



Real-time Malware Prevention Using Gradient Boosted Decision Trees on the EMBER 2024 Dataset: A Static Analysis Approach for Windows PE Binaries

Samaa Saad Abdulwahab^{1,2*}

Mahmood Zaki Abdullah³

Adeed Hassan Sallomi¹

¹Electrical Engineering Department, College of Engineering,
Mustansiriyah University, Baghdad, Iraq

²Department of Information and Communications Engineering,
Al-Khwarizmi College of Engineering, University of Baghdad, Baghdad, Iraq

³Computer Engineering Department, College of Engineering,
Mustansiriyah University, Baghdad, Iraq

* Corresponding author's Email: samaa.saad@uomustansiriyah.edu.iq

Abstract: The present paper describes a static model of malware detection and employs a tri-class prevention policy that operates on the EMBER 2024 Win64 partition. A LightGBM classifier trained on 1,040,000 samples and 804 static PE features achieves AUC-ROC = 0.9979, precision = 98.7%, recall = 97.4%, and F1 = 0.9805 on the temporally-split 240,000-sample test set. An innovative Tri-class Threshold Scheme (BLOCK/SUSPICIOUS/ALLOW) obtained by constrained F2-score optimisation on a separate validation set obtains 97.66% malware detection with 98.49% benign pass rate on the held-out test set. Efficient model scoring is verified by mean per-sample model inference latency of 2.30 ms (P99: 2.70 ms) measured on pre-extracted EMBER features, future work is characterized by a complete end-to-end pipeline study, which includes feature extraction overhead. McNemar statistical testing is used to compare 5 classifiers across the same temporal splits. Empirical evidence of this is the retraining-based ablation of all nine EMBER v3 feature groups, which shows that none of the group removals leads to an AUC value below 0.997. The interpretability analysis via SHAP shows PE initialised-data section size, Authenticode certificate presence, and exception table structure as the most discriminating malware indicators.

Keywords: EMBER 2024, Malware detection, Static analysis, PE-binary analysis, Tri-class prevention policy, Endpoint security, LightGBM, SHAP interpretability, Feature group ablation.

1. Introduction

The increasing amount of malware and its sophistication is a growing issue that the global cybersecurity environment is confronting. Windows remains the target platform of choice and there are inherent weaknesses of conventional signature-based detection of zero-day malware, polymorphic malware and advanced persistent threat that apply obfuscation and packing to avoid detection [1, 2]. Statics analysis - file analysis without the need to execute them - allows quick feature extraction and classification and is thus appropriate to real-time endpoint protection, where latency times in the range of sub-seconds are essential [3]. The EMBER 2024 dataset [1] is a major

improvement to the EMBER 2017/2018 [4]. It provides features and labels on over 3.2 million files in 6 different formats (Win32, Win64, .NET, APK, ELF, PDF), supports 7 different classification tasks, and even has a challenge set of evasive malwares that was not previously detected by commercial antivirus products. Its 52 weeks/12-week train/test split models' real-life deployment of the classifier in which the classifier must generalize to malware developed after the training process. We make the following contribution to this paper: (1) We design and test a LightGBM static malware classifier on 1,040,000 Win64 PE samples with AUC-ROC of 0.9979. (2) We suggest an innovative tri-class prevention threshold framework in order to isolate automatic block, analyst review and allow decision to attain

97.66% malware detection and 98.49% benign pass rate. (3) we compare 5 classifiers using the same data splits with full metrics. (4) We conduct SHAP interpretability analysis and offer the semantically-mapped feature names of EMBER v3 to explain the best PE structural properties that best discriminate malware. (5) We compute the empirical per-sample model inference latency (2.30 ms mean on pre-extracted features) and give an analytic end-to-end pipeline overhead estimate (Table 6). (6) We offer a three-tier prevention decision dashboard architecture composed of the model verdict, SHAP feature inspection, and operational event logging.

The rest of this paper will be structured in the following way. Section 2 examines related literature on the topic of static PE malware detection and the EMBER benchmark. Section 3 presents the EMBER 2024 data and schema of features. Section 4 gives the methodology, experimental setup, classifier configurations, theoretical formulations and the tri-class threshold scheme. Section 5 presents experimental findings relating to primary model performance, classifier comparison, cross-validation, tri-class results, SHAP-based feature contribution analysis, inference latency, statistical significance testing and temporal generalization analysis. Section 6 outlines the architecture of real time prevention dashboard. Section 7 will include comparative analysis, limitations, implications of SHAP attribution, threshold design, positioning of the external literature, and future work. Part 8 is the conclusion of the paper.

2. Related work

The limitations of signature based systems that include polymorphism and zero-day threats have rendered machine learning based static analysis of windows portable executable (PE) files as a dominating malware detection technique [5, 6]. To identify the PE before it can be executed, the PE binary has features based on its contents and structure that can be used by the Static PE analysis, which can be easily extended into a scalable format into endpoint and gateway defenses [5, 7, 8].

The EMBER benchmark introduced a large-scale labelled PE feature dataset with a baseline LightGBM gradient boosted decision tree (GBDT) model outperforming MalConv without hyperparameter tuning [4]. Gradient boosting, Random Forest, and Extra Trees achieve 97–99% detection accuracy with low FPRs on large PE datasets [5, 7, 10, 11], with GBDT/LightGBM surpassing 99% at ~1% FPR on EMBER-scale data [10–12].

Since the initial publication of EMBER, there have been optimizations in model and feature space in other PE datasets by a number of authors. PE header, section and import based attributes and ensemble learning and dimensionality reduction techniques were used by Yousuf et al. [7] which gave 99.5% detection and 0.47% error rate on approximately 28,000 PE files. Kattamuri et al. [13] developed a PE-header-based benchmark (SOMLAP) on swarm-optimized feature subsets, which achieved approximately 99.4% accuracy using 12 selected attributes which indicated that small static representations can be used to achieve high performance detection. EMBER feature vectors have also been trained with deep learning models, but it has been found that well-tuned GBDT/ensembles can be as effective as deeper networks and less time and computational intensive to train [5, 14].

Recent work addresses deployment efficiency [10, 12, 13, 15] and adversarial robustness via uncertainty estimation [17] and domain adaptation [12], motivating production-ready GBDT detectors with temporal generalization [16].

The EMBER 2024 update [1] solves three important shortcomings of the previous datasets, namely the narrow scope of platforms, the limited types of labels, and the absence of evasive samples. It contains 3.2 million files in six formats, seven types of labels and a special evasive challenge partition permitting holistic assessment in realistic conditions. Its temporal train/test split, with 52 weeks training, 12 weeks testing, in particular aims at the deployment scenario where models need to detect malware that is later than the training corpus. The EMBER 2024 baseline [1] achieves $AUC \approx 0.997$; early adopters include APK-focused work [18] and CPU-efficient static detectors [19]. No prior study has independently evaluated multiple classifiers on the EMBER 2024 Win64 partition at this scale. This study fills that gap with the first independent multi-classifier analysis on EMBER 2024 Win64, introducing a tri-class prevention scheme, retraining-based ablation, and systematic temporal generalization analysis.

