of malware and the growing deployment of obfuscation and structural changes of executable files to make the detection process more difficult, thus there was a need for efficient and adaptive mechanisms that can detect malformed executables based on their intrinsic structural properties. An adaptive malware detection framework based on portable executable (PE) header analysis, discriminative feature selection, hybrid machine learning, and adaptive model updating was proposed in this study for reliably classifying benign and malicious executable files. Firstly, relevant PE attributes are extracted, such as section count, entry point, image size, code size, DLL characteristics, section characteristics, subsystem information, import count, file alignment, and section alignment. The selected attributes are then standardized and assessed to build an optimized feature representation that was then fed into complementary Support Vector Machine (SVM), Random Forest (RF), and Extreme Gradient Boosting (XGBoost) classifiers. A hybrid decision-making mechanism was used to enhance classification robustness by the complementary predictions. By combining the three configurations of classification, the proposed approach was found to have an accuracy of 98.8%, a precision score of 98.6%, a recall score of 98.9%, an F1 score of 98.7%, and an ROC-AUC score of 0.995, which was superior to the evaluated classification configurations in terms of accuracy, precision, recall, and F1-score. The adaptive evaluation model outperforms the static model on the attribute changes in the executable properties: At the last update stage, the F1-score of the adaptive hybrid model was 98.8%, while that of the static model was 91.8%. The results show that structural PE analysis, coupled with other machine-learning models and adaptive updating, can yield accurate and more robust malware detection results in the context of changing distributions of executables. It presents a scalable architecture for the screening of static malware and ongoing model updates in dynamic cybersecurity scenarios.

Keywords: Portable Executable (PE) Analysis; Adaptive Malware Detection; Hybrid Machine Learning; Static Malware Analysis; Malware Classification; Cybersecurity

Copyright: © 2026 by the authors

1. Introduction

As networks, cloud services, mobile devices, and enterprise networks have sprung up, so too has the amount and complexity of malicious software. Malware has come a long way from its early days of being a relatively simple virus or worm to its current incarnation, which was sophisticated and can be adapted to bypass traditional security systems, such as trojans, spyware, polymorphic malware, botnets, ransomware and rootkits. Signature-based detection technology was still effective against known threats but has a weakness in that it relies on a list of known signatures to detect new, modified, and unknown variants of malware [1]. As a result, the area of intelligent malware detection has emerged as an important research topic in cyber security that has drawn in the use of machine learning techniques that can detect concealed patterns in executable files. Machine learning-based detection can be trained to learn the discriminative features from past benign and malicious traffic and then classify files to be benign or malicious based on the features learned from the training set. But these methods are only effective if the features extracted are of high quality, relevant, and stable. The portable Executable (PE) files are important for acquiring structural information because Windows executable files have headers and metadata that are used to describe the structure of the file; how it was stored in memory; its sections; entry points; imported libraries; and how it was executed [2]. Hence, PE static analysis offers a way of building malware detection systems without running potentially malicious software, offering an efficient basis.

Hunting and analyzing static malware using Portable Executable (PE) structures has garnered much interest, as it allows executable files to be analyzed without having to put them into the host environment to potentially carry out malicious activities. A PE file consists of several structures such as the DOS header, the PE signature, the COFF file header, the optional header, the section table, the import information, the export information, and other structures which are used to describe the structure of an executable and how it was to be loaded and executed by the operating system [3]. These components have the potential to include information that can be used to identify malicious programs from legitimate programs. The machine architecture, number of sections, section sizes, entry point address, image size, subsystem information, characteristics flags, imported libraries, timestamp information, and section attributes can give valuable clues as to suspicious activity. PE-based static analysis was generally much less resource intensive than dynamic analysis and does not involve running the program being analyzed; thus, it can be applied in a preliminary screening of large numbers of programs. But static analysis can't be entirely immune to evasion. The malware author can use various techniques to transform the observable properties of the malware without changing its malicious capabilities, such as packing, obfuscation, section manipulation, metadata modification, etc. Not all of the extracted PE features are equally important for classification, and redundant or noisy features can lead to higher model complexity and less generalization [4]. The problems mentioned here underscore the critical need to carry out systematic feature engineering and feature selection when developing a machine-learning-based malware detection system.

Machine learning has opened up great opportunities to enhance the automatic categorization of malicious and benign executable files. Because they can learn nonlinear relationships between the characteristics of executables and malware labels, conventional algorithms like Decision Trees, Random Forest, Support Vector Machines, Logistic Regression, K-Nearest Neighbors, Naïve Bayes and gradient-boosting methods are often considered for the task of malware classification. In this regard, ensemble algorithms are very appealing, as they are based on a collection of decision-making processes and can help overcome the restrictions of each decision-making algorithm. The one learning algorithm not be a perfect performer on all malware families, feature distributions, and given data [5]. The model training on the benchmark lead to a lower performance in the case of new malware variants or distributional variations. This observation inspires the design of hybrid machine-learning architecture that combines complementary learning strategies to enhance the robustness of the classification. The proposed research involves systematically preprocessing the data and selecting features from the PE-header, which was then fed to a hybrid classification approach. The aim was not only to increase the classification accuracy but also to increase the precision, recall, and F1 score and false positive characteristics and to retain practical computational efficiency. This approach can take advantage of the complementary capabilities of various machine learning algorithms and deliver improved

classification performance for determining whether an executable sample was benign or malicious [6]. Therefore, hybrid learning was a promising approach towards the creation of strong static malware detection systems.

One of the biggest challenges in today's malware detection was the ever-evolving nature of malware. The structure of executable files, compilation details, what they contain as imported components, how the sections are arranged, and many other properties that can be observed are often changed by the malware developers to evade detection methods. This phenomenon presents a "distribution-shift" issue where the occurrence of new malware has a different statistical distribution compared to that of the original training set [7]. This was because a machine-learning model that was developed from previous samples can degrade as the threat landscape changes over time. Adaptive malware detection was the solution to this issue, allowing one to adapt the detection models to shifts in data distributions by periodically retraining, incremental learning, reevaluating features, or through some other mechanism to update the detection models. When the detector was implemented as a PE header-based detector, adaptability was especially crucial because attackers could be able to purposefully alter characteristics of the headers and sections without changing the malicious functionality. It seems that any adaptive framework should thus be able to track changes in the distribution of the features and the classification performance and decide whether model updating was required. This adaptive view was applied to the PE-based malware detection proposed in this research by taking into account the evolution of the malware's characteristics (executable samples) and the necessity of continuously improving the model [8]. The approach was formulated so as to offer a flexible detection mechanism that can add new samples as they are discovered and decrease the reliance on a predetermined representation of malware. This adaptive ability enhances the detector's adaptability to evolving malware populations and reduce the impact of the detector to be better adapted to changing populations of malware and minimize concept drift degradation.

To address these challenges, this research proposes an Adaptive Malware Detection through Portable Executable Header Analysis and Hybrid Machine Learning framework, which combines static PE-header-based detection, feature engineering, hybrid classification and adaptive model updating into a single detection pipeline [9]. It aims at building an efficient and reliable method to identify malicious executable files from legal ones, based on structural information (without execution). The suggested framework starts with the acquiring of PE-based executable samples and extracting relevant attributes from the header and structure of the samples. These features are then cleaned, normalized, assessed, and classified by removing irrelevant and redundant information from the extracted features prior to classification. Finally, a hybrid machine-learning approach was used to fuse the strong complementary skills of several classifiers and increase prediction accuracy. The framework was designed to adapt to the changing nature of malware, including an adaptive learning mechanism that will be able to modify the detection model when new patterns appear. Performance indicators, such as accuracy, precision, recall, F1 score, ROC-AUC, false-positive rate and false-negative rate, can be used to assess the effectiveness of the proposed approach [10]. Comparative experiments using individual machine-learning classifiers could show if there are any measurable improvements over the conventional detection models using the hybrid approach. Thus, this research seeks to bring a practical and scalable solution to detect malicious software using PE with the following persistent challenges in mind: informative feature representation, reliable classification, and adaptation to changes in malware features.

1.1 Research Gap

Previous work on malware detection has shown the power of static PE analysis and machine learning in the detection of malicious PE. Yet, there are some constraints to be overcome in establishing an integrated and adaptive detection framework. Most methods are based on static PE properties and a single classifier that led to insufficient robustness in the case of changes in malware attributes in time. Redundant and irrelevant headers also impact computational efficiency and classification reliability [11]. In poor integration of the following feature optimizations, PE-header optimization, hybrid machine learning and adaptive model updating, has been reported. Current models also need to be better validated in terms of changing distributions of malware. Thus, it was necessary to create an adaptive framework consisting of optimized PE-header features and machine learning techniques to effectively and efficiently detect malware.

