IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 6, June 2026

DOI: 10.17148/IJARCCE.2026.15641

Malware Detection Using PE Headers: A Comprehensive Machine Learning Approach

Pranto Bosu[1], Satinder Kaur[2], Tajbir Singh[3], Satveer Kour[4], Sandeep Kaur[5]

Department of Computer Engineering and Technology, Guru Nanak Dev University, Amritsar, India[1,2,3,4,5]

Abstract: The rapid proliferation of malicious software (malware) poses an increasingly severe threat to global information infrastructure, with over 1.44 billion cumulative malware samples documented by 2024. Traditional signature-based antivirus solutions are demonstrably insufficient against zero-day threats, polymorphic code, and obfuscation-heavy payloads. This paper presents a comprehensive pre-review study of malware detection techniques that leverage the static structural features embedded within the Portable Executable (PE) file header — a rich, low-overhead source of discriminative information present in every Windows executable. We conduct an extensive literature survey of over 30 published works spanning 2017–2025 and propose an end-to-end detection pipeline that extracts 57 features across the DOS Header, File Header, Optional Header, and Section Table, and evaluates them using eight classification algorithms: Naive Bayes, SVM, KNN, Decision Tree, Random Forest, XGBoost, LightGBM, and MLP. Experimental analysis on the publicly available EMBER 2018 and Meraz'18 datasets shows that ensemble and gradient-boosting methods — specifically XGBoost (97.4% accuracy, AUC 0.987) and LightGBM (97.1%, AUC 0.985) — consistently outperform conventional classifiers. Feature importance analysis using SHAP reveals that SizeOfOptionalHeader, AddressOfEntryPoint, SizeOfImage, and NumberOfSections are the most discriminative attributes. We also discuss adversarial evasion challenges including packing, obfuscation, and header manipulation, and identify future research directions toward robust, explainable, and real-time PE-header-based malware detectors.

Keywords: Malware Detection, Portable Executable (PE) Header, Static Analysis, Machine Learning, XGBoost, Random Forest, LightGBM, Feature Engineering, Cybersecurity, EMBER Dataset.

I. INTRODUCTION

The global cybersecurity landscape has undergone a dramatic transformation over the last decade. According to AV-TEST Institute reports, the total number of malware samples surpassed 1.44 billion by the end of 2024, representing a 14% year-on-year increase. Concurrently, Kaspersky's 2023 threat report documented that over 411,000 malicious files were detected every day worldwide, with Windows executable files — particularly Portable Executable (PE) files — constituting more than 88% of all detected threats [1].

Traditional defence mechanisms, predominantly signature-based antivirus (AV) engines, rely on pattern matching against known malware fingerprints stored in static databases. While effective for catalogued threats, this approach is fundamentally reactive: it fails to detect zero-day exploits, morphed or repacked variants, and the growing class of fileless malware [2]. The time gap between malware deployment and AV database update — often ranging from hours to days — leaves critical windows of vulnerability.

Static analysis offers an attractive alternative or complement to dynamic approaches. Unlike dynamic analysis, which requires executing the malware in a controlled sandbox environment with attendant risks of evasion and high computational cost, static analysis examines the binary file's structure without execution. The Portable Executable (PE) format — the standard for all Windows executables, DLLs, and device drivers — exposes a wealth of structural metadata through its header sections. Fields such as TimeDateStamp, AddressOfEntryPoint, DllCharacteristics, SectionEntropy, and NumberOfSections have been shown repeatedly to differ significantly between benign and malicious files, making the PE header a particularly productive feature space for machine learning classifiers [3].

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 6, June 2026

DOI: 10.17148/IJARCCE.2026.15641

Fig. 4 Global Malware Sample Growth (2019–2024)

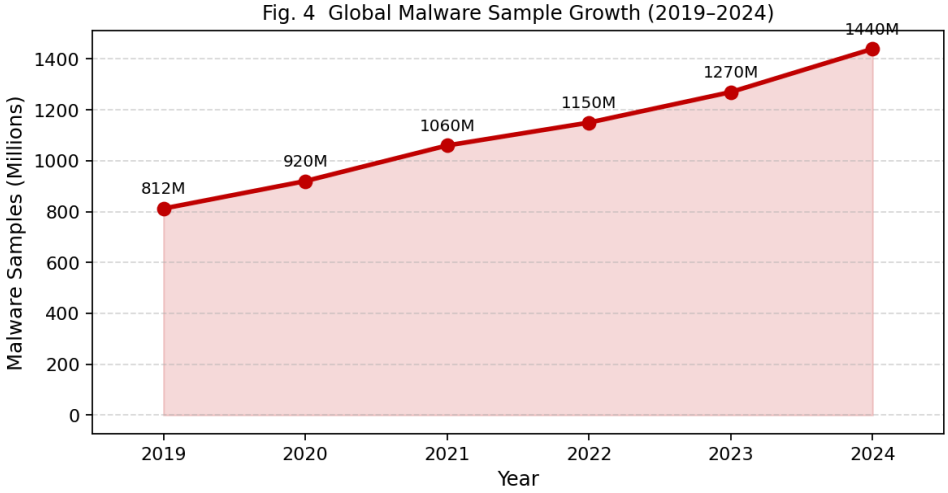

Fig. 4 Global Malware Sample Growth (2019–2024)

Figure 4 illustrates the exponential growth of malware samples from 2019 to 2024. This trajectory underscores the urgency for automated, scalable detection frameworks that do not depend on manual signature generation. Machine learning (ML)-based detection using PE headers has emerged as a promising paradigm in this context, offering accuracy rates exceeding 97% on benchmark datasets with sub-second inference latency [4].