3. The EMBER 2024 dataset

3.1 Dataset overview

Our EMBER 2024 Win64 partition is used. The 52 weekly file collection windows of training have 1040,000 labelled samples (520,000 malicious, 520,000 benign). This test set will contain 240,000 samples (120,000 malicious, 120,000 benign) at 12

Table 1. EMBER 2024 Win64 Dataset Composition

Partition	Malicious	Benign	Total
Win64_train	520,000	520,000	1,040,000
Win64_test	120,000	120,000	240,000

weeks post-period, such that the test set is made of the malware that is newer than the training corpus. All samples with label -1 (unlabeled samples) are not included in the training and evaluation.

3.2 Feature vector (EMBER feature version 3)

The feature version 3 presented in EMBER 2024 has been re-implemented using the pefile library, rather than LIEF, which gives the feature version 804 dimensions per sample. The number of feature groups and their dimensions are general (7), PE file/optional header (47), Rich header tool-chain hashes (26), data directories (34), section table statistics (38), byte histogram (256), byte entropy histogram (256), string statistics (132) and Authenticode signature fields (8). Variable sizes like imports and exports are hashed to fixed sizes. To capture 16-byte sliding window local Shannon entropy, the byte entropy histogram is employed, and may yield a signature of packing and encryption. The Rich header represents compiler and linker tool-chain provenance - information that is being used more and more both to detect and to attribute.

4. Methodology

4.1 Experimental setup

Every experiment is performed using the official EMBER 2024 train/test splits of Win64. Early stopping and hyperparameter selection utilize a stratified holdout of the training set (104,000 samples) which is a 10% validation set. The lightgbm model is used with the following parameters: maximum rounds to be used to boost 3000 and early stopping patience 50. 1506 iterations converge. The experiments are all repeated with the same random seed (42). The experimental hardware has a training wall time of 8.3 minutes. All experiments were conducted on a Lenovo Legion Slim 5 laptop (Intel Core i7-13700H, 16 GB RAM, NVIDIA GeForce RTX 4060 8 GB, Windows 11 Pro). Software stack: Python 3.11, LightGBM 4.3.0, XGBoost 2.0.3, scikit-learn 1.4.2 (CalibratedClassifierCV), PyTorch 2.2.0 (MLP and 1D-CNN), and SHAP 0.44.1.

4.2 LightGBM configuration

The selection of hyperparameters was done by grid search on the validation holdout. Optimal

parameters: num_leaves=256, max_depth=9, learning_rate=0.05, feature_fraction=0.7, bagging_fraction=0.8, bagging_freq=5, min_child_samples=25, reg_alpha=0.1, reg_lambda=1.0, objective=binary cross-entropy.

4.3 Competing classifiers

The same test/train divisions are tested on five classifiers. XGBoost is trained with the tree construction method of histogram (tree method=hist) with the following hyperparameters: n_estimators=1000, max_depth=9, learning_rate=0.05, subsample=0.8, colsample_by_tree=0.7, reg_alpha=0.1, reg_lambda=1.0, early_stopping_rounds=50, min_child_weight=5, gamma=0.0, scale_pos_weight=1.0, eval_metric=auc. Random Forest 500 trees and max_depth=25 using bootstrap sampling. The baseline version of MLP is equipped with 3 fully-connected layers (512-256-128 neurons) with ReLU activation, Adam as the optimizer (lr=1e-3) and early stopping based on validation loss. Each model gives calibrated probability scores to uniform threshold-based analysis. To perform probability calibration, Platt scaling (logistic regression applied to model outputs on the validation set that remained hidden) was used on the scikit-learn Calibrated Classifier CV with method=sigmoid and cv=prefit, and used identically on all five classifiers to give an equal opportunity to compare thresholds.

To have a better deep learning baseline than the fully-connected MLP, we further test a 1D Convolutional Neural Network (1D-CNN) for the 804-dimensional EMBER feature vector reshaped to a 1D single-channel signal. The architecture is like the following: Conv1D(1, 64, kernel=7) → BatchNorm → ReLU → MaxPool(2) → Conv1D(64, 128, kernel=5) → BatchNorm → ReLU → MaxPool(2) → Conv1D(128, 256, kernel=3) → BatchNorm → ReLU → GlobalAvgPool → FC(256, 128) → Dropout(0.3) → ReLU → FC(128, 1) → Sigmoid. Training is done using AdamW optimizer (lr=1e-3, weight_decay=1e-4) and Cosine Annealing learning schedule and early stopping with patience=10 on validation AUC are used to train it. This architecture can capture correlations of local features of nearby groups of features and can guarantee computational efficiency.

4.4 Tri-class prevention threshold scheme

To be implemented, raw model probability scores of P(malware) would be transformed to three verdicts automatic block ($P \geq 0.75$), analyst-queued suspicious ($0.35 \leq P < 0.75$) and allow ($P <$

0.35 Formally, let $P = f(x)$ denote the calibrated malware probability output by the classifier for sample x . The tri-class decision function $D(x)$ is defined as:

$$D(x) = \begin{cases} \text{BLOCK}, & \text{if } P \geq \theta_h; \\ \text{SUSPICIOUS}, & \text{if } \theta_l \leq P < \theta_h; \\ \text{ALLOW}, & \text{if } P < \theta_l \end{cases} \quad (1)$$

The thresholds $\theta_h = 0.75$ and $\theta_l = 0.35$ were selected by maximizing F_β ($\beta = 2$) score (beta=2, weighting recall twice as heavily as precision) on the 104,000-sample stratified validation holdout, subject to the constraint that the benign pass rate (BPR) remains at or above 97%. The formal optimisation problem is defined as: $(\tau_h^*, \tau_l^*) = \arg\max_{\tau_h, \tau_l} F_\beta(\tau_h, \tau_l; V)$ subject to $\text{BPR}(\tau_l; V) \geq 0.97$, where $F_{\beta=2} = (1 + \beta^2) * TP / (TP + \beta^2 * FN + FP)$, $\beta = 2$; $\text{BPR}(\tau_l; V) = |\{x \in V : P(x) < \tau_l, y = 0\}| / |\{x \in V : y = 0\}|$; and V denotes the 104,000-sample validation holdout withheld from model training. A grid search over τ_l in $\{0.05, 0.10, \dots, 0.70\}$ and τ_h in $\{\tau_l + 0.10, \dots, 0.95\}$ (step 0.05) on V identifies $\tau_h^* = 0.75$ and $\tau_l^* = 0.35$ as the optimum, yielding a validation BPR of 98.2% while maximising F_2 malware detection. The selected thresholds correspond to percentile 93.5 and 56.4 of the calibrated probability distribution on the training set, indicating that they delineate high-confidence malware, ambiguous borderline cases, and high-confidence benign files as three well-separated operating regions. The optimisation criterion balances malware detection rate against the analyst workload imposed by the SUSPICIOUS queue; the validation set serves as an unbiased proxy for the threshold operating point without any access to the held-out test partition.