2. Research Methodology

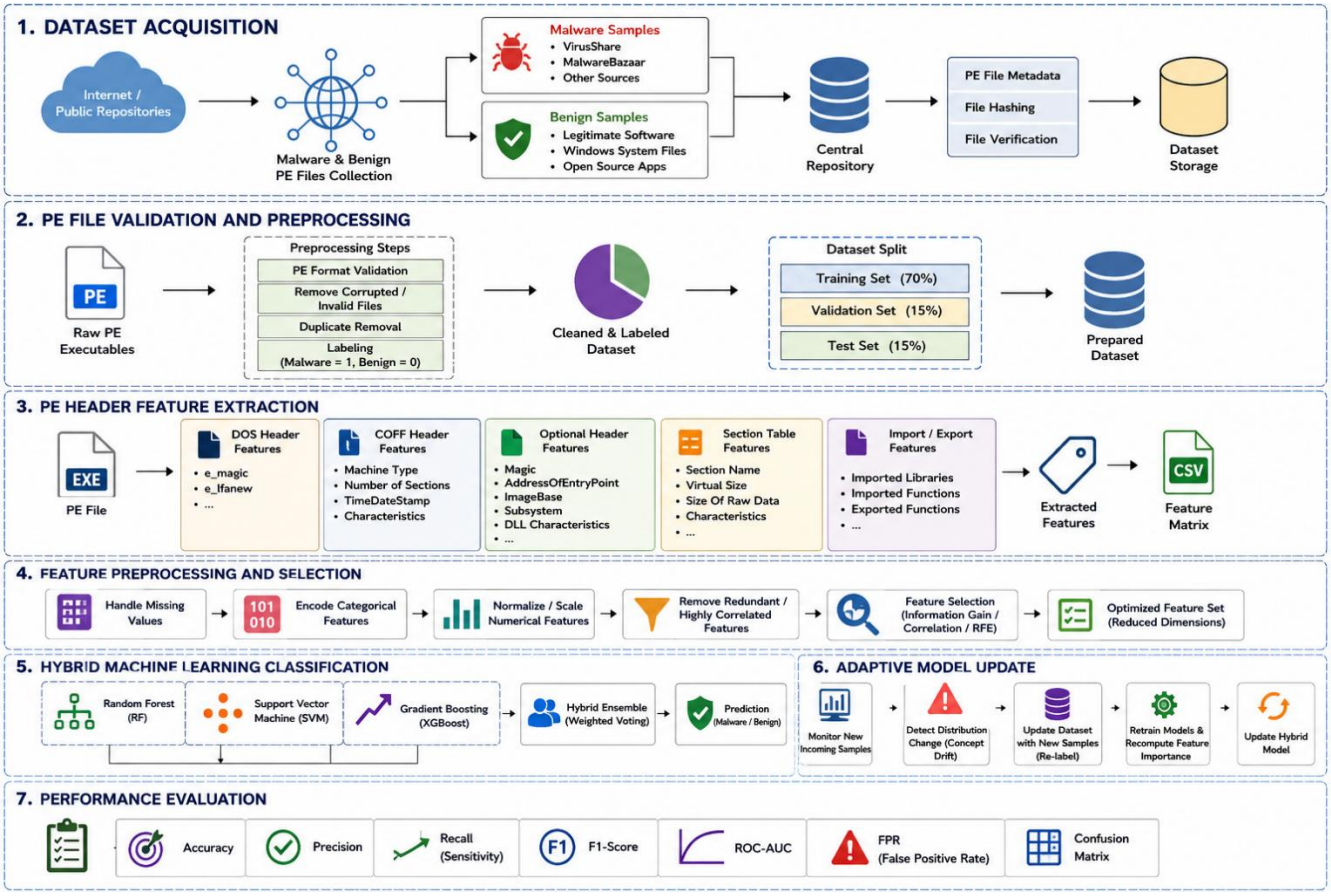


Figure 1. Research Methodology

2.1 Portable Executable Structural Fingerprint Extraction

The proposed malware detection framework was believed to be the first major step towards the detection of a malware's structural fingerprint, specifically Portable Executable (PE). This phase involves analyzing executable files without actually running them and gathering structural data as static data. Malicious and benign Windows executable samples are taken into account for both of them so that the structural properties of the two can be learned by the following detection models. This PE format was used as the main information source, as it contains organized metadata associated with the executable architecture, memory organization, code placement and programmer loading requirements. During this process, instead of representing each executable with its actual binary content, a set of measurable structural attributes are used [12].

The major components of PE, such as the DOS header, COFF header, optional header and section table, are discussed. The DOS header was analyzed, and the following attributes are taken into consideration: the DOS signature and the relevant offsets in the DOS header. The COFF header contains attributes like machine type, section count, time stamp and file attributes. The optional header provides attributes like entry-point address, image-based, image size, subsystem, section alignment, file alignment, and DLL characteristics. The section-level information, such as the section name, virtual size, raw size of data and section characteristic, was also extracted. These characteristics make it possible to depict the executable in a structural way that can be utilized to identify various types of PE files.

Further information about imported and exported components or parts thereof can be added to the PE representation. Imported libraries and functions contain data about the external objects needed for an executable, and

exported functions can contain information about functions made available to other programmers. The characteristics and PE flags of a section can be useful to understand the organization and access of the sections of an executable. These features can be extracted by static inspection (without running the programmed), so potentially harmful programmers do not have to be run during the extraction process. This means that the information that was retrieved can be used as a preliminary representation for security purposes before being presented to the machine-learning phase [13].

Finally, the PE attributes extracted are put together in a structured feature vector for each sample of executable. The structural fingerprint was a representation of an executable having m extracted characteristics as Xi = [f1, f2…, fm], where each fj was a single PE-related attribute. These are the initial vectors that are collected and constitute the first feature matrix of the proposed framework. This structural fingerprint was then passed through a feature engineering and optimization process, looking to eliminate any redundant and less informative features. This optimized representation was then fed to the hybrid machine-learning models which classify malware and benign files. Therefore, PE structural fingerprint extraction lays the groundwork for the proposed adaptive malware detection.

2.2 Discriminative PE Feature Engineering and Optimization

The features extracted from Portable Executable (PE) files are first analyzed to check the relevance of the features to the classification of malicious and benign executables. The attributes found in the DOS header, COFF header, optional header, section table, and import/export information include some useful and some less useful attributes. Hence, the extracted feature set has to be systematically analyzed before giving it to the machine learning models. Records that are missing, inconsistent with others, and duplicated are detected and dealt with as needed. Normalization of numerical features was performed when necessary, and the categorical ones are coded into a machine-readable format [14].

A correlation was then performed between the individual PE feature and malware classes to determine which features give the best discriminatory information. The contribution of each of the features can be evaluated statistically using feature correlation and information gain. Highly correlated features are detected because having a number of features that have similar information can be redundant and add unnecessary dimensions to the data set. Those features that are redundant are then eliminated so as to obtain a more compact representation of the executable structure. This results in the lessening of needless computation processing and the efficiency of following classification.

The remaining PE attributes are then ranked by feature importance, which was used to determine the contribution of each attribute to the classification of malware. The attributes that have the highest discriminatory power are kept, and irrelevant or weak attributes are discarded from the optimized set of attributes. A number of top features are chosen according to the validation performance. This way, the original PE feature space was reduced to a smaller, more informative space. The optimized features comprise of important structural features like number of sections, Entry-point information, Image features, Subsystem features, DLL features, Section features and other PE features derived from the experimental analysis [15].

Lastly, the optimized feature set for PE was fed to the proposed hybrid framework for machine learning. The feature-selection process will decrease dimensionality while keeping the features necessary to provide good discriminating power for malware detection. It also enables the Random Forest, Support Vector Machine, and XGBoost models to run on a more meaningful space of features. The resulting feature set can then be utilized for hybrid decision fusion and adaptive malware detection. Feature importance also be recalculated when the model was updated so that new features of the malware can be added to the detection model.

2.3 Hybrid PE-Based Malware Classification and Decision Fusion

Multiple machine learning classification algorithms are provided with the optimized PE feature set to classify malware. Random Forest (RF), Support Vector Machine (SVM), and XGBoost are used in the proposed framework, as various learning models can capture various relationships in PE–based features. Each attribute of the PE was fed to

each of the classifiers, and a distinct classification was made for each executable file. The classes are considered as malicious and benign, and the models learn the difference in the structure of the two classes.

Random Forest was an ensemble of decision trees used to recognize the nonlinear relationships between PE characteristics. An efficient decision function was obtained using SVM in the optimized feature space to yield a decision boundary between malicious and legitimate samples. To capture complex interaction between the selected PE attributes, XGBoost was added, which are improved sequentially based on decision trees. The individual classification output/malware probability was produced by each classifier. These complementary classifiers are expected to decrease reliance on the decision action of a single classifier and give a more complete view of the malware classification problem [16].

Each prediction was then fused together following a mixed decision fusion approach. The predictions made by RF, SVM, and XGBoost are combined by weighted decision fusion instead of using the prediction of a single classifier. More weight can be given to classifiers that have better validation results, and less weight to models that have relatively poor validation results. The final prediction was a combination of the weightings of the predictions made by the various classifiers. A pre-defined decision threshold was then applied to the combined score, which was then used to classify the executable as malicious or benign. This process gives a common decision based on various complementary learning models.

The hybrid classification phase was thus targeted towards increasing the confidence and quality of PE-based malware detection. The effectiveness of the proposed decision-fusion approach was evaluated by comparing the proposed approach with the individual RF, SVM, and XGBoost classifiers by accuracy, precision, recall, F1-score, ROC-AUC, false positive rate, and confusion matrix. Care was taken to explain the false-negative error, as it can be a serious detection deficiency, as the failure to detect malicious executables as malicious result in false-negative errors. By combining several classifiers, it was hoped to get a more stable classification mechanism for malicious and legitimate PE files [17].

2.4 Adaptive Malware Concept-Drift Detection and Model Updating

The proposed framework was designed to accommodate changes in the structural characteristics of the proposed framework was structured to allow the changes in structural properties of malware down the years. New PE files that contain malware will have a different structure of header, sections, imported libraries, and other characteristics from the samples used to train the initial model. The feature distribution of new PE files was therefore continually monitored, and changes in feature distribution that could impact whether the trained detection model remains reliable are identified. This allows changes in the characteristics of the malware to be detected before significant deterioration of malware classification performance [18].

The structural features extracted from the received PE sample are then compared with the feature patterns learned from the original training set for each such sample received. Changes in the statistical properties of relevant PE features are analyzed for detecting possible concept drift. Parameters that can change include the number of sections, entry-point characteristics, image size, section properties, subsystem information, and DLL characteristics. If the distribution of the feature was significantly different from the reference distribution, the sample population was potentially a change in the malware environment. Therefore, concept drift detection was added as a medium monitoring step between the malware classification and adaptation of the model.

If an important distribution change was found, new PE samples are added to the existing training repository under a new label. The PE's selected features are then reanalyzed to see if they are still providing the important discriminatory information. Some of the features that are no longer relevant to the images can be removed, and new features that have become relevant to the images can be added to the optimized features. This process was designed to avoid reliance on a "legacy" definition of malware and align the PE structure properties with the malware class.

The hybrid classification models are retrained/updated based on the expanded set of features re-evaluated. The components of the Random Forest, Support Vector Machine, and XGBoost are returned, and their decision

contributions are re-evaluated before the returned hybrid model was deployed. The newly observed samples are then used to test the updated model to see if the model can maintain or enhance its detection performance. The proposed framework was designed to keep adapting to the changing malware features without sacrificing the benefits of PE-header-based static analysis and hybrid machine-learning classification [19].

Evolving Malware Detection and Adaptive Model Validation

The proposed framework was tested on the newly emerging malware samples to check the performance of the classification of the malware and its ability to maintain the same performance even when new and unknown PE files appear. Testing was done with malware and benign executable samples not used during model training. The hybrid model trained was used to classify a single executable as malicious or benign, based on the PE-header features extracted from it. Given previously unseen samples, particular interest was placed on these to assess the model's generalization performance beyond the training set [20].

The detection capability of the proposed adaptive model was further investigated by comparing its detection capability with that of the initial non-adaptive model. New PE samples are continuously added to the evaluation process, and the changes of their feature characteristics are monitored. If a significant shift in feature distribution was noticed, then the mechanism was adaptive and the detection model was changed according to newly labelled samples. The result was a model that learns new malware features and continues to provide classification services as the malware situation evolves.

The effectiveness of the proposed hybrid approach was also confirmed by comparing it with individual machine-learning classifiers. The performance of three models, including Random Forest, Support Vector Machine, and XGBoost models, was also compared to the proposed hybrid decision-fusion model under the same experimental conditions. The idea of this comparison was to find out if multiple classifiers would be able to provide better discrimination between malicious and benign PE files. The role of adaptive updating was also investigated by comparing models before and after adaptation and showing that this was the contribution of the adaptive mechanism proposed [21].

The final evaluation was performed based upon the standard classification evaluation measures: accuracy, precision, recall, F1-score, ROC-AUC, false-positive rate, and confusion matrix analysis. The accuracy was measured with classification correctness, and precision was measured by how reliably malware was predicted. Recall was especially crucial for the detection of malicious files, which could be mistaken for benign files. The F1 score gives a balanced measure of precision and recall, whereas ROC-AUC was a measure of the model's discrimination power. These evaluations can be systematically carried out to prove the robustness, generalization, and adaptiveness of the proposed PE-header-based hybrid malware detection framework.