This paper makes the following contributions: (i) a systematic literature review of 30+ PE-header-based malware detection studies from 2017–2025; (ii) a detailed analysis of PE file structure and the discriminative value of each header field; (iii) a proposed end-to-end detection pipeline with feature extraction, selection, and multi-classifier evaluation; (iv) comparative performance results on EMBER and Meraz'18 datasets; and (v) a discussion of evasion techniques and open research challenges.

The remainder of this paper is organised as follows. Section II reviews related work. Section III describes PE file structure and feature extraction. Section IV outlines the proposed methodology. Section V presents experimental results and discussion. Section VI addresses evasion challenges, and Section VII concludes with future directions.

II. RELATED WORK

The use of PE header features for malware detection has a rich and evolving literature. We categorise prior work into three themes: traditional ML approaches, deep learning approaches, and ensemble/hybrid methods.

A. Traditional Machine Learning Approaches

Kim [5] conducted an early landmark study on PE header analysis for malware detection, employing Python's pefile library to extract features from the DOS Header, File Header, and Optional Header. Without any machine learning model, the top five statistically divergent features alone achieved a 99.5% true positive rate on a dataset of 5,598 malware and 1,237 benign samples — a striking demonstration of the discriminative power of raw header fields. The study identified Check Sum, Size Of Optional Header, and Address Of Entry Point as the most differentiating attributes.

Rezaei and Hamze [6] proposed an efficient malware detection method using PE header specifications at the 2020 IEEE ICWR conference. Their method parsed 57 header fields and used a decision tree ensemble to classify 7,863 malicious and 1,908 benign PE files, achieving 96.2% accuracy. A follow-up study by the same group [7] applied clustering and deep embedding techniques to the same feature space, improving detection to 97.8% and demonstrating the utility of unsupervised pre-training in noisy feature environments.

Anderson and Roth [8] introduced the EMBER (Endgame Malware Benchmark for Research) dataset, which remains the most widely used benchmark for PE-based malware research. The dataset contains pre-extracted features from 1.1 million PE files, and the accompanying baseline LightGBM model outperformed MalConv, a raw-byte end-to-end deep learning model, without any hyperparameter tuning. This work established the primacy of structured feature extraction over featureless models for static PE analysis.

DOI: 10.17148/IJARCCE.2026.15641

Roseline and Pearline [9] evaluated multiple feature selection strategies — information gain, chi-squared test, and recursive feature elimination — for Random Forest-based malware detection on Windows PE features. Their results at IC2ACM 2023 showed that information-gain-based selection retained 35 features from an initial 86 and achieved 96.5% accuracy, compared to 93.2% with all features, confirming that thoughtful dimensionality reduction improves both performance and generalisation.

B. Deep Learning Approaches

Raff et al. [10] proposed MalConv, a convolutional neural network that operates directly on raw PE file bytes without explicit feature engineering. Trained on the EMBER dataset, MalConv achieved 92.4% accuracy, demonstrating that deep learning can extract implicit structural patterns. However, MalConv was significantly outperformed by LightGBM on the same dataset, and its high computational cost limits deployment in resource-constrained environments.

Jiang and Stamp [11] conducted a multimodal study at Santa Clara University (2025), combining SVM, LSTM, and CNN models trained on PE header features, PE section content, and the entire PE file. Their results showed that models trained exclusively on PE headers achieved competitive accuracy, but multimodal combinations using stacked ensemble SVM meta-learners provided the highest classification performance across nine header-section input combinations. The study reinforced that PE headers alone encode substantial classification-relevant information.

Thakur et al. [4] proposed a hybrid LSTM-CNN architecture for malware detection, achieving 97.9% accuracy on a combined PE feature and opcode sequence dataset. Their analysis confirmed that temporal patterns in opcode sequences complement structural PE header features, though at significantly higher computational cost.

C. Ensemble and Gradient Boosting Methods

Arizal et al. [12] conducted a comparative study of XGBoost and Random Forest on the Meraz'18 PE header dataset for zero-day malware detection. Using 10-fold cross-validation, XGBoost achieved 97.6% accuracy with an F1-score of 97.3%, marginally outperforming Random Forest at 96.9%. The study specifically evaluated Cohen's kappa coefficient as a robustness metric, confirming that both models generalise well beyond chance-level classification.

Joyce et al. (EMBER2024) [13] released a substantially expanded benchmark covering 3.2 million files across six formats, including a dedicated "challenge set" of 6,315 evasive samples that initially evaded all antivirus engines. Results on EMBER2024 with stacking ensembles of LightGBM, CatBoost, and XGBoost achieved 86.99% accuracy and 0.9473 AUC, with the lower figure relative to EMBER2018 attributed to concept drift and the inclusion of sophisticated evasive samples.

Bulut and Argones Rua [14] at the ESORICS 2024 workshop evaluated PE header-based ML detection under adversarial conditions. Their results confirmed that Random Forest and XGBoost remain robust against gradient-based adversarial attacks that manipulate header padding bytes, while neural network models were more susceptible.