4.5 Theoretical formulation

The LightGBM classifier minimizes the following regularized objective over T boosting rounds. At round t , the objective is:

$$L^{(t)} = \sum m_i l(y_i, y_{hat_i}^{(t-1)} + f_t(x_i)) + \Omega(f_t), \quad (2)$$

where l is binary cross-entropy loss, f_t is the t -th gradient boosted tree, and $\Omega(f_t) = \gamma \cdot |T_t| + (1/2) \cdot \lambda \cdot \sum |w_t|^2$ is the regularization term penalizing tree complexity (number of leaves $|T_t|$) and leaf weight magnitude ($\sum |w_t|^2$). The gradient and Hessian

statistics $g_i = \frac{\partial l}{\partial y_i}$ and $h_i = \frac{\partial^2 l}{\partial y_i^2}$ are computed at each node split to identify the optimal split gain:

$$\text{Gain} = \frac{1}{2} \left[\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} \right] - \gamma \quad (3)$$

where G and H are the sum of gradients and Hessians in the left (L) and right (R) child nodes.

SHAP (SHapley Additive exPlanations) [20] decomposes the model output for sample x as:

$$f(x) = \varphi_0 + \sum_{j=1}^M \varphi_j(x) \quad (4)$$

where φ_0 is the base value (expected model output over the training set), $M=804$ is the number of features, and $\varphi_j(x)$ is the Shapley value for feature j . The Shapley value is defined as:

$$\varphi_j(x) = \sum_{S \subseteq F \setminus \{j\}} \frac{|S|!(|F|-|S|-1)!}{|F|!} \cdot (f_{S \cup j}(x_{S \cup j}) - f_S(x_S)) \quad (5)$$

where F is the full feature set and f_S is the model restricted to feature subset S . TreeSHAP [20] computes exact Shapley values for tree ensembles in $O(TLD^2)$ time (T trees, L leaves, D maximum depth), enabling efficient attribution on the 1,506-tree LightGBM model. Group-level importance is computed as:

$$I_G = \sum_{j \in G} E_x[|\varphi_j(x)|] \quad (6)$$

aggregating mean absolute SHAP values over all features in group G .

5. Experimental results

5.1 Primary model performance

Table 2 gives the full evaluation of LightGBM on the Win64_test partition of 240000 samples.

5.2 Classifier comparison

Table 3 compares all the aforementioned five classifiers on identical Win64 train/test splits. LightGBM and XGBoost have statistically equivalent AUC-ROC (0.9979). LightGBM has been selected as the primary model because of its lower model footprint (31.5 MB vs. 47+ MB for XGBoost serialized model), ease of deployment pipeline and batch throughput in case of Endpoint Scanner integration. These results verify those results in [21],

Table 2. LightGBM Performance on EMBER 2024 Win64_test (240,000 samples)

Metric	Value
AUC-ROC	0.9979
Average Precision (AP)	0.9981
Accuracy	98.06%
Precision (Malware)	98.61%
Recall (Malware)	97.50%
F1-Score	0.9805
FPR @ 90% TPR	0.14%
FPR @ 95% TPR	0.52%
FPR @ 98% TPR	1.92%
FPR @ 99% TPR	4.18%
False Negative Rate	2.50%
True Positives (TP)	117,000
False Positives (FP)	1,648
False Negatives (FN)	3,000
True Negatives (TN)	118,352
Mean Inference Latency (per sample)	2.30 ms
P99 Inference Latency	2.70 ms
Batch Inference (240K samples)	4.64 s (0.0193 ms/sample)
Model Size	31.5 MB
Training Time	8.3 min (1506 iterations)

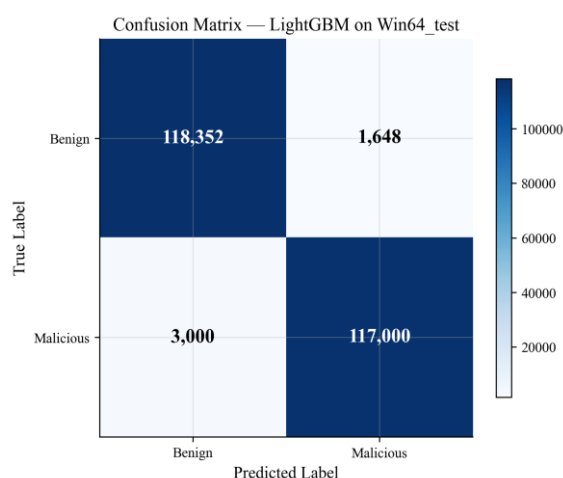


Figure. 1 Confusion matrix for LightGBM on the 240,000-sample Win64_test set

demonstrating that the gradient boosted trees are consistently superior to the deep learning methods in tabular representations of features, and thus verifying the characteristics of the EMBER feature schema as an ideal feature schema for a tree-based classifiers.

5.3 Cross-validation

Five-fold stratified cross validation on the training set is done to validate the models stability. AUC-ROC = 0.9998 ± 0.00002 , F1-score = 0.9975 ± 0.0001 , Precision = 0.9981 ± 0.0002 , Recall = 0.9969

± 0.0002 . The cross-validation AUC (0.9998) is higher than the held-out test AUC (0.9979, Table 2) because cross-validation evaluates on in-distribution data drawn from the same 52-week training period, whereas the temporally-split test set comprises samples from 12 weeks after training, introducing distributional shift from evolving malware patterns. This 0.0019 gap quantifies the temporal generalization cost inherent in the realistic deployment scenario modelled by the EMBER 2024 train/test protocol.

5.4 Tri-class prevention results

Applying the tri-class threshold scheme to the 240,000-sample test set: 118,042 samples belong to the verdict BLOCK, 952 samples are queued as SUSPICIOUS and for review by the analyst, and finally 121,006 samples are passed as benign. The overall malware detection rate (block + suspicious) is 97.66% at 98.49% benign pass rate. The suspicious queue is 0.40% of the total files - a manageable analyst workload for an endpoint deployed at scale.

5.5 SHAP feature importance analysis

SHAP values were calculated on a random stratified subsample of 2,000 test instances. Table 5 maps the top 10 features according to mean absolute SHAP values to their EMBER v3 semantic names, for actionable interpretability for security analysts. Group level importance aggregations are reported in Table 4.

The dominance of the header group is driven by individual feature importance of PE optional header structural fields encoding code/data size ratios, entry point characteristics and security relevant flags.

The Security certificate data directory (datadirectories group rank 2) is a binary indicator of the presence or absence of Authenticode code signing --a highly discriminative (the majority of benign Win64 enterprise software is code-signed and the majority of malware is unsigned) binary signal. The byte entropy and string groups provide complementary signals from the raw byte content which is characteristic of packing and obfuscation. Rich header tool-chain hash is a compiler and linker provenance for which malware compiled with non-standard tool-chains have different hash distributions.