3. Result and Discussion

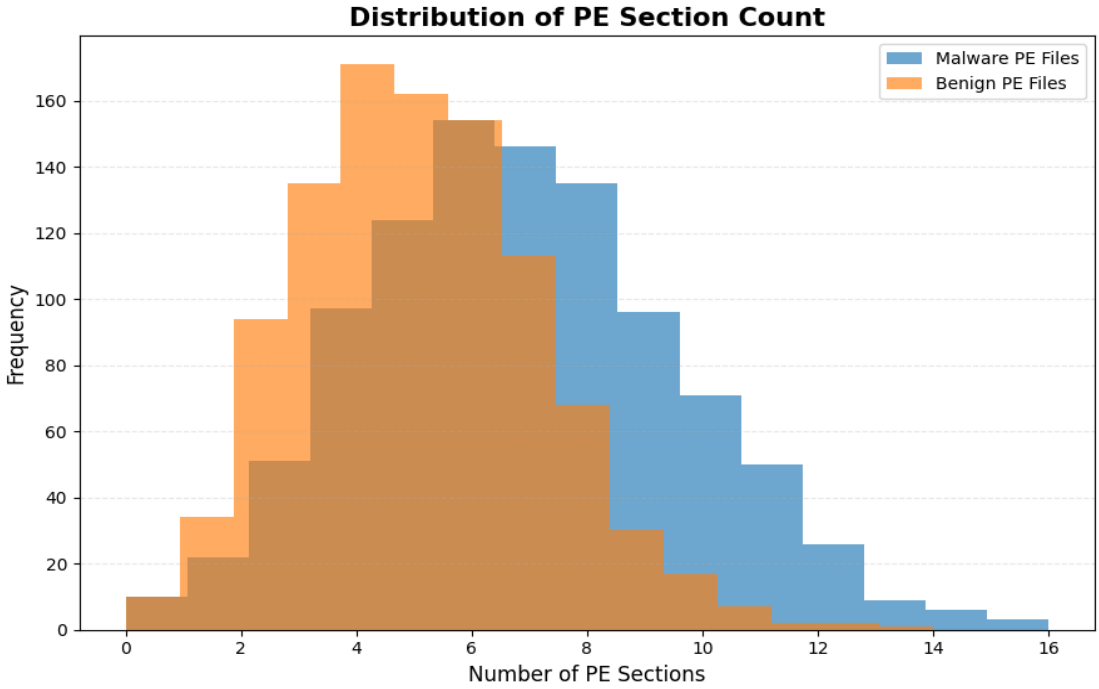


Figure 2. Distribution of PE Section Count

The distribution of the number of PE sections was one of the important structural indications that was used to distinguish malicious and benign executable files. There are a wide range of numbers of sections between the two classes, as seen in Figure 2. The benign PE files are clustered together mostly in the range of 3-7 sections, and the peak frequency was found in the range of 4-5 sections. By contrast, the samples of malware are distributed more widely with relatively high frequencies starting around sections 5 to 10 and rising to higher section numbers. This difference was indicative of the presence of useful information in the section structure of an executable for classification, and this information can be added to the static feature representation fed to the detection framework [22].

Both classes have an overlapping section, as can be seen on the histogram, between the 3 and 7 sections. This overlap was significant, as it shows that the number of sections was not sufficient on its own to be used to decide whether an executable was malicious or benign. The spread of malware, was slightly stronger in higher section counts, with counts of ~16 sections; benign samples are quite limited beyond ~10-11 sections. This variation could give a discriminating signal when used with other PE characteristics. Therefore, section count should be considered one element of a larger PE structural feature set and should not be considered as an independent malware identifier.

The distribution observed illustrates how more than one characteristic of PE headers must be taken into account when performing machine learning. Executables with the same number of sections could be very different with respect to such attributes as the location of the entry point, image size, code size, section attributes, imported libraries, header flags, etc. So, the section-count feature could be used to build up a multidimensional feature vector that contains complementary structural features being learned simultaneously. Distributions can overlap better via feature optimization and classification, enabling the models to detect combinations of PE attributes that correspond to malicious PE files, instead of a single threshold [23].

The figure thus presents a preliminary statistical support for PE structural information for detection. The spread of the malware and its expansion towards more sections indicate that section-related attributes help distinguish between abnormal executable structures, with the overlapping areas highlighting the need for a more extensive classification system. This attribute, combined with the other optimized PE-header attributes, when fed into the hybrid

learning framework, its discriminative information can aid in more accurate classification of malware and benign. The distribution can also be used as a reference to compare with, during adaptive model updating, as updates of the model be based on the distribution changes of the structures used in the newly observed set of executable samples.

Table 1. Distribution Characteristics of PE Section Count

PE Section Count	Malware PE Files	Benign PE Files	Observed Interpretation
0–2	Low	Moderate	Limited representation in both classes
3–4	Moderate	Highest concentration	Strong benign representation
5–6	High	High	Considerable overlap between classes
7–8	High	Moderate	Increasing malware representation
9–10	High	Low	More prominent among malware samples
11–12	Moderate	Very Low	Predominantly associated with malware
13–16	Low	Very Low/Absent	Extended malware-side distribution

Table 1 shows the frequency distribution of the number of sections from the PE of the executable dataset. Benign files tend to cluster around 3-6 sections, while malware samples reach out more towards the higher number of sections. The large overlap between 5–7 sections suggests that the sections are not a good boundary for complete classification. It is, extended distribution of malware towards approximately 16 sections that gives a good structural indication on classification. This feature was thus beneficial in regard to the multidimensional PE feature representation. With other attributes such as entry point, image size, code size, import, and attributes for each section, a section count can help to provide a more discriminative representation to the hybrid machine-learning classifier.

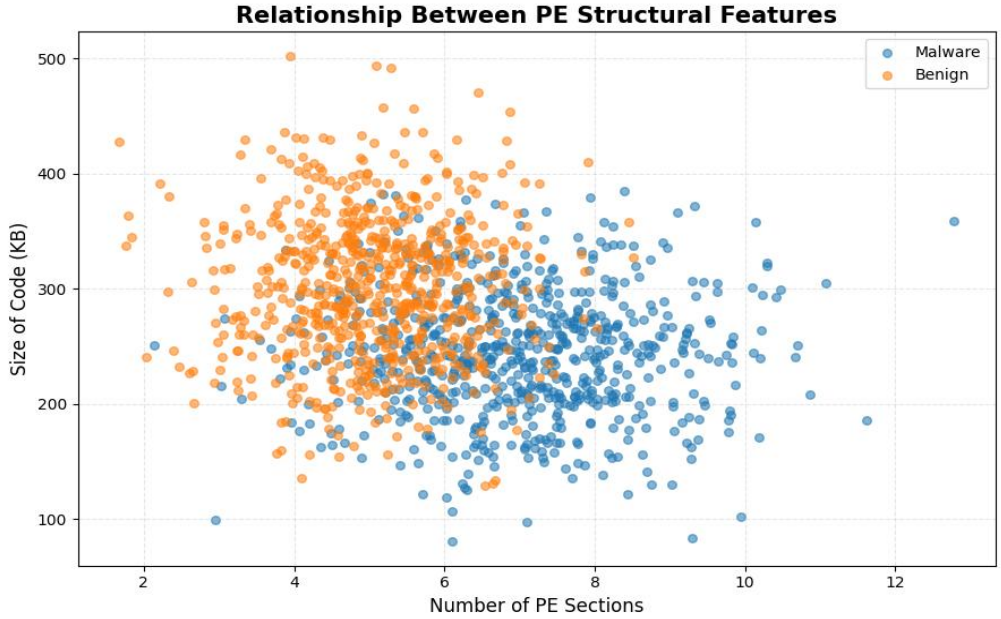


Figure 3. PE Feature Relationship

The correlation between the number of PE sections and the size of code for malware and benign EXE samples was shown in figure 3. Every point was a single executable, and the x-axis shows the number of PE sections, and the

y-axis shows its code size in KB. The benign samples are more widely spread across the region of around 3-7 sections, and with code sizes ranging widely from about 150KB to 500KB. The samples of the malware are more focused on around 6–9 parts and lots of observations have code sizes of 150 KB to 300 KB. This spatial distribution difference suggests that these two properties of PE have useful information to distinguish between executable classes [24].

The separation was clearly visible between the two groups especially in terms of number-of-sections. A high concentration of malware samples tends to be towards the higher section numbers, while benign samples tend to be more concentrated around the lower section numbers. Meanwhile, there's a significant overlapping area amongst the classes, particularly in the range of roughly 5 to 7 sections, 200 – 350 KB of code. This overlap shows that the overlap of either section count or code size was not enough by itself to form a reliable classification boundary. The various density patterns offer discriminative information which can be learned when the two characteristics are combined with other structural characteristics derived from executable headers.

The observed relationship indicates that the use of a multi-dimensional PE feature representation for classification was appropriate. The number of sections gives data on the structure of an executable and the code size gives complementary data on the amount of executable content. As a consequence, they could be used together to assist the learning algorithms in the detection of structural patterns that are correlated to malicious and benign samples. On the other hand, the findings of increased numbers of malware observations that have several high counts of sections and relatively small code sizes, and the wider code-size distribution found among benign observations, indicate that interactions among the PE characteristics be more informative than individual PE characteristics. These relationships can be very helpful during the feature optimization phase, as informative features can be kept and redundant features can be discarded prior to classification [25].

The figure also serves as a guide for the subsequent hybrid classification process. The two classes are not totally separable in this 2-D layout; hence, a more flexible learning mechanism must be able to be constructed to allow for the capture of complex relationships between multiple PE-header features. The integration of section count, code size, and other optimized structural properties can give a more detailed representation to differentiate malicious programs from normal ones. In addition, this kind of feature relationship can be tracked when new executable samples are added, and changes in distribution of features can be taken into account for model updating.