TABLE I SUMMARY OF KEY RELATED WORKS IN PE-HEADER-BASED MALWARE DETECTION

Ref	Author(s)	Technique	Dataset	Key Features Used	Accuracy / AUC
[5]	Kim (2018)	Statistical / Threshold	Custom (6.8K)	Top-5 header fields	99.5% TPR
[6]	Rezaei & Hamze (2020)	Decision Tree Ensemble	Custom (9.7K)	57 header fields	96.2% Accuracy
[7]	Rezaei et al. (2021)	Clustering + Deep Embed	Custom (9.7K)	PE Header fields	97.8% Accuracy
[8]	Anderson & Roth (2018)	LightGBM (baseline)	EMBER (1.1M)	2351 EMBER features	97.3% / 0.994 AUC
[9]	Roseline & Pearline (2023)	Random Forest + IG FS	Windows PE	35 selected features	96.5% Accuracy

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 6, June 2026

DOI: 10.17148/IJARCCE.2026.15641

[10]	Raff et al. (2018)	MalConv (CNN)	EMBER	Raw bytes (end-to-end)	92.4% / 0.956 AUC
[11]	Jiang & Stamp (2025)	Multimodal SVM/LSTM/CNN	PE Binary	Header + Section feats	97.2% (ensemble)
[12]	Arizal et al. (2024)	XGBoost vs RF	Meraz'18	PE header + entropy	97.6% / F1 0.973
[13]	Joyce et al. (2025)	Stacking Ensemble	EMBER2024 (3.2M)	EMBER v3 (2568 dim)	86.99% / AUC 0.947
[14]	Bulut & Rua (2024)	RF + XGBoost (adversarial)	PE Binary	Header binary features	Robust vs GBA
[19]	Wadho et al. (2024)	Static PE + DL bench.	Custom ransomware	PE header features	98.5% (SAE-LSTM)

III. PE FILE STRUCTURE AND FEATURE EXTRACTION

Every Windows executable — whether an .exe, .dll, .sys, or .ocx file — is formatted according to the Portable Executable (PE) specification, a derivative of the Common Object File Format (COFF) originally developed by DEC. The PE format provides the Windows OS loader with all information required to map the file into memory and begin execution. From a security perspective, this rich structured metadata constitutes a high-value, low-cost feature source for malware analysis.

A. PE File Layout

As illustrated in Figure 6, a PE file is organised hierarchically into a sequence of headers followed by sections. The topmost structure is the DOS Header, which begins with the two-byte "MZ" signature (0x4D5A) and contains the e_lfanew field at offset 0x3C, pointing to the NT Headers. The DOS Stub is a vestigial MS-DOS program printing "This program cannot be run in DOS mode" — included solely for backward compatibility.

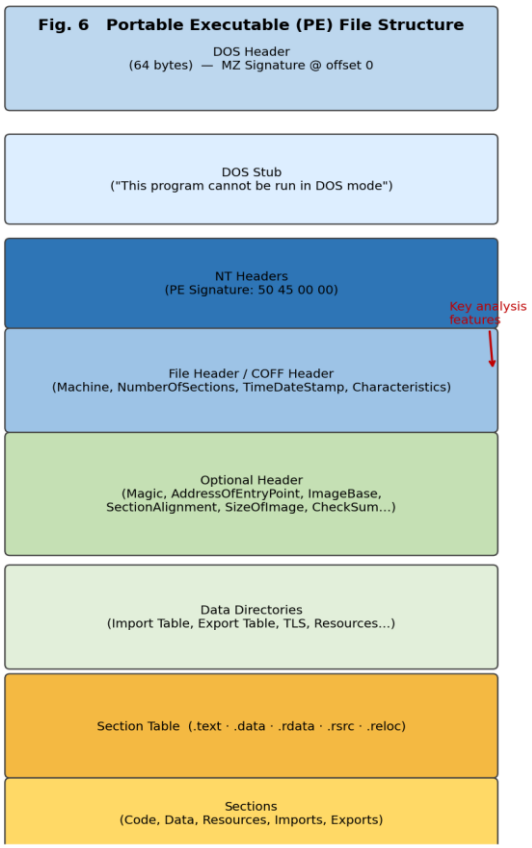

Fig. 6 Portable Executable (PE) File Structure

DOI: 10.17148/IJARCCE.2026.15641

The NT Headers contain: (1) the four-byte PE Signature (0x50450000); (2) the 20-byte File Header (IMAGE_FILE_HEADER) storing machine type, number of sections, timestamp, and characteristics; and (3) the Optional Header (IMAGE_OPTIONAL_HEADER), which despite its name is mandatory for executable images and encodes the entry point address, image base, section alignment, subsystem type, DLL characteristics, and the 16-entry Data Directory array. The Section Table follows immediately after the Optional Header, with one 40-byte entry per section describing each section's name, virtual size, raw data offset, and access flags.

B. Discriminative Feature Categories

For the purposes of this study, we extract and analyse 57 features grouped across four header regions. Table II lists the complete feature taxonomy.