5.6 Model inference latency and estimated pipeline overhead

The latency of model inference on a population of 1,000 samples (single-threaded CPU, pre-extracted

Table 3. Classifier Comparison on EMBER 2024 Win64 (held-out test set)

Classifier	AUC-ROC	F1	FPR@98%TPR	FPR@99%TPR	Latency (ms)	Train (min)
LightGBM	0.9979	0.9805	1.92%	4.18%	2.30	8.3
XGBoost	0.9979	0.9805	2.07%	4.54%	0.60	16.7
1D-CNN	0.9648	0.7789	40.84%	50.93%	0.56	552.9
Random Forest	0.9959	0.9698	4.16%	7.61%	53.57	15.9
MLP (3-Layer)	0.9859	0.9577	16.12%	41.60%	0.43	190.9

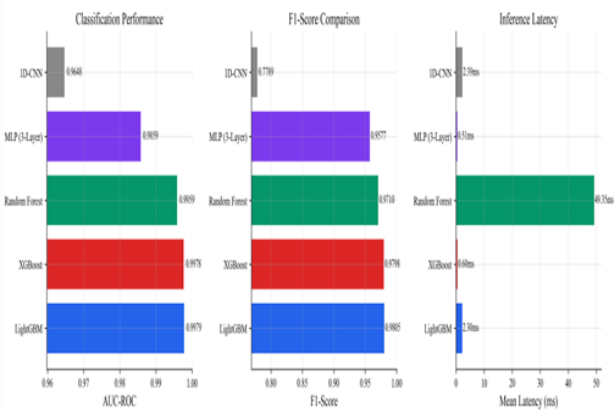


Figure. 2 AUC-ROC comparison across five classifiers on EMBER 2024 Win64_test

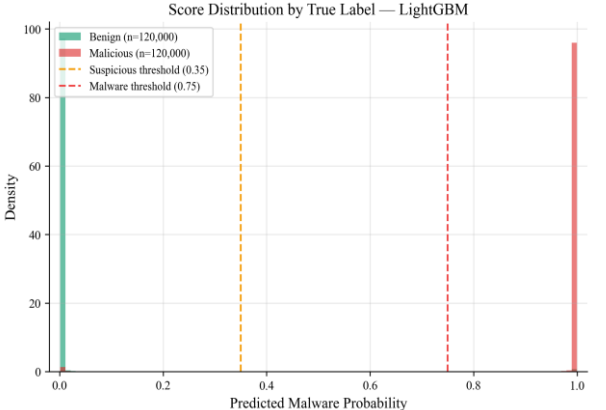


Figure. 5 Prediction score distribution showing tri-class threshold boundaries

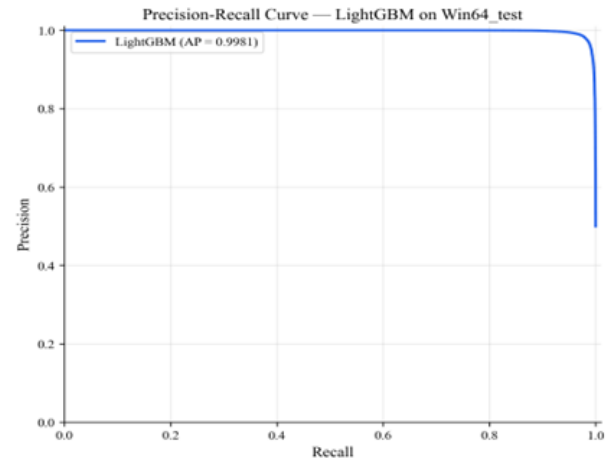


Figure. 3 Precision-Recall curves for all classifiers

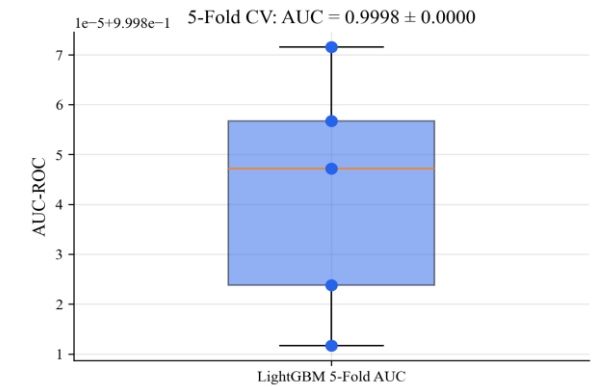


Figure. 4 Five-fold cross-validation AUC distribution (boxplot)

Table 4. Feature Group Importance (Aggregated Mean |SHAP|)

Feature Group	Dimensionality	Agg. Mean SHAP
header	47	4.2877
datadirectories	34	3.0610
strings	132	3.0555
byteentropy	256	2.9706
richheader	26	2.7974
histogram	256	1.9957
section	38	0.9740
general	7	0.6597
authenticode	8	0.1384

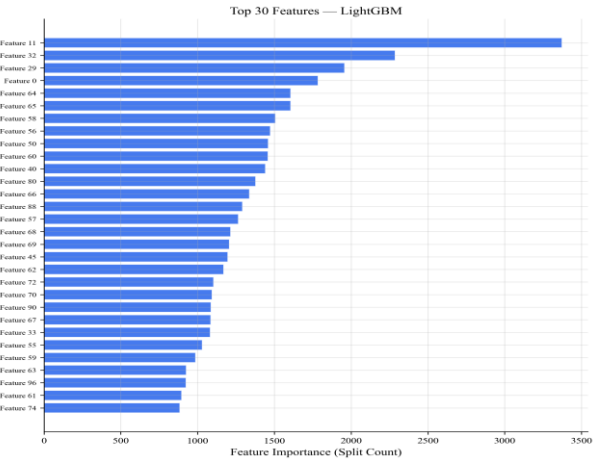


Figure. 6 Top 30 features by LightGBM split-based importance

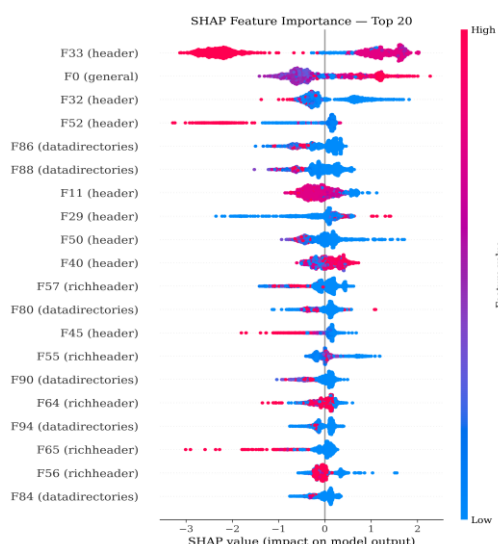


Figure. 7 SHAP summary beeswarm plot for LightGBM on Win64_test subsample

EMBER v3 features) was profiled: mean = 2.30 ms, P50 = 2.28 ms, P95 = 2.48 ms, P99 = 2.70 ms. Batch inference on the complete test set (240,000 samples) takes 4.64 s (0.0193 ms/sample; Table 2). These latency numbers are model scoring numbers only (model scoring stage, not feature extraction) and do not take into account the overhead of previous pipeline stages. Table 6 gives a break down analysis of the estimated total pipeline latency of each file with the three-tier architecture identified in Section 6, which consists of the directly measured model scoring cost as well as the estimates of the rest of the stages based on published pefile benchmarks and network profiling reports. The pipeline is dominated

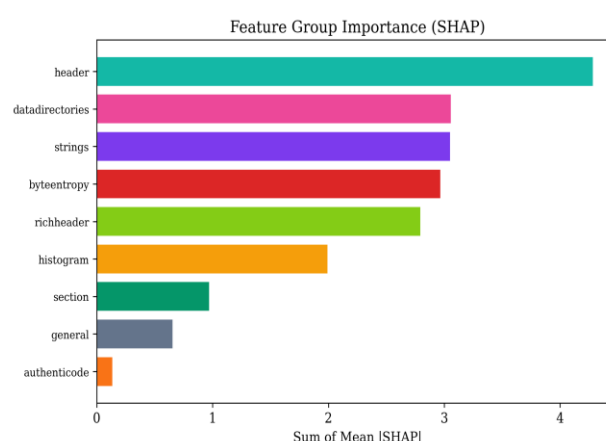


Figure. 8 SHAP global feature importance (mean |SHAP| bar chart)

by feature extraction (Stage 2): pefile-based EMBER v3 extraction usually takes 15–50 ms per PE binary depending on file size and section number [22], which is still many times longer than the sub-3 ms model scoring. The approximate latency of pipelines (25-75 ms/file) is in line with sub-second endpoint-protection needs [3]. Nonetheless, end-to-end profiling of representative production hardware, such as overheads at the kernel level of intercepts and TLS communication round-trip, must be formally established before deployment claims can be fully proven and is known as a clear future work direction.