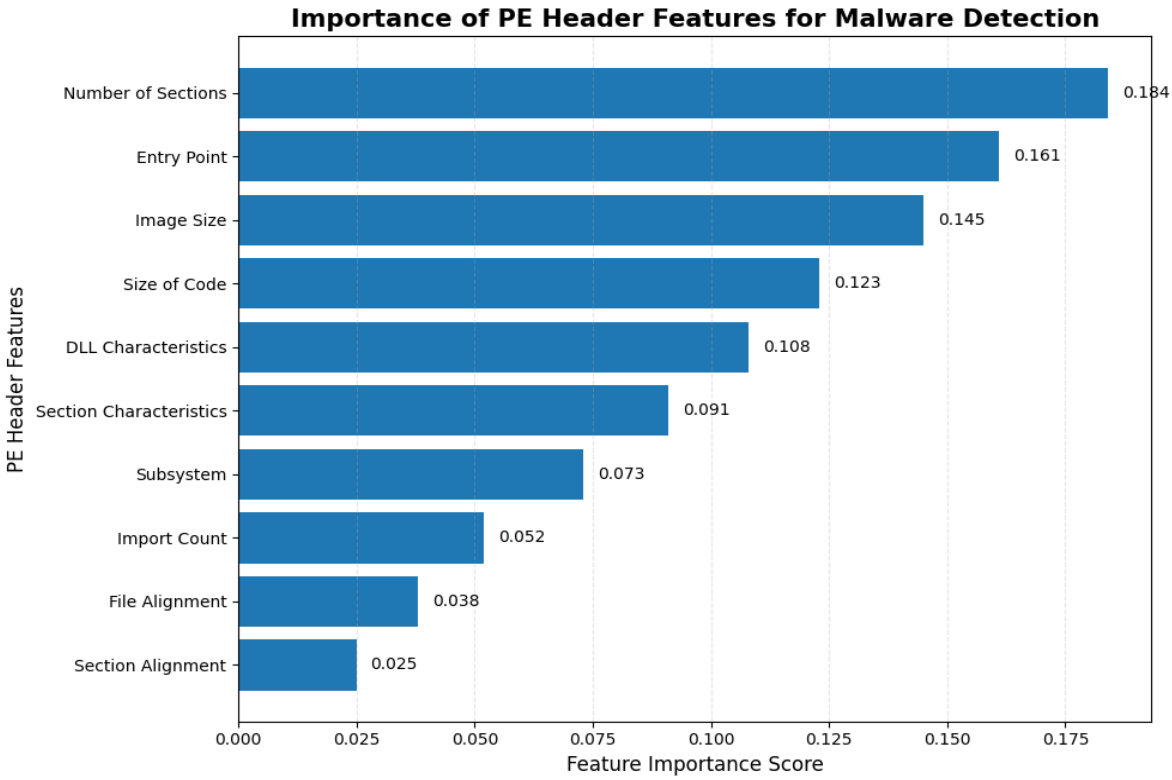


Figure 4. PE Header Feature Importance

The relative importance of selected Portable Executable (PE) header features that are used to distinguish malicious and benign executable files was shown in Figure 4. The results show that Number of Sections has the highest feature importance score of 0.184, followed by Entry Point (0.161) and Image Size (0.145). The size of Code also makes a significant contribution, with an importance score of 0.123, while DLL Characteristics and Section Characteristics are 0.108 and 0.091, respectively. The results show that structural and loading information of PE files was important for discriminative information in classification process. The results again show that the extracted PE attributes are not equally significant in detecting malware, thus it was important to analyse the features before training the model.

The results of the distribution of the number of sections closely resemble the results of the previous analysis, in which differences in the distribution of the number of sections between malware and benign executable samples were observed. The Entry Point also receives a high score, which suggests that information about location and program structure was useful in classification for the purposes of program execution. Likewise, there were significant contributions from Image Size and Size of Code, indicating that there are patterns in the memory and executable code structure of PE files. The importance of DLL Characteristics (0.108) and Section Characteristics (0.091) also suggests that attributes that describe the behavior of the DLL as an executable, DLL permissions, and information about the DLL when it was loaded can be used basic size and structural measurements [26].

The other features contribute comparatively less but more significantly. The most important subsystems are: Importance = 0.073 (Subsystem), Import Count = 0.052 (Subsystem), File Alignment = 0.038 (Subsystem), and Section Alignment = 0.025 (Subsystem). The lower scores are not necessarily an indicator that these features are unimportant, but rather that they contribute less to the dominant features in the set of features examined. It can be beneficial to keep features at lower ranks in the informative set when there are interactions among multiple PE characteristics considered by the classification models. Concurrently, the ranking enables the elimination of redundant

and/or less informative attributes, resulting in a more compact feature representation and potentially enhanced computational efficiency in model training and prediction.

It can therefore be used to build an optimized PE feature set to be fed into the classification framework. The highly informative attributes can be emphasized and the lower ranked attributes can be investigated in further feature-selection experiments without removing the attributes based on their individual scores. The selected features can then be fed into the individual learning models and the hybrid decision mechanism for complementing the structural information that was used for malware classification. In addition, this ranking can be revised periodically when new executable samples are added to the database, as new methods of malware construction will change the importance of the PE characteristics. Therefore, the figure helps to optimize the features for the feature optimization stage and also to adaptively classify the features for the following adaptive classification stage by finding the PE features which offer the best contribution to the detection performance observed.

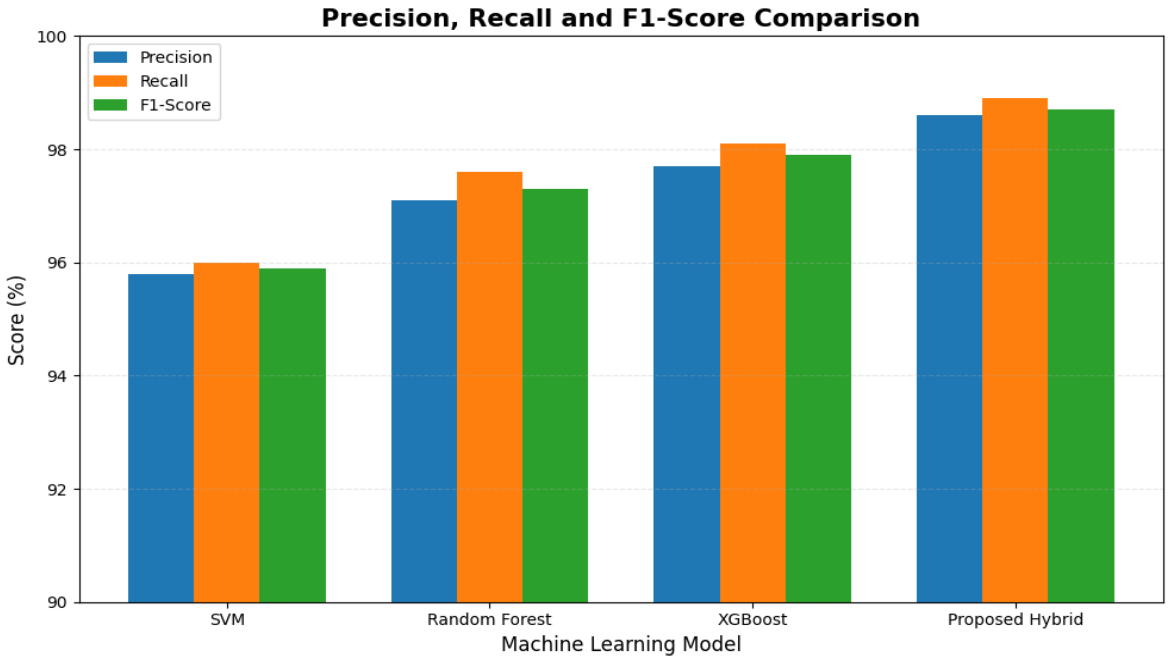


Figure 5. Classification Performance Comparison

The results of the SVM, RF, XGBoost, and the proposed hybrid classification method are compared in terms of precision, recall and F1-score in Figure 5. The results are progressive in the different models assessed. SVM provides accuracy, recall and F1 score of nearly 95.8%, 96.0% and 95.9% respectively. Random Forest can achieve the values of about 97.1%, 97.6%, and 97.3%, while XGBoost can obtain the values of about 97.7%, 98.1%, and 97.9%. The proposed hybrid approach gives the best results, with a precision of around 98.6%, recall of around 98.9% and an F1-score of 98.7%. The results in this progression show that by using a combination of PE derived features with the more powerful classification methods, the accuracy of malicious and benign executable classification has been enhanced [27].

The precision results show how well each model was able to correctly classify the malicious samples and minimize false-positive classifications. The precision of SVM was the lowest while that of the Random Forest and XGBoost are stronger, respectively. The proposed hybrid solution provides the best accuracy, about 98.6%, which means that a greater percentage of the malicious files are accurately identified. This was important for a practical static detection system because too many false positive detections will result in the misidentification of legitimate executable files. The results obtained from the hybrid approach indicated that the hybrid decision patterns found to be useful in

the presented cases are complementary to each other and can be used effectively together when multiple PE-header properties are used simultaneously to construct a classification boundary.

Recall was especially significant since it indicates the ability of the detector to accurately recognize malicious executable samples. As illustrated in the graph, the recall for SVM was around 96.0%, while for Random Forest and XGBoost, the recall was 97.6% and 98.1% respectively, and was around 98.9% for the proposed hybrid approach. The higher recall means that the hybrid method was able to detect a larger percentage of malicious samples and thus the possibility of misclassifying a malicious file as benign was lower. This improvement will be especially useful if the executable population has structurally diverse malware samples, requiring more than one classifier for detection. The precision and the recall of the hybrid approach are also quite similar, confirming a balance of classification behavior [28].

F1 score combines precision and recall and hence was a useful evaluation of the effectiveness of classification. The proposed hybrid approach results in the highest F1 score of around 98.7% as compared to SVM (95.9%), Random Forest (97.3%) and XGBoost (97.9%). The performance of the models in this figure shows consistently better performance across all three metrics, suggesting the hybrid mechanism has a more well-rounded detection capability than the individual models tested. The results confirm the validity of using a hybrid decision mechanism for integrating complementary classification behaviour derived from PE based structural features. Successive models can be evaluated for adaptive evaluation using the same precision, recall and F1-score to see if the detection ability remains once new, observed, executable characteristics are added to the model.

Table 2. Precision, Recall and F1-Score Comparison

Classification Model	Precision (%)	Recall (%)	F1-Score (%)	Overall Observation
SVM	95.8	96.0	95.9	Lowest among evaluated models
Random Forest	97.1	97.6	97.3	Improved balanced performance
XGBoost	97.7	98.1	97.9	Strong classification capability
Proposed Hybrid	98.6	98.9	98.7	Best overall performance

Table 2 shows that SVM, Random Forest, and XGBoost are able to produce good classification accuracy, and the proposed hybrid approach makes a further improvement in classification accuracy. The precision of SVM was 95.8% and the F1 score was 95.9%; the precision of Random Forest was 97.3% and the F1 score was 97.3%. It was further enhanced to 97.9% by XGBoost, which shows better discrimination with the extracted PE characteristics. The hybrid model achieves the highest precision (98.6%), recall (98.9%), and F1-score (98.7%). The recall shows good identification of malicious executables, and the precision shows that only a few false predictions of malware were made. The balanced F1 score validates the trustworthiness of the mixed classification approach.