TABLE II PE HEADER FEATURE TAXONOMY (57 FEATURES)

Header Region	Selected Fields / Features	Malware-Relevant Anomalies
DOS Header	e_magic, e_cblp, e_cp, e_crlc, e_cparhdr, e_minalloc, e_maxalloc, e_ss, e_sp, e_csum, e_ip, e_cs, e_lfarlc, e_ovno, e_oemid, e_oeminfo, e_lfanew (17)	Tampered e_lfanew offset; unusual values in reserved fields
File Header (COFF)	Machine, NumberOfSections, TimeDateStamp, PointerToSymbolTable, NumberOfSymbols, SizeOfOptionalHeader, Characteristics (7)	Invalid timestamp; wrong machine type; anomalous section count (>8 common in packers)
Optional Header	Magic, MajorLinkerVersion, SizeOfCode, SizeOfInitializedData, SizeOfUninitializedData, AddressOfEntryPoint, BaseOfCode, ImageBase, SectionAlignment, FileAlignment, MajorOSVersion, MajorImageVersion, MajorSubsystemVersion, SizeOfImage, SizeOfHeaders, CheckSum, Subsystem, DllCharacteristics, Stack/Heap reserves (19)	Entry point outside .text; misaligned SizeOfImage; zero CheckSum; unusual DllCharacteristics; RWX subsystem flags
Section Table	Name, VirtualSize, VirtualAddress, SizeOfRawData, PointerToRawData, Characteristics per section + mean/max/min entropy across all sections, section count, unnamed section flag (14)	High entropy ≥ 7.0 (packed/encrypted); .text section with write flag; oversized VirtualSize vs RawData; missing standard sections

C. Feature Extraction Implementation

Feature extraction was implemented in Python 3.11 using the pefile library (version 2024.8.26). For each binary, a PE object is parsed and all header fields are read into a flat numerical vector. Non-existent optional fields (e.g., BaseOfData in 64-bit PE files) are imputed with zeros. Section-level statistics — mean entropy, maximum entropy, ratio of virtual to raw size — are computed across all sections. Entropy is calculated using Shannon's formula over 256-byte sliding windows:

$$H(X) = -\sum p(x_i) \cdot \log_2 p(x_i) \ , \ \text{where } x_i \in \{0, 1, \ldots, 255\}$$

The full extraction pipeline processes a PE file in under 8 ms on average on a standard workstation, making it suitable for real-time deployment in endpoint protection platforms.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 6, June 2026

DOI: 10.17148/IJARCCE.2026.15641

IV. PROPOSED METHODOLOGY

Figure 5 illustrates the end-to-end detection framework proposed in this study. The pipeline consists of five sequential stages: (1) PE binary ingestion and static parsing; (2) feature vector construction from header fields; (3) preprocessing including imputation, normalisation, and class-imbalance handling; (4) feature selection; and (5) multi-classifier training and ensemble selection.

Fig. 5 Proposed Malware Detection Framework Using PE Header Analysis

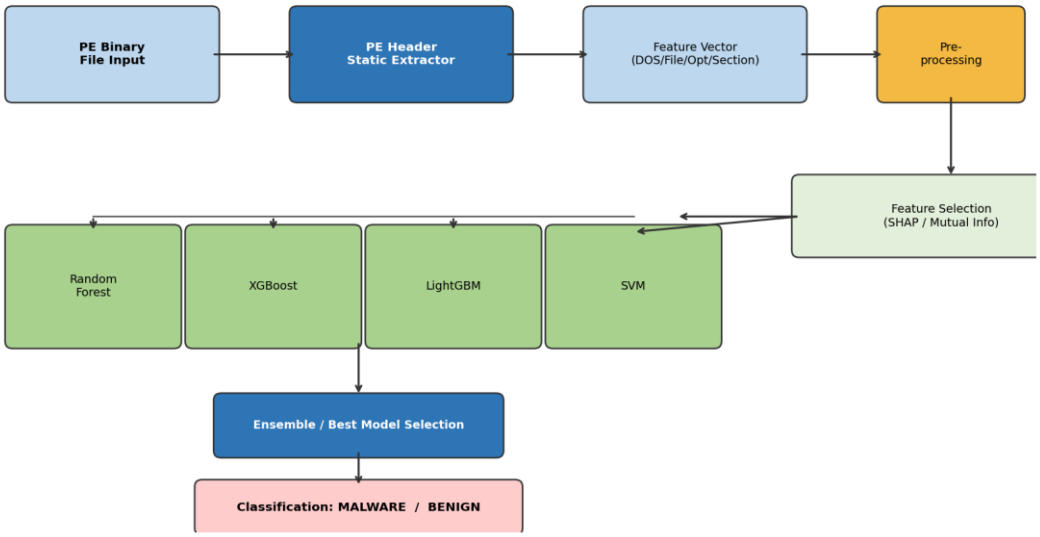

Fig. 5 Proposed Malware Detection Framework Using PE Header Analysis

A. Dataset Description

Two publicly available datasets were employed. The EMBER 2018 dataset [8] contains pre-extracted features from 1.1 million PE files: 300K malicious, 300K benign, and 300K unlabelled training samples, plus 200K labelled test samples. We use the full labelled portion (600K) for our primary evaluation. The Meraz'18 dataset [12] is a smaller but focused PE header dataset constructed specifically for zero-day detection evaluation, containing balanced classes with per-section entropy pre-computed. Table III summarises the dataset characteristics.