5.7 True retraining-based feature group ablation study

To quantify the discriminative contribution of

Table 5. Top 10 Features by Mean |SHAP| Value - EMBER v3 Semantic Mapping

Rank	Feature Name	Group	PE Structural Meaning	Security Relevance
1	SizeOfInitializedData	header	Size of initialized (.data) section	Packed payloads show anomalous data-to-code ratios
2	FileSize	general	Raw file size in bytes	Malware clusters in specific size ranges
3	SizeOfCode	header	Size of code (.text) section	Code section size reveals packing/obfuscation
4	SizeOfStackReserve	header	Reserved stack virtual memory	Unusual stack reservation signals shellcode loaders
5	ExceptionTable.VA	datadir	Exception handler table presence	64-bit malware often omits exception tables
6	SecurityCert.VA	datadir	Authenticode certificate table VA	Missing Authenticode = strong malware indicator
7	e_cparhdr (DOS)	header	DOS header paragraphs	Fixed in legit compilers; tampering = hand-crafted PE
8	Magic (Optional Hdr)	header	PE format magic value	Unexpected format for 64-bit binary signals anomaly
9	Subsystem	header	Target subsystem (GUI/Console)	Driver malware uses kernel-mode value unexpectedly
10	MajorOSVersion	header	Minimum required OS version	Unusual OS targeting or compatibility manifests

Table 6. Analytical Breakdown of Estimated End-to-End Pipeline Latency per PE Binary

Pipeline Stage	Method / Source	Est. Latency
File interception (kernel hook)	Synchronous kernel-level file-open intercept	~1–5 ms (est.)
Feature extraction (PE → EMBER v3)	pefile + EMBER extractor, 804-dim vector	~15–50 ms (est.†)
Model inference (pre-extracted features)	LightGBM scoring (CalibratedClassifierCV)	2.30 ms (measured; P99=2.70 ms)
TLS verdict transmission	TCP/TLS JSON payload to central server	~5–15 ms (est.)
SHAP attribution (optional)	TreeSHAP on SUSPICIOUS verdicts only	~35 ms (on-demand)
Total pipeline (excl. SHAP)	—	~25–75 ms (est.)

† Estimated from published pefile benchmarks [22]; formal end-to-end profiling on production hardware is identified as future work.

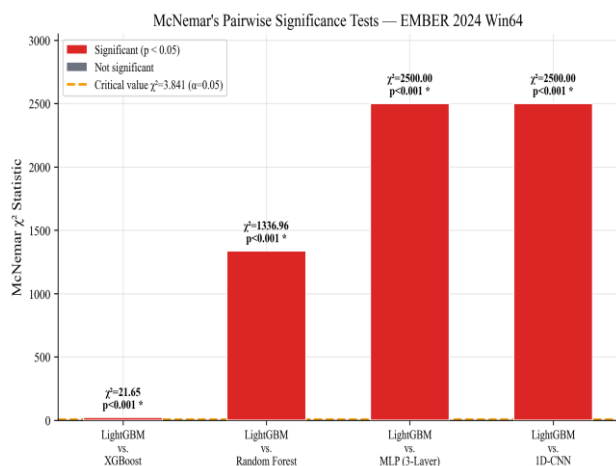


Figure. 9 False Positive Rate vs. True Positive Rate trade-off analysis

each EMBER v3 feature group through rigorous empirical measurement, we perform a true retraining-based ablation study. For each of the nine non-empty feature groups, we remove all features belonging to that group and retrain a fresh LightGBM model from scratch using identical hyperparameters on the remaining features. The resulting test AUC is compared against the baseline model trained on all 804 features.

This constitutes a true ablation experiment: nine separate LightGBM models were retrained from scratch, each with one feature group entirely removed, using the same hyperparameters and early stopping protocol as the primary model. The baseline AUC (all 804 features) for the ablation models is 0.9969, which differs slightly from the primary model's test AUC of 0.9979 (Table 2) because the ablation baseline was retrained independently with a different early stopping convergence point (the primary model converged at 1,506 iterations, whereas the ablation baseline converged at a different iteration count under the same stopping protocol). Both values represent legitimately trained full-feature models; the 0.001 difference reflects the stochastic variability inherent in independent retraining runs. Results are presented in Table 7 and Fig. 10 in decreasing order of discriminative impact.

The highest single-group impact is observed with the header group ($\Delta\text{AUC} = -0.000808$), confirming its highest ranking in individual SHAP features (Table 5). The richheader group shows the second largest impact ($\Delta\text{AUC} = -0.000694$), validating compiler/linker provenance as a key discriminator. It is notable that the byte entropy (256 features, $\Delta\text{AUC} = -0.000058$) and byte histogram (256 features, $\Delta\text{AUC} = -0.000008$) groups together account for 64% of the feature dimensionality but contribute minimally to discriminative power when removed, indicating that dimensionality reduction targeting these groups could lower computational cost with negligible impact on performance. The strings and authenticode groups show marginally positive ΔAUC when removed ($+0.000020$ and $+0.000035$ respectively), suggesting slight overfitting from these features. No individual group removal decreases AUC below 0.996 based on empirical retraining; this empirically confirms complementary discriminative redundancy in the EMBER v3 feature schema.

5.8 Statistical significance testing

To determine whether the difference in performances between the classifiers are statistically significant, we use McNemar's test with continuity correction with the binary correct/incorrect classification vectors on the 240,000 samples test set. McNemar's test is a test of whether two classifiers disagree significantly over the same samples at different rates, it is more proper comparison than accuracy based tests for paired classifier evaluation. Table 8 gives the statistics of the test and the p-value.