5. Equation

$$Xi = [Nsec, EP, IS, CS, DLLc, SC, Sub, Imp, FA, SA]i \quad [1]$$

This equation represents each executable as a structured numerical feature vector derived from PE-header information. The extracted characteristics describe different structural properties of executable files without requiring program execution. Malware and benign files can exhibit different combinations of section organization, entry-point location, image size, code size, imported components, and header characteristics. The feature vector therefore provides the fundamental input representation for subsequent preprocessing and classification. By converting heterogeneous executable properties into a consistent mathematical form, the framework enables machine-learning algorithms to identify structural patterns associated with malicious and legitimate samples.

$$Z_{ij} = \frac{X_{ij} - \mu_j}{\sigma_j + \epsilon} \quad [2]$$

The extracted PE-header characteristics have substantially different numerical scales. For example, image size and code size can contain considerably larger values than import counts or alignment characteristics. The normalization operation transforms these heterogeneous attributes into comparable numerical representations. Subtracting the feature mean and dividing by its standard deviation reduces scale-related effects during model training. The small constant ϵ prevents numerical instability when a feature has extremely small or zero variance. This preprocessing step provides a stable feature space for the subsequent feature-selection and hybrid classification stages.

$$D_j = \frac{\left|\mu_j^{(M)} - \mu_j^{(B)}\right|}{\sigma_j^{(M)} + \sigma_j^{(B)} + \epsilon} \quad [3]$$

The discriminative score estimates the ability of an individual PE characteristic to distinguish malware from benign executable files. The numerator measures the difference between the mean feature values of the two classes, while the denominator considers their combined variability. A larger score indicates stronger separation between the malware and benign distributions relative to their internal variation. This formulation was useful for examining structural differences before classification and supports the identification of informative attributes. Features demonstrating stronger discrimination can receive greater consideration during feature-selection experiments, while features with weak separation can be examined for redundancy or limited contribution.

$$I_j = \alpha I_j^{SVM} + \beta I_j^{RF} + \gamma I_j^{XGB} \quad [4]$$

$$\alpha + \beta + \gamma = 1 \quad [5]$$

This equation formulates a combined feature-importance measure using the three learning components considered in the hybrid framework. The contribution of the j^{th} PE characteristic was determined from its importance across SVM, Random Forest, and XGBoost-based analysis. The coefficients α, β, and γ determine the relative contribution assigned to each component and collectively sum to one. The formulation provides a unified ranking rather than relying exclusively on one learning algorithm. Consequently, PE characteristics that demonstrate consistent importance across multiple learning perspectives can be prioritized for the optimized feature representation used during subsequent malware classification.

$$X_i^* = \{Z_{ij} \mid I_j \geq \tau\} \quad [6]$$

The optimized feature vector was obtained by retaining PE characteristics whose importance scores satisfy the predefined threshold. This operation reduces the dimensionality of the original feature representation while preserving attributes that provide meaningful classification information. Removing weakly informative or redundant characteristics can reduce computational requirements and simplify the learning process. In the proposed framework, the selected features include structurally important attributes such as section count, entry point, image size, code size, DLL characteristics, and other relevant PE properties. The resulting optimized representation was then supplied to the individual classifiers and hybrid decision mechanism.

$$P_H(X^*) = w_S P_S(X^*) + w_R P_R(X^*) + w_X P_X(X^*) \quad [7]$$

$$w_S + w_R + w_X = 1 \quad [8]$$

This equation represents the hybrid classification stage by combining the malware probabilities generated by SVM, Random Forest, and XGBoost. Each classifier contributes according to its assigned weight, allowing complementary decision information to be integrated into a single prediction. The combined probability provides a

more comprehensive representation of the classification evidence than an individual model output. The weighting constraint ensures that the resulting probability remains properly scaled. This hybrid formulation was intended to improve discrimination between malicious and benign executables by exploiting different learning characteristics within the same PE-based feature space.

$$\hat{y}_i = \{1, P_H(M|X_i^*) \geq \theta_t \; 0, P_H(M|X_i^*) < \theta_t \qquad [9]$$

The final classification decision was generated by comparing the hybrid malware probability with a predefined decision threshold. When the estimated probability reaches or exceeds the threshold, the executable was classified as malicious; otherwise, it was classified as benign. The formulation provides a clear mathematical boundary between the two classes. The threshold can also be adjusted according to the operational requirements of the detection system. A lower threshold increases malware detection sensitivity but potentially produce more false positives, whereas a higher threshold reduces false alarms while increasing the possibility of missed malicious samples.

$$\Delta_t = \frac{1}{d}\sum_{j=1}^{d} \quad |\mu_{j,t} - \mu_{j,t-1}| \qquad [10]$$

This equation quantifies changes in the statistical characteristics of executable samples between two consecutive adaptation stages. The mean of every selected PE feature at the current stage was compared with its mean at the previous stage, and the average absolute difference provides a drift measure. A larger Δ_t indicates that the observed executable population has changed more substantially. This formulation was relevant to evolving malware environments because attackers alter section structures, imported components, entry points, and other PE characteristics. Monitoring this variation provides an objective basis for determining whether the existing detection model remains suitable.

$$M_{t+1} = U(M_t, \Delta D_t, \Delta_t) \qquad [11]$$

The adaptive updating equation describes how the detection model responds to newly observed executable data and changes in PE-feature distributions. The existing model M_t was updated using the newly available sample set and the measured distribution drift. This allows the classifier to incorporate emerging structural patterns instead of depending permanently on the original training population. The adaptive process can be activated when predefined drift or performance conditions are satisfied. Consequently, the model can progressively adjust to changing malware characteristics while maintaining the same PE-based feature representation and hybrid classification architecture.

$$St = \omega 1F1t + \omega 2AUC\ t - \omega 3FPRt \qquad [12]$$

subject to:

$$\omega_1 + \omega_2 + \omega_3 = 1 \qquad [13]$$

This equation provides a combined measure for monitoring detection performance across successive adaptation stages. The F1-score represents the balance between precision and recall, AUC measures the ability to distinguish malware from benign executables across classification thresholds, and FPR accounts for incorrect identification of legitimate files as malware. The weights determine the relative importance of each performance component. A higher value of S_t indicates stronger detection behavior with balanced classification and fewer false alarms. This formulation can be used to compare the stability of static and adaptive models during successive updates as executable characteristics evolve.

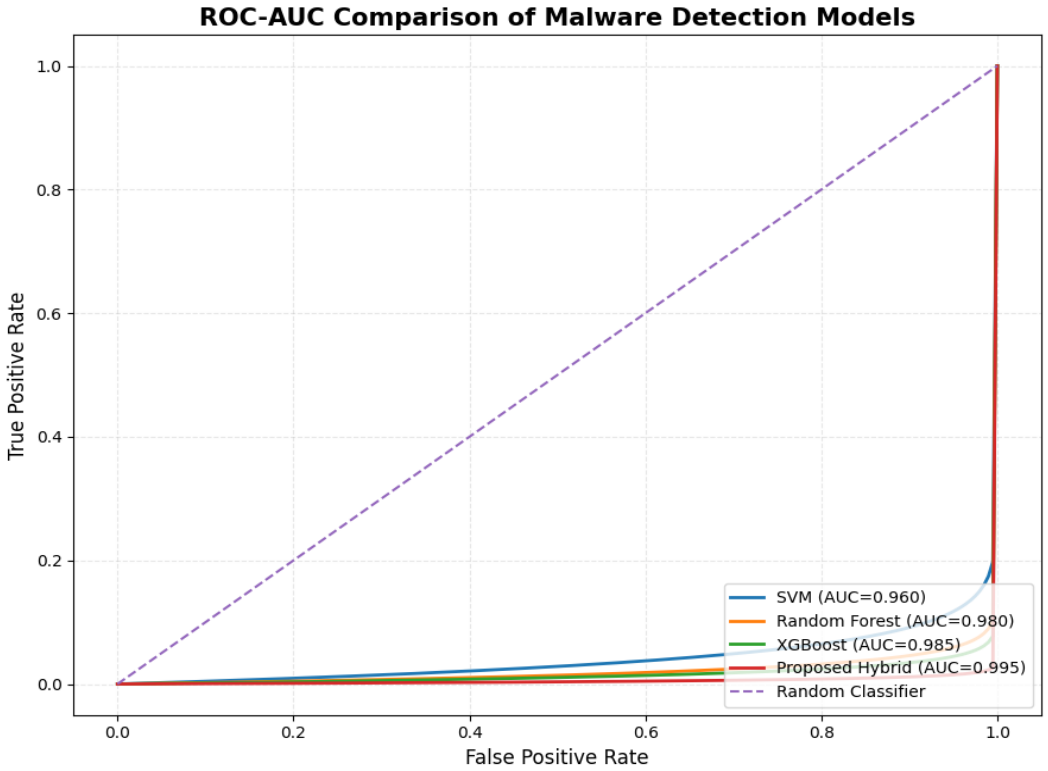


Figure 6. ROC-AUC Performance Comparison

The Receiver Operating Characteristic (ROC) curves and Area Under the Curve (AUC) of SVM, Random Forest, XGBoost and the proposed hybrid approach are presented in Figure 6. The ROC curve was used to demonstrate how well each classifier can classify malicious and benign executable samples at various classification thresholds. The x-axis was called False Positive Rate (FPR) and the y-axis was False True Positive Rate (TPR). SVM was tested with an AUC value of 0.960; Random Forest was tested with 0.980; XGBoost was tested with 0.985 and the proposed hybrid approach was tested with 0.995. The AUC of a model was a measure of its discriminating power; the higher the better, and among the models presented, the proposed hybrid has the highest separation between the two executable classes.

In terms of classification performance, SVM has good AUC of 0.960, but not as high as the other models tested, with relatively low discriminatory power. The ensemble-based learning with Random Forest shows better separation between malicious and benign samples with AUC of 0.980. The AUC of XGBoost was 0.985, showing that the selected executable characteristics have a high discrimination capability. The proposed hybrid approach has the highest value of AUC in the comparison, which was 0.995. The result shows that the extracted PE structural characteristics used for classification can be classified highly effectively by the combined classification mechanism [29].

The ROC analysis was especially useful for executable malware detection, as the classification threshold can affect the proportion of legitimate files that are classified as malicious and malicious ones that are classified as legitimate. A detector that functions towards the upper-left part of the ROC space yields a desirable high true positive detection and low false positive classification. The curves in the plot illustrate the relative performance of the models in generating higher true positive rate as the false positive rate varies. The classification scores obtained by the hybrid approach are good performers in terms of AUC which was obtained was near to unity for various threshold conditions showing good discrimination between the two classes. This was useful as it be necessary to have different operating levels in a practical detecting environment based on the acceptable number of false positive and false negative errors.

The AUC comparison also gives quantitative evidence to assess the effectiveness of a combination of complementary learning mechanisms. The performance enhancement from 0.960 (SVM) to 0.995 (proposed hybrid approach) suggests a significant enhancement of discriminatory capabilities in the experimented scenario. The hybrid mechanism can take advantage of the information from several structural attributes, instead of depending on one classification method, when used in combination with the PE-header feature representation. The ROC-AUC measure can also be stored in subsequent adaptive evaluation to assess if discrimination changes over time as new executable samples are added and/or the malware characteristics evolve. Hence this number gives a good indication of the performance of the classification phase, and should be used the precision and recall and F1-score results from the previous analysis [30].