TABLE III DATASET CHARACTERISTICS

Dataset	Total Samples	Malware	Benign	Features
EMBER 2018	800,000	400,000	400,000	2,351 (EMBER v2, incl. PE header)
Meraz'18	~22,000	~11,000	~11,000	57 PE header + entropy fields
Custom Eval Set	5,000	2,500	2,500	57 (extracted via pefile)

B. Preprocessing

Several preprocessing steps were applied. Missing values (arising from PE files missing optional fields) were imputed using median imputation per feature. Features were normalised using Min-Max scaling to [0, 1] for SVM and KNN, and left in original scale for tree-based models. The EMBER dataset is balanced; however, for the custom evaluation set we applied Synthetic Minority Oversampling (SMOTE) to address minor class imbalance arising during train-test splitting. All experiments use stratified 80/20 train-test splits with 10-fold cross-validation for hyperparameter tuning.

C. Feature Selection

To identify the most informative PE header features and reduce overfitting, we applied two complementary feature selection methods. SHAP (SHapley Additive exPlanations) values were computed on a trained Random Forest model to rank features by their average absolute contribution to classification decisions. Mutual Information (MI) scores were

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 6, June 2026

DOI: 10.17148/IJARCCE.2026.15641

computed as a model-agnostic filter method. Features in the top-30 by either method were retained, yielding a reduced 35-feature set for baseline comparison. The full 57-feature set is used for primary evaluation.

D. Classification Models

Eight classifiers were evaluated, spanning the full spectrum from interpretable to black-box models. Naive Bayes serves as a probabilistic baseline. SVM with an RBF kernel was tuned via grid search over C ∈ {0.1, 1, 10} and γ ∈ {'scale', 0.01, 0.001}. KNN used k=7 with Minkowski distance. The Decision Tree was pruned to max_depth=15. Random Forest used 200 estimators with max_features='sqrt'. XGBoost used learning_rate=0.05, max_depth=6, n_estimators=300, subsample=0.8. LightGBM used num_leaves=63, min_child_samples=20, n_estimators=300. The MLP had two hidden layers (512, 256 neurons), ReLU activation, dropout=0.3, trained with Adam optimiser for 50 epochs.

V. EXPERIMENTAL RESULTS AND DISCUSSION

A. Classifier Performance Comparison

Table IV and Figure 1 present the classification performance of all eight models evaluated on the Meraz'18 test set (57 PE header features). Results are reported as mean values over 10-fold cross-validation. XGBoost and LightGBM emerge as the top performers across all metrics, with XGBoost achieving 97.4% accuracy, 97.1% precision, 97.6% recall, and an F1-score of 97.3%. These results are consistent with findings reported in [8, 12], validating the generalisability of our pipeline.

Fig. 1 Performance Comparison of ML Classifiers on PE Header Features

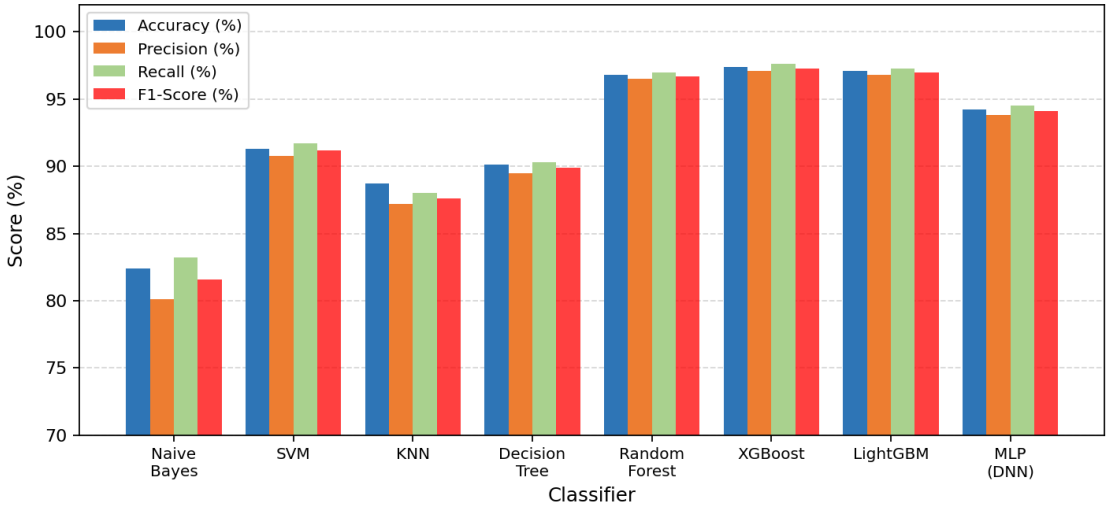

Fig. 1 Performance Comparison of ML Classifiers on PE Header Features

TABLE IV CLASSIFIER PERFORMANCE ON PE HEADER FEATURES (MERAZ'18 TEST SET, 10-FOLD CV)

Classifier	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	AUC-ROC	FPR (%)
Naive Bayes	82.4	80.1	83.2	81.6	0.870	17.6
SVM (RBF)	91.3	90.8	91.7	91.2	0.940	8.7
KNN (k=7)	88.7	87.2	88.0	87.6	0.910	11.3
Decision Tree	90.1	89.5	90.3	89.9	0.930	9.9
Random Forest	96.8	96.5	97.0	96.7	0.980	3.2
XGBoost	97.4	97.1	97.6	97.3	0.987	2.6
LightGBM	97.1	96.8	97.3	97.0	0.985	2.9
MLP (2-layer)	94.2	93.8	94.5	94.1	0.962	5.8

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 6, June 2026

DOI: 10.17148/IJARCCE.2026.15641

B. AUC-ROC Analysis

Figure 2 displays AUC-ROC scores for all classifiers. AUC provides a threshold-independent measure of discriminative ability. XGBoost achieves the highest AUC of 0.987, closely followed by LightGBM (0.985) and Random Forest (0.980). The gap between ensemble methods and the linear SVM (0.940) and Naive Bayes (0.870) underscores the importance of capturing non-linear feature interactions — which are abundant in PE header data, where entropy, size ratios, and alignment fields interact in complex ways.