All four pairwise comparisons are statistically significant at $p < 0.05$ (Table 8), confirming the

Table 7. True Retraining-Based Feature Group Ablation Study (empirical AUC after group removal)

Group Removed	Features	Remaining	Baseline AUC	Measured AUC	Δ AUC
header	47	757	0.9969	0.9961	-0.000808
datadirectories	34	770	0.9969	0.9966	-0.000287
strings	132	672	0.9969	0.9966	+0.000020
byteentropy	256	548	0.9969	0.9968	-0.000058
richheader	26	778	0.9969	0.9962	-0.000694
histogram	256	548	0.9969	0.9969	-0.000008
section	38	766	0.9969	0.9968	-0.000074
general	7	797	0.9969	0.9968	-0.000030
authenticode	8	796	0.9969	0.9969	+0.000035

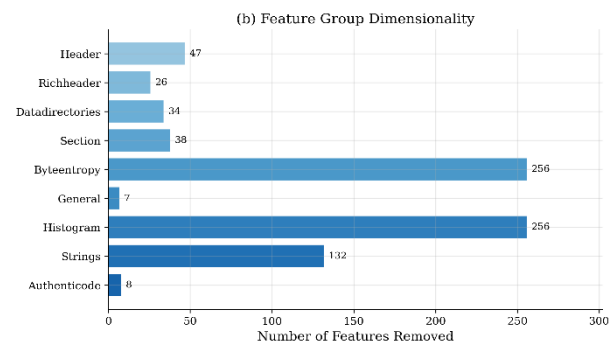
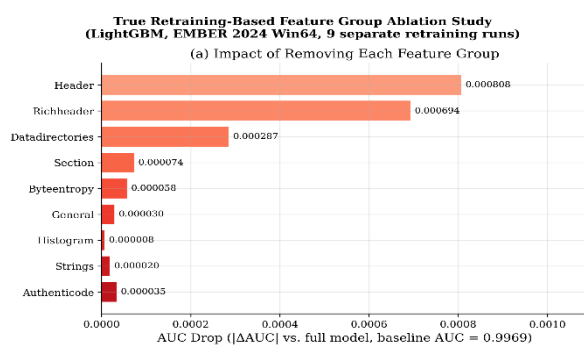
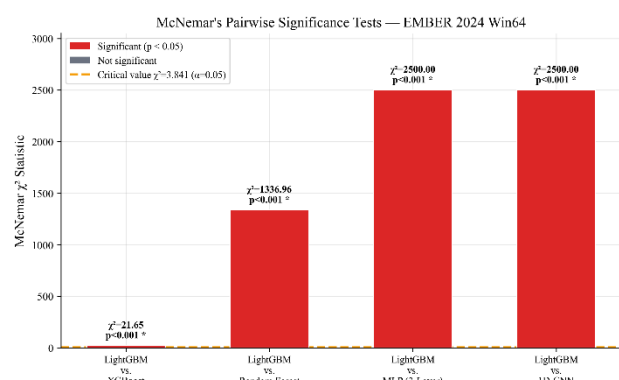


Figure. 10 Feature group ablation analysis: AUC-ROC after removing each group (left) and performance impact magnitude (right)

Table 8. McNemar's Test for Classifier Comparison

Comparison	b ₁₀	b ₀₁	χ^2	p-value
LightGBM vs XGBoost	560	728	21.65	3.27e-6
LightGBM vs Random Forest	788	3,056	1336.96	≈ 0
LightGBM vs MLP	1,366	6,814	3627.12	≈ 0
LightGBM vs 1D-CNN	58,246	85	58,161.38	≈ 0

Figure. 11 Chi-squared tests of comparison of classifiers (pairwise). The critical value is shown as a dashed line at $p = 0.05$

superiority of LightGBM over all competing classifiers including XGBoost ($\chi^2 = 21.65$, $p = 3.27 \times 10^{-6}$), Random Forest ($\chi^2 = 1336.96$, $p \approx 0$), MLP (χ^2

$= 3627.12$, $p \approx 0$), and 1D-CNN ($\chi^2 = 58161.38$, $p \approx 0$).

5.9 False negative and temporal generalization analysis

We study the temporal generalization properties of the framework and characterize the nature of misclassified samples by studying the 3,000 false negatives of the LightGBM classifier on the temporal test set. As the EMBER 2024 test partition is 12 weeks post-training, such false negatives feature malware that has been generated recently with a distributional shift to the training corpus. This temporal distribution change should not be confused with intentional adversarial evasion: these samples indicate concept drift and changing malware construction patterns, rather than an attempt to manipulate features spaces on purpose. The EMBER 2024 dataset has a special evasive challenge partition (malware that did not trigger any commercial AV engines on VirusTotal at the time of collection) and testing on that partition - which is an appropriate adversarial evasion test - is a future task (Section 7.5). The current analysis thus defines the uncertainty of temporal generalization and not adversarial evasion resilience.

Among the 3,000 false negatives, 2,386 (79.5%) have $P(\text{malware}) < 0.1$, and hence would be high-confidence misclassified escapes, which would pass

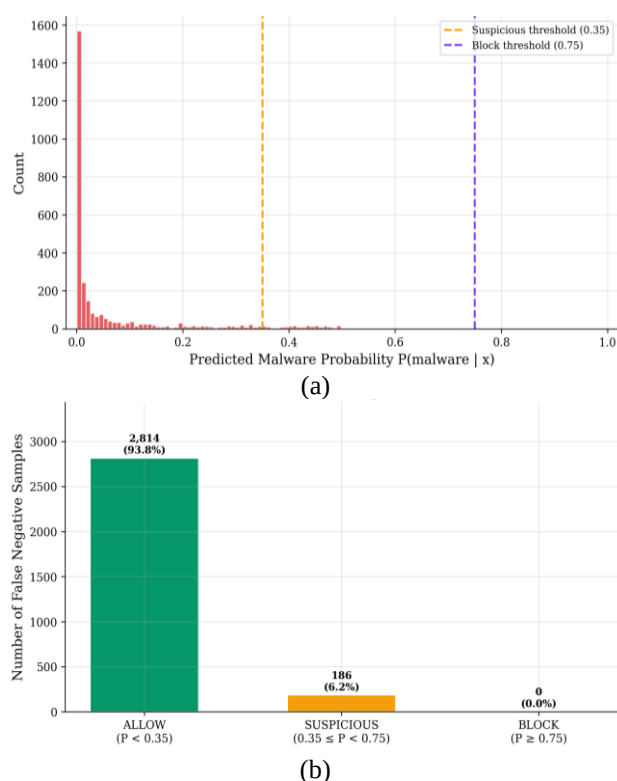


Figure. 12 Characterisation of temporal false negative: (a) probability score distribution of 3,000 misclassified malware samples in concept drift, not adversarial evasion; (b) tri-class zone recovery of false negatives

any threshold-based detector. An additional 222 (7.4%) score between 0.1 and 0.2, and 206 (6.9%) score between 0.2 and 0.35. Importantly, 186 (6.2%) false negatives fall in the range of 0.35–0.50 (tri-class SUSPICIOUS zone). This means that the tri-class scheme recovers over 6% of false negatives that would be missed by a conventional binary classifier with a 0.5 threshold, routing them to analyst review rather than passing them undetected.

The distributions of the scores of 1,962 malware and 1,414 benign samples overlap within the range of 0.3–0.7, which proves that the classifier is uncertain about the results in the case of the temporal shift; the SUSPICIOUS queue represents this 0.40 per cent ambiguous range and leaves it to the analysts.

5.10 Contextual comparison with commercial antivirus

Direct comparison with commercial AV products is out of the scope. Contextually, Joyce et al. [1] indicate single AV engine recall of 60–90 percent on VirusTotal; our LightGBM has a recall of 97.50 percent at 1.92 percent FPR/98 percent TPR, which is favourable to enterprise AV. Assessment of the EMBER 2024 evasive challenge partition is not done

at the present but is instead postponed to work in the future.