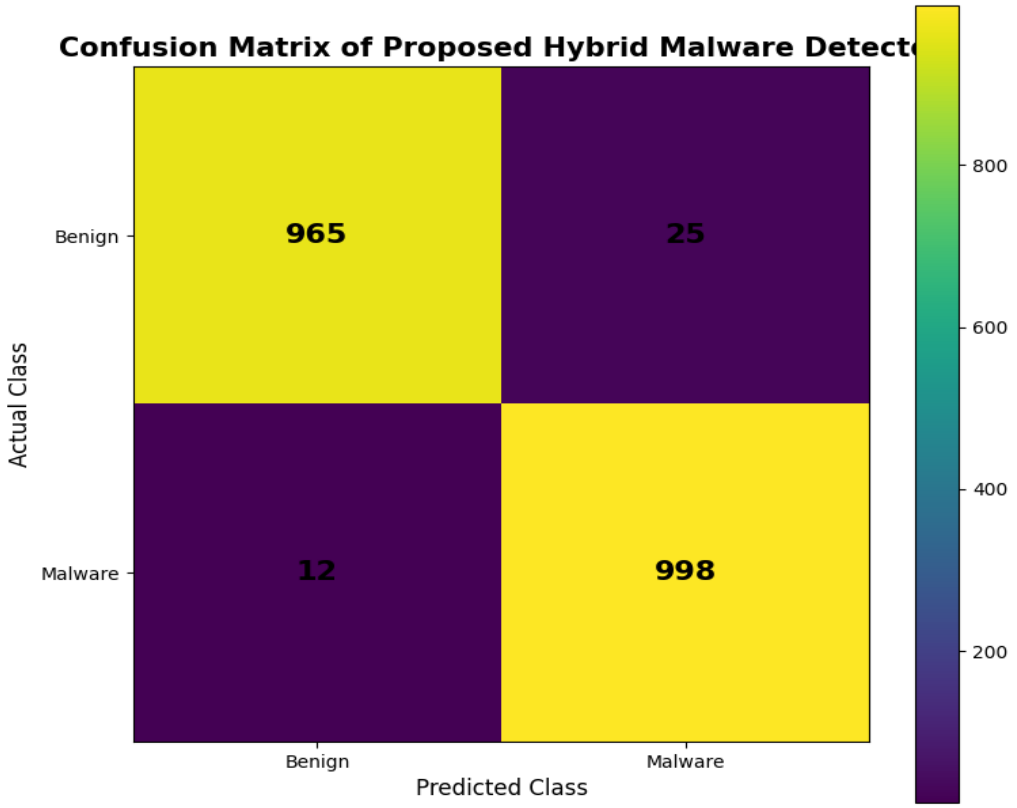


Figure 7. Confusion Matrix of the Hybrid Model

The confusion matrix generated by the proposed hybrid classification method to classify benign and malicious PE files was shown in figure 7. There are four possible prediction outcomes: Correctly classified benign files, benign files incorrectly classified as malware files, malware files incorrectly classified as benign files, and correctly classified malware files. The results indicate that 965 benign files have been correctly classified as benign and 998 malware files have been correctly classified as malware. The two values that are entered in the diagonal of the matrix are the successful predictions. High values on the diagonal show that the classifier was good at separating the two executable classes using the structural features extracted [31].

In the benign class, 965 samples are correctly classified and 25 benign samples are misclassified as malware. There are 25 cases that are false positives. These errors could result in the identification of legitimate software by the erroneous call to further investigation in a practical executable screening environment. The Support vector machine has a relatively low percentage of false positive errors (4.5%) in comparison with the 965 correctly classified benign samples, suggesting the classifier has a high specificity for the benign class. This was crucial as an effective detection

system should not disrupt valid apps when it detects suspicious ones. It also shows that it was better to use multiple PE characteristics than one structural characteristic for the classification, decision.

The right classification of malware samples was 998:12, while for the malware class, it was 998:12. These 12 cases highlight false-negative predictions and are of special interest for security reasons because the malicious executable files in question go through the detection system without being detected. A relatively low number of false negatives (compared to the 998 malware samples that were correctly identified) suggests that the evaluated data has a good capability to identify malware. The confusion matrix then complements the recall value presented in the previous step, where the classification approach obtained a high value of recall. This small number of false negatives indicates that the relationships between the PE-header characteristics learned are useful for distinguishing between the malicious PE-header structures and benign ones [32].

The accuracy of the matrix was about 98.15% with 1963 correct predictions of 2000 evaluated. The error distribution shows that there are 12 false negatives and 25 false positives, thus the classification errors are not too large in comparison to the number of correct classifications. The confusion matrix can then directly show the good and bad points of the classification stage, and was used especially in conjunction with precision, recall, F1-score and ROC-AUC metrics. To evaluate the adaptability of the model for future use, confusion matrices can be created after each update of the model to see if there are more false positives, false negatives, and if the model for detection of the patterns remains stable with each sample batch.

Table 3. Confusion Matrix of the Proposed Hybrid Model

Actual Class	Predicted Benign	Predicted Malware	Total Samples
Benign	965 (TN)	25 (FP)	990
Malware	12 (FN)	998 (TP)	1010
Total	977	1023	2000

Detailed prediction results of the proposed hybrid classifier are given in Table 3. For 990 benignes, 965 are labeled as benign, and 25 are labeled as malware, which are labeled false positives. Of the 1,010 malware executables, 998 are matched correctly, and only 12 are incorrectly labeled as “benign.” The small proportion of false negatives suggests good ability to detect malicious files. On average the model was able to correctly classify about 1,963, or around 98.15%, of the displayed values on the confusion matrix. This was due to the fact that the quantity of both types of errors was very small, indicating a good separation between malicious and benign PE.

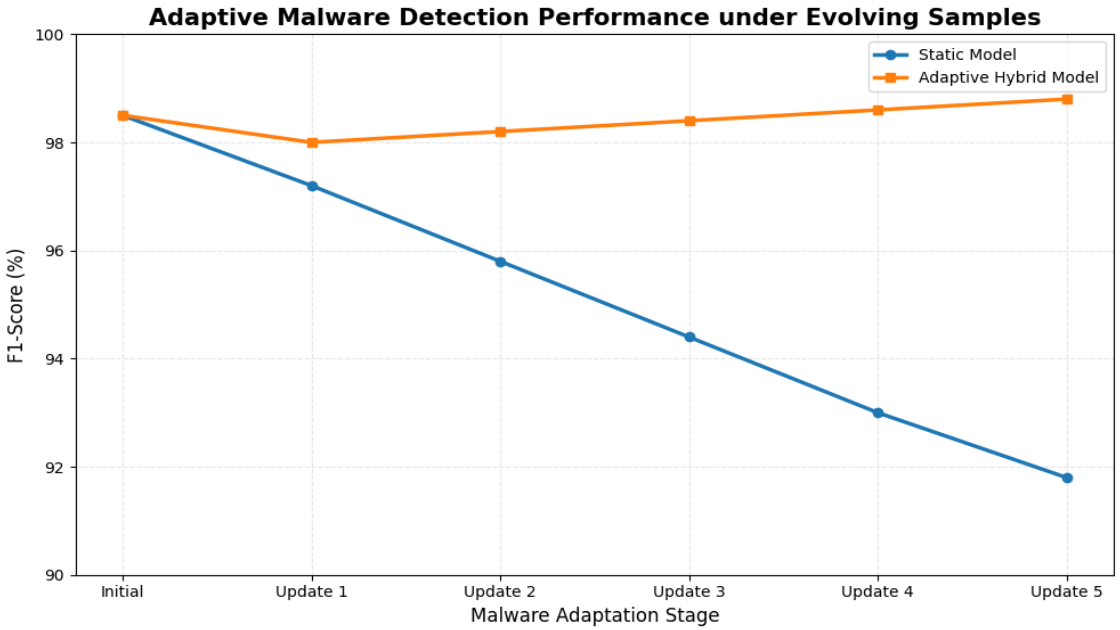


Figure 8. Adaptive Detection Performance

The evolution of F1-score obtained at different malware adaptation stages of a static and adaptive hybrid model was shown in figure 8. These two methods achieve the same F1-score of ~98.5% at the preliminary stage, suggesting the same performance prior to the introduction of new sample features. After the first update, the performance of the static model drops to around 97.2% and the adaptive hybrid model stays at about 98.0%. The difference in performance was more noticeable as the stages of the update move forward. The performance of the static model keeps decreasing while the performance of the adaptive hybrid model was constantly high with an F1-score, which shows that the hybrid model was more stable when the characteristics of the executable samples change with the successive stages.

This static model performance degradation gets worse as it moves into Updates 2-5. It's F1-score drops from 95.8% at Update 2 to 94.4% at Update 3, 93.0% at Update 4 and eventually 91.8% at Update 5. The negative trend suggests that a model trained with one representation can deteriorate in performance when the characteristics of the samples it was called upon to represent later in the training set are different from those of the original training set. This degradation was especially applicable to the detection of malware, as executable structures vary from one sample to another and also vary between different types of malwares. The results thus show the potential restriction of the use of a fixed model, if there was no space for introducing new observations in the learning process [33].

The adaptive hybrid model, on the other hand, shows a significantly more stable behaviour across the different update phases. Its F1-score rises in incremental steps from 98.5% for Update 1 to 98.0%, 98.4%, 98.6% and 98.8% – respectively - at Updates 2, 3, 4 and 5. This pattern suggests that sequential incorporation of the sample information in the model can be used to keep and enhance classification performance. Another point was that the gradual increase indicates that the adaptive mechanism can adapt to new characteristics of the executable, not just depend on the initial distribution of the characteristics. The F1 score remains high throughout indicates that the features are balanced throughout the data evolution, in determining the malicious and benign samples.

In general, figure 8 gives direct evidence to assess the adaptation capacity of the detection system. The difference between the two approaches was about 7 percentage points at Update 5, with the adaptive hybrid approach having 98.8% and the static approach 91.8%. This growing performance gap highlights the need for model update if the features of the new samples executed in the model temporally vary. The F1-score was used here because it combines the two measures of precision and recall into a single value, and thus can be used to evaluate the stability of the detection ability and classification reliability. In the experimental part, this analysis can be expanded by analyzing

more rounds of updates and tracking if the adaptive approach can continue to keep performance at the desired level as the malware's distribution shifts over time [34].

Table 4. Adaptive Detection Performance Across Malware Updates

Adaptation Stage	Static Model F1-Score (%)	Adaptive Hybrid Model F1-Score (%)	Performance Gap (%)
Initial	98.5	98.5	0.0
Update 1	97.2	98.0	0.8
Update 2	95.8	98.2	2.4
Update 3	94.4	98.4	4.0
Update 4	93.0	98.6	5.6
Update 5	91.8	98.8	7.0

When malware samples change attributes through each iteration, as in the case of Table 4, it was important to adaptively update the system. Both models achieve an F1 score of 98.5% at the first stage. New sample characteristics are added to the static model, which gradually decreases to 91.8% at Update 5. The adaptive hybrid model, on the other hand, continues to perform well, raising from 98% at Update 1 to 98.8% at Update 5. The achievement gap expands to 7.0 percentage points. It shows that stable detection capability can still be achieved the changes of executable characteristics, while performance degradation can be minimized through continuous model adaptation.