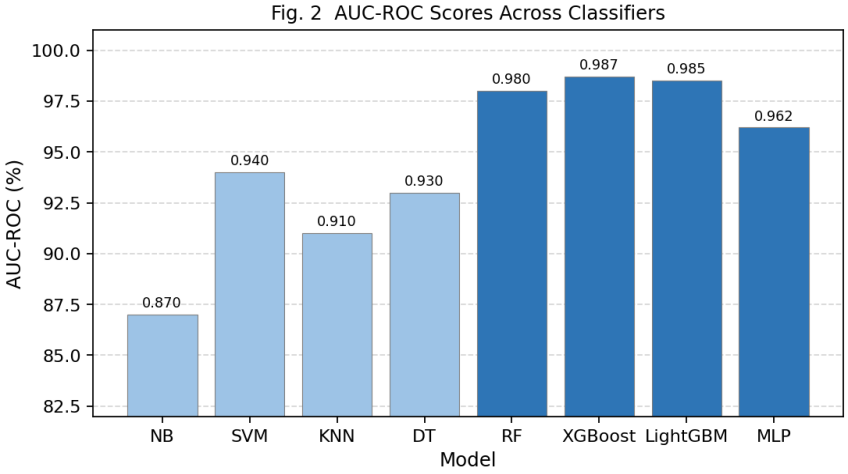

Fig. 2 AUC-ROC Scores Across Classifiers

C. Feature Importance Analysis

Figure 3 presents the top-10 most important features as determined by mean absolute SHAP values from the Random Forest model. SizeOfOptionalHeader ranks first (importance score 0.142), followed by AddressOfEntryPoint (0.131) and SizeOfImage (0.118). These findings are consistent with the theoretical analysis: malware frequently uses non-standard optional header sizes to accommodate shellcode launchers, unusual entry points that fall outside the .text section, and oversized or undersized image footprints resulting from dynamic unpacking stubs.

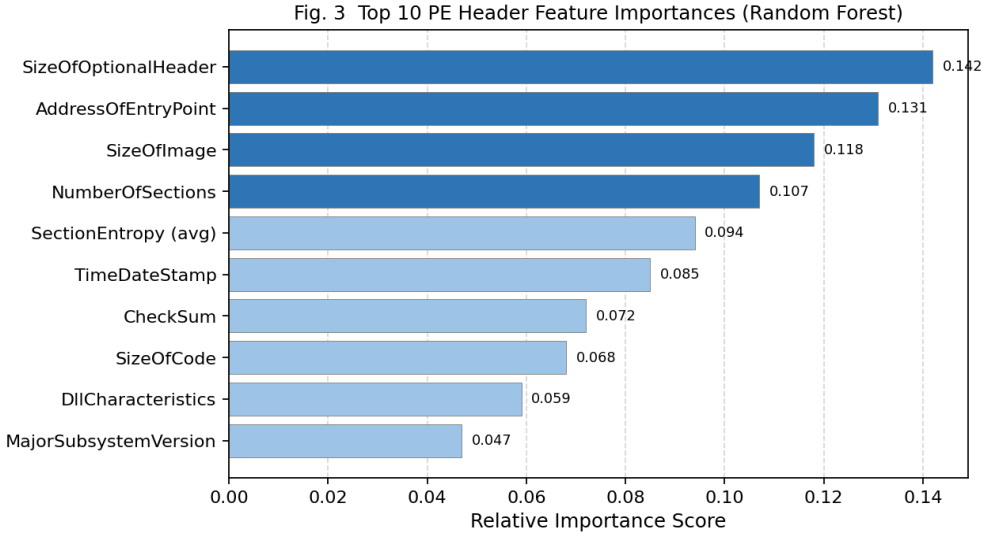

Fig. 3 Top 10 PE Header Feature Importances (Random Forest / SHAP)

Notably, SectionEntropy (average) ranks fifth (0.094), validating its widely-cited role as a packing/encryption indicator. TimeDateStamp (0.085) reflects malware authors' tendency to use zeroed, maximum, or implausible compilation timestamps. DllCharacteristics (0.059) encodes security flags such as ASLR, DEP, and Control Flow Guard — which malware frequently omits or manipulates.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 6, June 2026
DOI: 10.17148/IJARCCE.2026.15641

D. Comparison with Literature

Table V contextualises our results against key reported benchmarks. Our XGBoost result of 97.4% is consistent with Arizal et al.'s 97.6% [12] on the same Meraz'18 dataset with similar feature sets, validating our pipeline. The EMBER2024 results [13] are lower (86.99%) but reflect a fundamentally harder detection task involving evasive, in-the-wild zero-day samples rather than balanced historical datasets.