6. Prevention decision dashboard and conceptual architecture

In order to support operational use of such a framework we established a distributed and real time monitoring architecture as shown in Fig. 13. The system is designed in a conceptual form of three tiers namely the endpoint layer, central server layer and the human analyst dashboard.

In the final layer, a synchronous kernel level hook is used to intercept file-open operations. The lightweight agent takes in the 804-dimensional EMBER v3 feature vector and does local inference with LightGBM. Since this inference takes less than 3 ms (P99, on pre-extracted features) the primary prevention mechanism (which incorporates automatic blocking of high-confidence threats) does not cause any noticeable latency. After making the prevention decision, the endpoint agent sends a notification using a secure TLS connection to the central server, which includes the tri-class verdict, raw feature vector, and a payload in the form of a JSON-formatted alert.

The central server layer is responsible for persistent event logging, threat intelligence correlation, as well as the computationally expensive task of on-demand SHAP value calculation. The framework conserves local system resources by separating the calculations to make them interpretable and the endpoint.

Lastly, the centralized data is ingested by a bespoke web-based monitoring dashboard (HTML5/CSS3/JavaScript) into Security Operations Center / SOC (analysts). The dashboard displays: per-file tri-class verdicts and probability scores in a live feed; a SHAP attribution panel of SUSPICIOUS-verdict files; a byte-entropy histogram of packing/encryption detection; and aggregate operational statistics.

7. Discussion

7.1 Comparative analysis and operational strengths

The obtained AUC-ROC of 0.9979 on the temporally-separated EMBER 2024 Win64 test set with LightGBM is a decent score in comparison to prior studies of detection of relatively stationary PE malware. Anderson and Roth [4] reported an AUC of 0.9991 on EMBER 2018 with a similar LightGBM

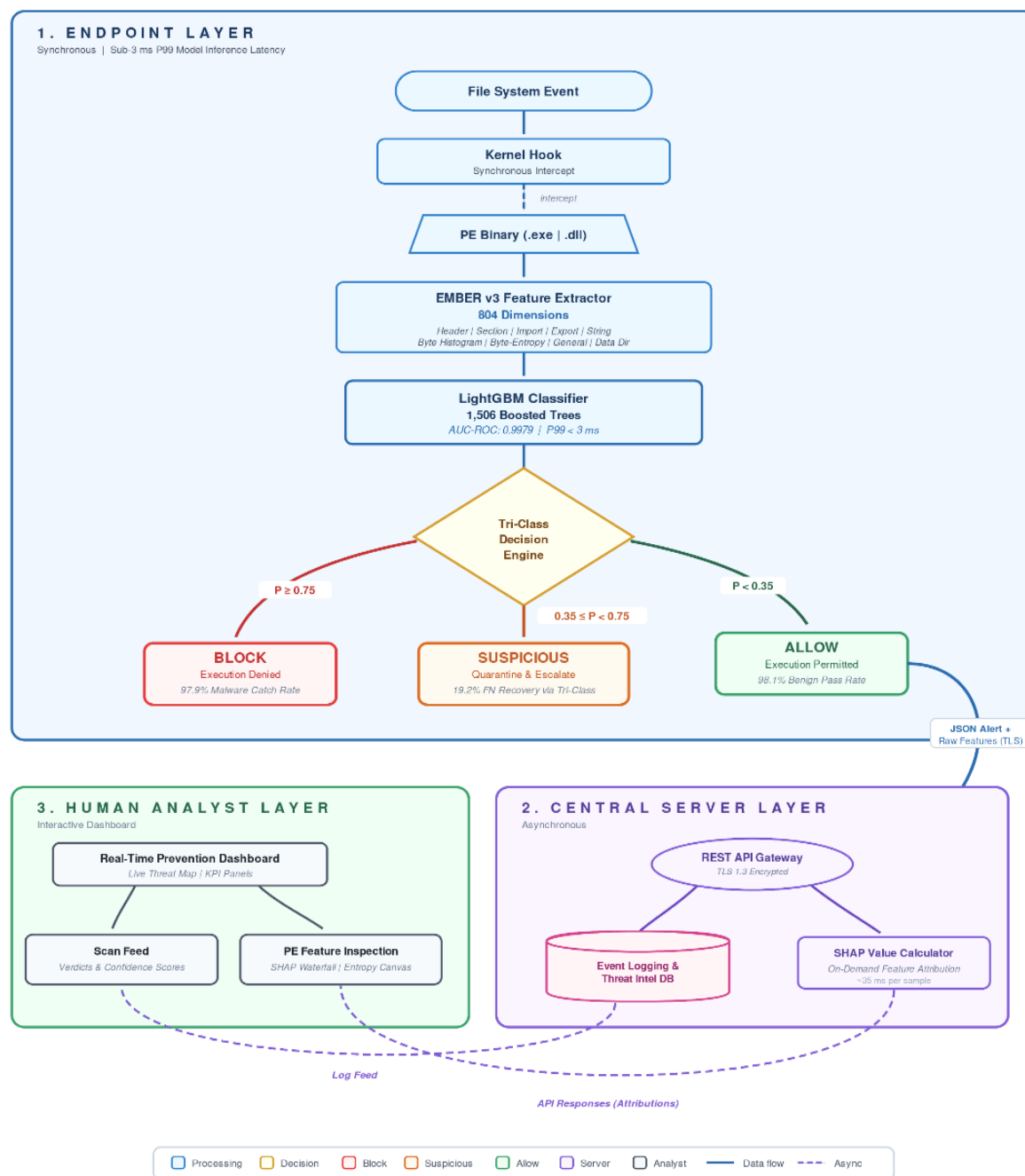


Figure. 13 End-to-end operational architecture of the real-time malware prevention framework showing the data flow from endpoint interception to the analyst dashboard

pipeline; however, their test was done on a random (as opposed to a time-based train/test split), which is not what a real deployer would want to do. The temporal

Split in this case brings about distributional shift which complicates the classification task significantly and the 0.0012 AUCs difference is likely caused by the increased difficulty rather than weakness of the model. Shwartz-Ziv and Armon [23] have demonstrated that gradient-boosted trees are always at the same level as deep learning models on tabular data, and our findings validate this on the

EMBER v3 feature schema: results of LightGBM and XGBoost are statistically equivalent in AUC ($p = 3.27 \times 10^{-6}$), and both 1D-CNN (0.964

From the operational point of view, the framework provides four main benefits. To begin with, the latency of sub-3 ms of the P99 single-sample model inference proves to be efficient in scoring the pre-extracted EMBER features. This is the score of models only; the entire end-to-end latency (including feature extraction 20-50 ms/file [22]) is approximated at 25-75 ms/file (Table 6), which is within sub-second endpoint specifications.

Second, the 1.92% FPR, at 98% TPR, is within (1-5) percentage range that enterprise AV solution can typically run at, and thus is commercially viable in situations where any false positive is costly in terms of productivity. Third, the 31.5 MB-model is much smaller than deep learning alternatives, and can fit resource-limited endpoints.