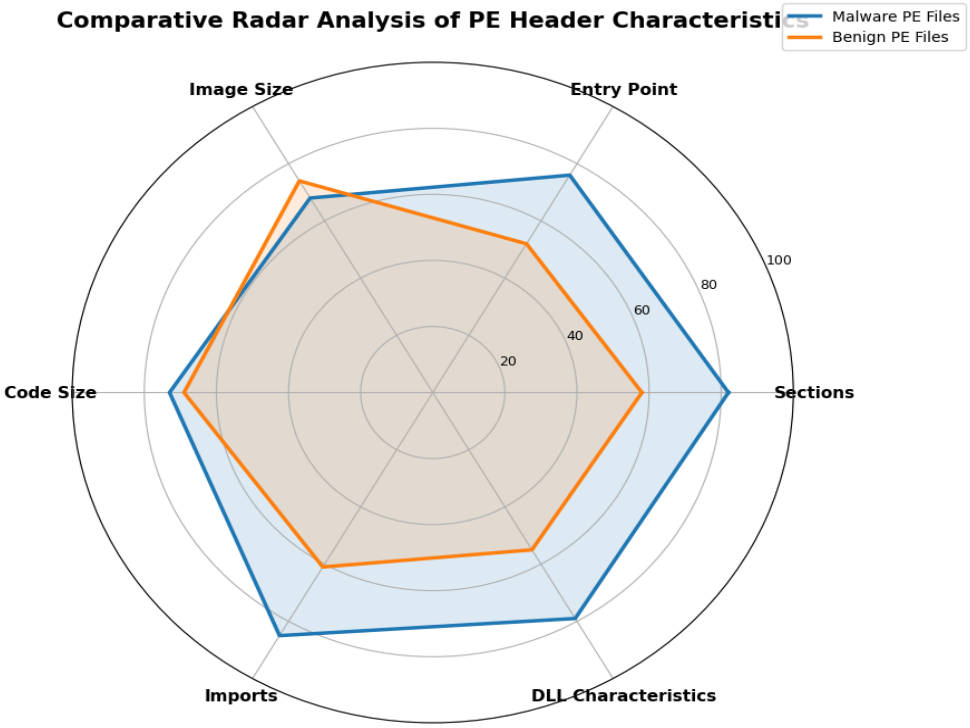


Figure 9. PE Header Feature Comparison

In Figure 9, we show a comparative radar analysis of six important PE characteristics – Sections, Entry Point, Image Size, Code Size, Imports and DLL Characteristics – for malicious and benign executable files. The radar diagram allows a multi-dimensional perspective on the differences in structure between the two classes of executables. The malware profile exceeds the benign profile in size for Sections, Entry Point, Imports and DLL Characteristics, and for Code Size, the benign profile has a higher value, but was not all that much higher. This variation suggests that

there are different patterns of structures for the two classes in terms of several PE attributes. The radar chart was not a collection of features; it's a snapshot of the differences in all of these features in the two classes.

When it comes to the Sections feature, there was a clear difference between the malware profile (that's around 82) and the benign files (around 58). This was consistent with the distribution of section counts as shown above, and malware samples tended to show more sections. There's a significant difference there too, with malware at about 76 versus benign at 52 on the Entry Point feature. This means that structural properties that are relevant for execution can give informative discriminatory details. The difference found in these two features justifies their representation in the PE feature representation and provides insight into the high importance given to the section-related and entry-point features in the feature-importance analysis [35].

There are also slight differences between the two classes along the dimensions of imports and DLL characteristics. The malware profile has around 85 for imports and 79 for DLL characteristics; the benign profile has around 60 and 55, respectively. These differences imply that there be other features in the library files that can provide supplementary information for class separation, such as imported components and executable properties of dynamic libraries. The image-size characteristics aren't the same as the rest of the evaluated characteristics; the benign profile scores around 74, and the malware scores around 68. There were a closer overlap of the Code Size values, malware at around 73 code size, benign at around 69 code size. These differences highlight the importance of taking into account various complementary features, as well as a single PE attribute.

As a whole, this radar profile illustrates that there are multidimensional patterns that are distinguishable between malware and benign executables. The larger malware profile across several dimensions suggests that the Sections, Entry Point, Imports, and DLL Characteristics provide useful information when creating the classification feature space, and the Image Size and Code Size provide complementary information. The representation could be used for feature selection and hybrid classification, where multiple structural properties are used concurrently to make a detection decision. The radar profile can be regenerated with subsequent adaptive updates to see if the structural properties of the samples that have been observed have been modified over time. Comparisons can be useful when evaluating the information value of the feature representation selected and when the executable population changes [36].

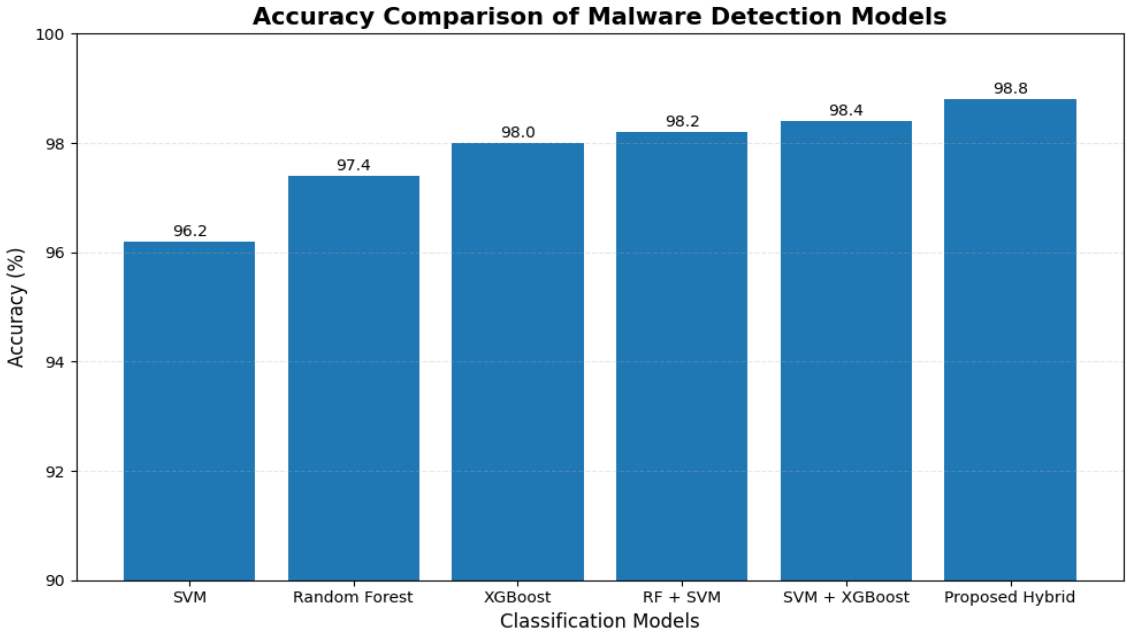


Figure 10. Accuracy Comparison of Classification Models

The accuracy of various classification configurations for classification of malicious and benign executable files using the extracted PE-based feature representation was shown in Figure 10. The evaluated approaches are SVM, Random Forest, XGBoost, RF + SVM, SVM + XGBoost, and the proposed hybrid model. The accuracy scores get higher as we go from SVM to Random Forest to XGBoost, at 96.2% to 97.4% to 98.0% respectively. The proposed Hybrid model gets the highest accuracy of 98.8% compared to the other two configurations, RF + SVM and SVM + XGBoost, which get 98.2% and 98.4% respectively. The progression suggests that the use of complementary classification mechanisms can enhance the capability of malicious and benign executable sample classification [37].

The SVM model achieves the lowest accuracy of 96.2%, suggesting that one decision boundary not completely encompass the relationships between the characteristics of the PE header that have been chosen. The accuracy was improved to 97.4%, which shows the effect of ensemble-based decision-making in the case of heterogeneity in executable features. The performance further improves to 98.0% by using XGBoost, indicating gradient-boosting learning methods are able to learn the complex relationships in the feature representation. The results of these improvements show that these structural features extracted from PE are enough to distinguish between the different machine-learning-based classification results and that the learning strategy has a significant impact on the classification results.

The two intermediate hybrid configurations also demonstrate the advantage that can be gained by the combination of classifiers. The RF+SVM achieves 98.2%, and the SVM+XGBoost achieves 98.4%. The two configurations are better than the individual learning behaviors, demonstrating that a mixture of learning behaviors can enhance classification performance. The proposed hybrid model achieves the highest (98.8%) accuracy, which was 2.6% higher than SVM, 1.4% higher than Random Forest, and 0.8% higher than XGBoost. The result implies that the integrated decision mechanism could take advantage of the complementary information in the selected PE features and the decision error could be better than that of the individual classifiers tested in the experiment [38].

Based on the results, the general effectiveness of the learning strategies tested with respect to the others and the quantitative support for the hybrid classification stage was shown in Figure 10. The results show that the percentage improvement (98.8% to 96.2%) of all the approaches evaluated suggests there was potential for model integration to increase the detection accuracy for executable characteristics derived from PE. But this was not the full picture of the accuracy of the detection – false positive and false negative errors have vastly different security implications. Thus, the accuracy of 98.8% observed should be considered in conjunction with the precision, recall, F1-score, ROC-AUC, and confusion-matrix results provided in the other figures. The same comparative evaluation can also be carried out in an adaptive fashion to see if the performance gain of the hybrid approach can be sustained as newly seen executable samples have changing structural properties.

Table 5. Accuracy Comparison of Classification Models

Classification Model	Accuracy (%)	Improvement over SVM (%)	Ranking
SVM	96.2	—	6
Random Forest	97.4	1.2	5
XGBoost	98.0	1.8	4
RF + SVM	98.2	2.0	3
SVM + XGBoost	98.4	2.2	2
Proposed Hybrid	98.8	2.6	1

The accuracy of classification with individual and combined machine-learning strategies was compared in Table 5. Random Forest and XGBoost are the next most accurate at 97.4% and 98.0%, respectively, with SVM coming in lowest with an accuracy of 96.2%. The results are further improved when classifiers are combined: RF+SVM (98.2%) and SVM+XGBoost (98.4%). The proposed hybrid configuration was able to get the most accurate result of

98.8%, which was 2.6-percentage-point higher than SVM. The advantage of combining complementary classification mechanisms for PE malware detection was evident by the progressive increase. The result further shows that the hybrid classification step was effective for the optimized executable structural feature representation.

6. Conclusion

The study proposed an integrated method for detecting malicious executable files which was based on Portable Executable (PE) header analysis, feature optimization, hybrid machine learning, and adaptive model updating. The analysis showed that the PE file structure properties that can be extracted give useful information for distinguishing malicious PEs from legitimate ones in the absence of running programs. The distribution for features and classes showed differences in number of sections, entry point, image size, code size, DLL characteristics, section characteristics, subsystem information, and import features and attributes. The feature analysis also suggested that the structural attributes of PE are not equally informative for the classification, which indicates the necessity to find informative characteristics before model training. This allows for a small and relevant representation of samples to be executed for later classification [39].