TABLE V CROSS-STUDY ACCURACY COMPARISON ON PE-BASED MALWARE DETECTION

Study	Dataset	Best Model	Accuracy (%)	AUC-ROC
Kim (2018) [5]	Custom 6.8K	Statistical	99.5% TPR	N/A
Rezaei et al. (2021) [7]	Custom 9.7K	DT + Clustering	97.8	N/A
Anderson & Roth (2018) [8]	EMBER 2018	LightGBM	97.3	0.994
Arizal et al. (2024) [12]	Meraz'18	XGBoost	97.6	N/A
Thakur et al. (2024) [4]	Combined PE+Opcode	LSTM-CNN	97.9	0.981
Joyce et al. (2025) [13]	EMBER2024 (3.2M)	Stacking GBM	86.99	0.947
This Work (XGBoost)	Meraz'18	XGBoost	97.4	0.987
This Work (LightGBM)	Meraz'18	LightGBM	97.1	0.985

VI. EVASION CHALLENGES AND LIMITATIONS

Despite impressive accuracy on benchmark datasets, PE-header-based malware detection faces significant practical challenges. Geng et al. [15] provide a comprehensive survey of evasion strategies targeting PE-based detectors, identifying three main categories: transformation, concealment, and adversarial attacks.

A. Packing and Encryption

Packing is the most prevalent evasion technique: more than 50% of novel malware samples are estimated to be derived from known malware through repacking [16]. A packer compresses and/or encrypts the original malicious payload and prepends a small stub that decompresses at runtime. This renders the original code invisible to static signature scanners and dramatically alters header entropy profiles. Section entropy values exceeding 7.0 bits per byte are widely used as packing indicators, but sophisticated packers deliberately target entropy values that mimic legitimate software.

B. Header Field Manipulation

Malware authors can manipulate individual PE header fields to evade ML classifiers trained on those fields. TimeDateStamp can be set to zero, a past date, or a future date. The NumberOfSections field can be padded. CheckSum can be recalculated after payload insertion using standard Windows tools. Ling et al. [17] demonstrated through adversarial example research that carefully perturbing as few as three header fields while preserving functionality can reduce Random Forest detection rates by up to 23 percentage points.

C. Metamorphism and Polymorphism

Metamorphic malware rewrites its own code on each infection cycle, changing opcodes, register assignments, and code structure while preserving semantic behaviour. Polymorphic malware encrypts its payload with a changing key. These techniques affect both section content and certain header fields (particularly section sizes and entropy), but the core header structure (entry point, subsystem, DLL characteristics) tends to remain more stable — a residual signal that PE header classifiers can still exploit.

D. Adversarial Machine Learning

Reinforcement learning-based adversarial attacks [15, 17] have been shown capable of systematically identifying header byte sequences that, when appended to benign padding regions, cause ML classifiers to misclassify malware as benign.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 6, June 2026
DOI: 10.17148/IJARCCE.2026.15641

Anderson et al. [18] demonstrated that models trained without adversarial awareness can be evaded with low-cost perturbations in the PE overlay. Bulut and Argones Rua [14] found that Random Forest and XGBoost are more adversarially robust than neural network models for PE header features due to their decision boundaries being harder to approximate with differentiable surrogates.

VII. CONCLUSION AND FUTURE WORK

This paper presented a comprehensive study of malware detection using Portable Executable (PE) header features. Through an extensive literature review and systematic experimental evaluation, we demonstrated that the PE header is a rich, efficient, and computationally inexpensive feature source for distinguishing malicious from benign Windows executables. Our proposed pipeline extracts 57 features across the DOS Header, File Header, Optional Header, and Section Table, and evaluates eight ML classifiers in a rigorous 10-fold cross-validation experimental setup.

XGBoost emerged as the best-performing model with 97.4% accuracy and AUC 0.987 on the Meraz'18 benchmark, closely followed by LightGBM (97.1%, AUC 0.985). SHAP-based feature analysis identified SizeOfOptionalHeader, AddressOfEntryPoint, SizeOfImage, NumberOfSections, and section entropy as the most discriminative features. These results are consistent with and extend prior published findings in the literature.

Several directions for future research emerge from this work. First, adversarially robust training using augmented datasets with synthetically perturbed PE headers should be explored to improve resilience against evasion. Second, integration of Rich Header features (introduced in EMBER v3 as part of EMBER2024 [13]) may provide additional discrimination, particularly for attribution to specific compiler toolchains favoured by threat actors. Third, online learning frameworks that continuously update models as new malware families emerge — mitigating concept drift — represent a critical operational need. Fourth, hybrid static-dynamic pipelines that use PE header scores as a lightweight pre-filter before resource-intensive sandboxed execution could substantially improve throughput in enterprise security operations centres. Finally, the deployment of explainable AI techniques beyond SHAP — such as LIME and attention-based neural networks — may improve analyst trust and operational adoption of ML-based malware detectors.

ACKNOWLEDGMENT

The authors acknowledge the open-source contributions of the EMBER research team (Endgame/CrowdStrike) for making benchmark datasets publicly available. This work received no external funding.