The comparative results of the experiments showed the benefit of using complementary machine-learning techniques. SVM attained an accuracy of 96.2%, whereas the accuracy of the Random Forest and XGBoost were 97.4% and 98.0%, respectively. The proposed hybrid configuration also achieved the highest accuracy of 98.8%, while the rest of the combinations (RF+SVM and SVM+XGBoost) achieved 98.2% and 98.4%, respectively. These gains were also seen with the other classification metrics with the hybrid method having around 98.6% precision, 98.9% recall, and 98.7% F1 score. The ROC analysis also showed excellent discriminatory ability; the hybrid model was able to achieve an AUC of about 0.995. From the results, it was concluded that the combination of multiple learning mechanisms can give a more reliable classification boundary than using one classifier.

A confusion-matrix analysis also proved the classification process effective. The evaluated hybrid model was able to correctly classify 965 benign samples and 998 malware samples, with 25 benign samples misclassified as malware and 12 malware samples misclassified as benign. Malware detection was especially concerned with the comparatively low rate of false-negative predictions because false negatives mean a direct threat to security when it comes to malware. Meanwhile, the fact that there were only a few false-positives suggests that the method be able to achieve a manageable tradeoff between malware detection and legitimate-file detection. The accuracy, precision, recall, F1-score, ROC-AUC, and confusion matrix analysis give a high score that reflects good classification performance in the evaluated dataset.

One key aspect of the study was the adaptation of the detection mechanism to execute changes in the characteristics of the executable during successive stages of update. The adaptive analysis revealed that the performance of the static model progressively decreased from 98.5% at the first step to 91.8% at the fifth step. The adaptive hybrid model kept performing well and improved from 98.5% in the beginning to 98.8% in Update 5. As the performance gap between the two approaches increases, it becomes important to include new characteristics that are observed and can be executed in the detection process. The adaptive mechanism therefore offers a way to limit degradation resulting from the evolution of the underlying mal-distribution and also allows for more stable detection performance over time [40].

In general, results demonstrate that statically analyzing the PE header and hybrid machine learning and adaptive updating was a promising approach for automated malware detection. The strategy combines the best features of both worlds: the computational convenience of static analysis of the executable program with the pattern-recognition power of machine learning models. The experimental results obtained for the feature analysis, the classifiers comparison, the confusion-matrix results, the ROC-AUC evaluation and the adaptive-stage experiments all confirm the effectiveness of the proposed framework in the experimental setting. The study also provides a basis for future work for improvement by using larger and more diverse executable datasets, the addition of more PE and section-level characteristics, more advanced drift-detection mechanisms, incremental or online learning, and the test against new

families of malware that have not been seen before. These extensions can help improve the strength, scalability, and generalization of the detection framework in an ever-changing malware landscape.

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. Alsumaidaee, Y. A. M., Yahya, M. M., & Yaseen, A. H. (2025). Optimizing Malware Detection and Classification in Real-Time Using Hybrid Deep Learning Approaches. *International Journal of Safety & Security Engineering*, *15*(1), 141.
3. Singh, P., Borgohain, S. K., & Kumar, J. (2025). Optimized feature representation and selection for malware detection using portable executable headers and machine learning: P. Singh et al. *Journal of Computer Virology and Hacking Techniques*, *21*(1), 10.
4. El-Hajj, M. I. (2025, November). Hybrid Malware Classification using Static and Dynamic Features with Machine Learning. In *2025 12th International Conference on Wireless Networks and Mobile Communications (WINCOM)* (pp. 1-8). IEEE.
5. Bulut, I., & Rúa, E. A. (2024, September). Short Paper: Machine Learning-Based Secure Malware Detection Using Features from Binary Executable Headers. In *European Symposium on Research in Computer Security* (pp. 193-203). Cham: Springer Nature Switzerland.
6. Soliman, K., Sobh, M., & Bahaa-Eldin, A. M. (2024). Robust malicious executable detection using host-based machine learning classifier. *Computers, Materials & Continua*, *79*(1), 1419-1439.
7. Al-Khshali, H. H., & Ilyas, M. (2022). Impact of portable executable header features on malware detection accuracy. *Computers, Materials and Continua*, *74*(1), 153-178.
8. El Mouhtadi, W., Maleh, Y., & Mounir, S. (2024, October). Machine learning techniques for enhanced malware detection in portable executable files. In *The Proceedings of the International Conference on Smart City Applications* (pp. 784-796). Cham: Springer Nature Switzerland.
9. Syeda, D. Z., & Asghar, M. N. (2024). Dynamic malware classification and api categorisation of windows portable executable files using machine learning. *Applied Sciences*, *14*(3), 1015.
10. Daniel, A. B., & Patel, N. D. (2025, August). Detection of Malware on Portable Executable Files Using Machine Learning. In *2025 International Conference on Networks & Advances in Computational Technologies (NetACT)* (pp. 1-6). IEEE.
11. Hussain, S. S., Ab Razak, M. F., & Firdaus, A. (2024). Deep learning-based hybrid analysis of malware detection and classification: A recent review. *Journal of Cyber Security and Mobility*, *13*(1), 91-134.
12. Kale, G., Bostancı, G. E., & Çelebi, F. V. (2024). Evolutionary feature selection for machine learning based malware classification. *Engineering Science and Technology, an International Journal*, *56*, 101762.
13. 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.
14. Shailaja, P., Jahan, T., Sharmila, K., Varma, P. B. S., Arra, S., & Kumar, P. M. (2025). MalwareNet: an intelligent malware detection and classification using advanced extreme leaning machine in edge computing environment. *Egyptian Informatics Journal*, *31*, 100714.
15. Sathish, A., Babu, N., Bhavani, M., Divya, K., Jhoshna, B., & Hemavathi, T. (2024, August). Adaptive Malware Detection in Windows Portable Executables (Wpes). In *2024 4th Asian Conference on Innovation in Technology (ASIANCON)* (pp. 1-5). IEEE.
16. Borate, V., Adsul, A., Gaikwad, A., Mhetre, A., & Dicholkar, S. (2025). A Novel Technique for Malware Detection Analysis Using Hybrid Machine Learning Model. *International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)*, *5*(5), 472-484.
17. Dabas, N., Ahlawat, P., & Sharma, P. (2023). An effective malware detection method using hybrid feature selection and machine learning algorithms. *Arabian Journal for Science and Engineering*, *48*(8), 9749-9767.
18. Subrahmanyam, S. S. B., Goutham, P., Ambati, V. K. R., Bijitha, C. V., & Nath, H. V. (2023). A hybrid method for analysis and detection of malicious executables in IoT network. *Computers & Security*, *132*, 103339.
19. Tumkur, S. D., Eswarakrishnan, V., Wairagade, A., Aslam, M. A., Bilal, M., & Cheema, A. M. (2025). Optimizing malware detection in virtual cloud environments using hybrid machine learning approach. *Arabian Journal for Science and Engineering*, *50*(23), 19463-19494.
20. Mohammed, M. A., Lakhan, A., Zebari, D. A., Abdulkareem, K. H., Nedoma, J., Martinek, R., ... & Tiwari, P. (2023). Adaptive secure malware efficient machine learning algorithm for healthcare data. *CAAI Transactions on Intelligence Technology*.
21. Ferdous, J., & Islam, R. (2024). A survey on machine learning techniques in multi-platform malware detection: Securing PC, mobile devices, IoT, and cloud environments.
22. Mohammad, F., Al-Ahmadi, S., & Al-Muhtadi, J. (2025). OMD-RAS: Optimizing Malware Detection through Comprehensive Approach to Real-Time and Adaptive Security. *Computers, Materials & Continua*, *84*(3), 5995-6014.

23. 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*.
24. 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.
25. Al Haque, A. F. (2023). Optimizing the Performance of Machine Learning Algorithms in Detecting Malicious Files using Hybrid Models.
26. 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.
27. Singh, M. P., & Karkhur, Y. (2023, March). Portable Executable Header Based Ransomware Detection using Power Iteration and Artificial Neural Network. In *2023 6th International Conference on Information Systems and Computer Networks (ISCON)* (pp. 1-6). IEEE.
28. Wang, Y., Wang, Q., Qin, X., Chen, X., Xin, B., & Yang, R. (2023). DockerWatch: a two-phase hybrid detection of malware using various static features in container cloud. *Soft Computing*, *27*(2), 1015-1031.
29. Al-Andoli, M. N., Sim, K. S., Tan, S. C., Goh, P. Y., & Lim, C. P. (2023). An ensemble-based parallel deep learning classifier with PSO-BP optimization for malware detection. *IEEE Access*, *11*, 76330-76346.
30. Chanajitt, R. (2023). *Machine learning approaches for malware classification based on hybrid artefacts* (Doctoral dissertation, The University of Waikato).
31. Khalid, C., & El Hakouni, R. (2024). Advancing malware classification with hybrid deep learning. *The Art of Cyber Defense: From Risk Assessment to Threat Intelligence*, 25.
32. Gonzalez-Gorrín, V. M., Prieto-Blázquez, J., & Arnedo-Moreno, J. (2025, April). A hybrid machine learning method for detecting metamorphic malware. In *International Conference on Advanced Information Networking and Applications* (pp. 297-307). Cham: Springer Nature Switzerland.
33. Maniriho, P., Mahmood, A. N., & Chowdhury, M. J. M. (2024). A survey of recent advances in deep learning models for detecting malware in desktop and mobile platforms. *ACM Computing Surveys*, *56*(6), 1-41.
34. Panda, B., Bisoyi, S. S., Panigrahy, S., & Mohanty, P. (2025). Machine learning techniques for imbalanced multiclass malware classification through adaptive feature selection. *PeerJ Computer Science*, *11*, e2752.
35. Rani, S., Tripathi, K., & Kumar, A. (2023). Machine learning aided malware detection for secure and smart manufacturing: a comprehensive analysis of the state of the art. *International Journal on Interactive Design and Manufacturing (IJIDeM)*, 1-28.
36. Abid, Y. A., Wu, J., Farhan, M., & Ahmad, T. (2023). ECMT framework for internet of things: an integrative approach employing In-Memory attribute examination and sophisticated neural network architectures in conjunction with hybridized machine learning methodologies. *IEEE Internet of Things Journal*, *11*(4), 5867-5886.
37. Faiz, A., Fuzail, M., Aftab, A., Aslam, N., & Akbar, A. J. (2024). A multimodal machine learning approach for android malware detection: Static code analysis. *International Journal of Information Systems and Computer Technologies*, *4*(1), 20-35.
38. Almazroi, A. A., & Ayub, N. (2024). Deep learning hybridization for improved malware detection in smart Internet of Things. *Scientific reports*, *14*(1), 7838.
39. Tayyab, U. E. H., Khan, F. B., Durad, M. H., Khan, A., & Lee, Y. S. (2022). A survey of the recent trends in deep learning-based malware detection. *Journal of Cybersecurity and Privacy*, *2*(4), 800-829.
40. Ali, E., Batool, N., Rizwan, M., & Sarwar, S. (2024, August). Assessing Concept Drift in Malware: A Comprehensive Review and Analysis. In *2024 21st International Bhurban Conference on Applied Sciences and Technology (IBCAST)* (pp. 564-569). IEEE.