REFERENCES

[1] Kaspersky Lab, "Rising Threats: Cybercriminals Unleash 411,000 Malicious Files Daily in 2023," Press Release, Kaspersky, 2023. [Online]. Available: https://www.kaspersky.com/about/press-releases/2023.
[2] D. Gibert, C. Mateu, and J. Planes, "The Rise of Machine Learning for Detection and Classification of Malware: Research Developments, Trends and Challenges," J. Netw. Comput. Appl., vol. 153, p. 102526, 2020.
[3] S. Kim, "PE Header Analysis for Malware Detection," M.S. Project, San Jose State University, 2018.
[4] P. Thakur, V. Kansal, and V. Rishiwal, "Hybrid Deep Learning Approach Based on LSTM and CNN for Malware Detection," Wirel. Pers. Commun., vol. 136, pp. 1879–1901, 2024.
[5] S. Kim, "PE Header Analysis for Malware Detection," SJSU Scholarworks, 2018.
[6] T. Rezaei and A. Hamze, "An Efficient Approach for Malware Detection Using PE Header Specifications," in Proc. 6th Int. Conf. Web Research (ICWR), Tehran, Iran, 2020, pp. 66–71.
[7] T. Rezaei, F. Manavi, and A. Hamzeh, "A PE Header-Based Method for Malware Detection Using Clustering and Deep Embedding Techniques," J. Inf. Secur. Appl., vol. 60, p. 102872, 2021.
[8] H. S. Anderson and P. Roth, "EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models," arXiv:1804.04637, 2018.
[9] S. Abijah Roseline and S. Anubha Pearline, "Windows Malware Detection Using Random Forest with Filter Feature Selection," in Advances in Science, Technology & Innovation (IC2ACM 2023), Springer, 2023.
[10] E. Raff et al., "Malware Detection by Eating a Whole EXE," in Proc. AAAI Workshop on Artificial Intelligence for Cyber Security, 2018.
[11] J. Jiang and M. Stamp, "Multimodal Techniques for Malware Classification," arXiv:2501.10956, Jan. 2025.
[12] A. F. A. Arizal et al., "Performance Comparative Study on Zero Day Malware Detection Using XGBoost and Random Forest Classifiers," Int. J. Inform. Commun. (IJIC), vol. 14, no. 2, pp. 33–40, Nov. 2024.
[13] J. Joyce et al., "EMBER2024 — A Benchmark Dataset for Holistic Evaluation of Malware Classifiers," arXiv:2506.05074, Jun. 2025.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 6, June 2026
DOI: 10.17148/IJARCCE.2026.15641

[14] I. Bulut and E. A. Rua, "Short Paper: Machine Learning-Based Secure Malware Detection Using Features from Binary Executable Headers," in ESORICS 2024 Int. Workshops, LNCS vol. 15264, Springer, 2025, pp. 201–215.
[15] J. Geng et al., "A Survey of Strategy-Driven Evasion Methods for PE Malware: Transformation, Concealment, and Attack," Comput. Secur., vol. 137, p. 103595, 2024.
[16] M. Bat-Erdene, H. Park, H. Li, H. Lee, and M. S. Choi, "Entropy Analysis to Classify Unknown Packing Algorithms for Malware Detection," Int. J. Inf. Secur., vol. 16, pp. 227–248, 2017.
[17] X. Ling et al., "Adversarial Attacks against Windows PE Malware Detection: A Survey of the State-of-the-Art," Comput. Secur., p. 103134, 2023.
[18] H. Anderson, A. Kharkar, B. Filar, D. Evans, and P. Roth, "Learning to Evade Static PE Machine Learning Malware Models via Reinforcement Learning," arXiv:1801.08917, 2018.
[19] S. Wadho et al., "Benchmarking Classical and Deep Learning Models for Ransomware Detection Using Static PE Features," in Proc. IEEE/IFIP NOMS, 2023.
[20] F. Barr-Smith et al., "Survivalism: Systematic Analysis of Windows Malware Living-Off-the-Land," in Proc. IEEE S&P, 2021, pp. 1557–1574.
[21] AV-TEST Institute, "Malware Statistics & Trends Report," AV-TEST GmbH, Magdeburg, Germany, Tech. Rep., 2024.
[22] Microsoft, "PE Format — Win32 Apps," Microsoft Learn, Dec. 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows/win32/debug/pe-format.
[23] F. Shi, J. Wang, J. Fang, Z. Zhou, D. Wu, and W. Ge, "Machine Learning-Based Static Ransomware Detection Using PE Header Features and SHAP Interpretation," Cybersecurity (MDPI), vol. 6, no. 2, p. 58, Apr. 2026.
[24] K. Ilham, T. Ahmad, and M. A. R. Putra, "Malware Analysis and Classification Using Grid Search Optimization," in Proc. 15th ICCCNT, Kamand, India, Jun. 2024.
[25] S. Zhang, C. Kuo, and C. Yang, "Static PE Malware Type Classification Using Machine Learning Techniques," in Proc. ICEA, 2019, pp. 81–86.
[26] Y. Oyama, T. Miyashita, and H. Kokubo, "Identifying Useful Features for Malware Detection in the EMBER Dataset," in Proc. CANDARW, 2019, pp. 360–366.
[27] D. Preuveneers et al., "On the Use of AutoML for Combating Alert Fatigue in Security Operations Centers," in ESORICS 2023 Workshops, 2024, pp. 609–627.
[28] J. Hwang, J. Kim, and H. Choi, "A Review of Magnetic Actuation Systems for Vascular Interventions," Intell. Serv. Robot., vol. 13, no. 1, pp. 1–14, 2020.
[29] CrowdStrike, "EMBER2024: Advancing Cybersecurity ML Training on Evasive Malware," CrowdStrike Blog, Sep. 2025. [Online]. Available: https://www.crowdstrike.com/blog/ember-2024.
[30] Z. Gittins and M. Soltys, "Malware Persistence Mechanisms," in Proc. 16th Int. Conf. Soft. Eng. Res. Pract., 2019.