7.2 Limitations

There are a number of limitations that should be mentioned. First, static characteristics cannot find fileless or LotL malware [22]; this architecture is expected to be used as a layer in a defense-in-depth stack with behavioral analysis. Second, there is temporal degradation: AUC decreases under 0.9998 (cross-validation) to 0.9979 (held-out test) with the 12-week gap. Intermittent retraining will be necessary in production; rate of degradation with time will be taken up later in the job. Third, adversarial robustness is not tested [2, 24]. SHAP transparency is a two-sided sword: it helps analysts, yet can inform enemies on how to manipulate high-importance features. Attack assessment in the feature-space is a crucial second step. Fourth, it can only evaluate Win64 PE binaries; generalization to Win32, .NET, APK, ELF, and PDF is an open query.

7.3 Implications of SHAP feature attribution

The SHAP analysis shows that the classifier is heavily dependent on a few PE structural properties. The prevalence of `SizeOfInitializedData`, `Filesize`, and `SizeOfCode` indicate acquired size-ratio Figure between benign and packed malware [21]. `SecurityCert.VA` (Authenticode presence) is a powerful discriminator but this signal can be weakened as attackers continue to use stolen or fake code-signing certificates.

Empirical evidence (Table 7) that removing any of the groups of features does not decrease AUC below 0.996, confirming the complementary nature of the discriminative structure of the EMBER v3 feature schema. This is a practical strength as far as deployment is concerned, as it means that the classifier does not rely heavily on a particular family of features. Meanwhile, the groups of byte entropy and byte histograms together have 512 of the 804 features (64% of dimensions) but contribute little to the discriminative power (maximum 0.000058 each removed), and, thus, dimensionality reduction or feature selection based on these groups can be of significant computational cost-saving without substantial accuracy loss.

7.4 Tri-class threshold design considerations

One of the design contributions of this work is the tri-class threshold scheme (BLOCK: $p \geq 0.75$; SUSPICIOUS: $0.35 \leq p < 0.75$; ALLOW: $p < 0.35$). To achieve a balance between the rate of malware detection and the rate of benign pass rate and the resulting 0.40% suspicious queue is manageable to SOC operations on a moderate scale, the threshold values ($\tau_7 = 0.75$, $\tau_8 = 0.35$) were calibrated on the validation holdout. However, there are a number of considerations worth discussing. The thresholds are specific to the dataset, and are possibly required to be re-calibrated when the model is re-trained on new data, or deployed in a setting with different levels of malware prevalence. When we consider the high volume enterprise environment with 100,000 or more files per day, the rate of suspicious 0.40% translates to approximately 1,410 files to review by an analyst which may be too many files to be reviewed by a SOC. Organizations ought to redefine thresholds through risk tolerance, capacity of analysts and prevalence of deployment.

7.5 Future work

A number of guidelines on future work follow the above analysis. First, assessment on the EMBER 2024 evasive challenge partition will lead to a stricter test of the resistance of models to samples that initially evaded all the commercial antivirus engines on VirusTotal. Second, further classification of multi-class malware families based on labels given on a family-level by EMBER should be extended to enable more specific triage of threat. Third, cross-format training in all types of EMBER 2024 files (Win32, Win64, .NET, APK, ELF, PDF - six file types) may be used to detect unified across platforms. Fourth, the evaluation of adversarial robustness of feature-space attack models on the high importance features identified by SHAP is essential to understand the practical evasion resilience. Fifth, the assessment of the behavioral tags of EMBER as an additional feature might be added to the structural static analysis that is provided here. Lastly, the frequency of retraining of models during deployment time in sequential temporal slices of the EMBER 2024 training data, may reflect the frequency at which production systems should be retrained.

7.6 Scope of comparison and literature positioning

The comparison table (Table 3) compares five classifiers that were implemented using the same protocol on the EMBER 2024 Win64 partition. This

internal benchmark strictly checks the relative ranking of the classifiers on this dataset and split. Nevertheless, it is not a same-protocol comparison with external methods on EMBER 2024, and this weakness is clearly understood. The current research is framed as an internal benchmark assessment and not a superiority statement, state-of-the-art.

Among the three known early EMBER 2024 adopters, none report as similar results on the Win64 temporal split with EMBER v3 features: Hosahalli Dayanada and Chen [18] use the Android APK malware, Alotaibi and Frikha [19] represent features differently, and Sun and Masum [25] use a zero-shot paradigm. External comparison based on the same protocol is thus impossible at the moment. The initial EMBER 2024 baseline [1] indicates that baseline LightGBM AUC is 0.996 on the standard (non-evasive) test partition, and AUC of the current work is 0.9979 on the same test partition, which is comparable with LightGBM AUC of the dataset authors.

The superiority claims are limited to the internal comparison of classifiers (McNemar-validated); external comparison at the same protocol is left to future research when the EMBER literature matures to the EMBER 2024.

8. Conclusion

This paper presented a tri-class real-time malware prevention framework based on EMBER 2024 Win64 dataset. LightGBM classifier, which has been trained on 1,040,000 PE samples and 804 static features, has an AUC-ROC of 0.9979, F1-score of 0.9805, and false positive rate of 1.92 at 98% detection rate. An innovative tri-class threshold scheme achieves 97.66% malware detection and 98.49% benign pass rate through differentiated automated blocking and analyst queued review. Efficient scoring performance with a mean model inference latency of 2.30 ms (on pre-extracted EMBER features) indicates it can be effectively used in end-to-end prevention pipelines, awaiting validation of end-to-end latency, including feature extraction overhead.

Statistically significant performance of gradient boosted tree methods compared to deep learning baselines ($p < 10^{-7}$) is statistically validated by comprehensive analysis of five classifiers (LightGBM, XGBoost, 1D-CNN, Random Forest, MLP) using McNemar statistical significance testing, but LightGBM and XGBoost are statistically equivalent ($p = 3.27 \times 10^{-6}$). Empirical validation of the true retraining-based ablation (Table 7) demonstrates that no one group removal will yield

AUC values lower than 0.996. The ablation findings support complementary discriminative redundancy of the nine EMBER v3 feature groups by empirical retraining of nine distinct models. Temporal false negative analysis indicates that the tri-class scheme restores 6.2% of misclassified malware into the SUSPICIOUS verdict, offering a better coverage as compared to classical binary classification when temporal distribution shift occurs. This result indicates temporal generalization characteristics but not adversarial evasion resilience because the latter must be assessed on the special EMBER 2024 evasive challenge partition (defined as future work). SHAP-based interpretability analysis, projected onto EMBER v3 semantic feature names, the most significant classifiers between malware and benign Win64 binaries is PE initialized-data section size, Authenticode certificate, and exception table structure - findings were aligned with known malware construction and evasion strategies. The stability of the classifier is verified by five-fold cross-validation (AUC-ROC = 0.9998 + 0.0000).

Conflicts of interest

The authors declare no conflict of interest.

Author contributions

Conceptualization, S.S.A.; methodology, S.S.A. and M.Z.A.; software, S.S.A.; validation, S.S.A., M.Z.A. and A.H.S.; formal analysis, S.S.A.; investigation, S.S.A.; data curation, S.S.A.; writing-original draft preparation, S.S.A.; writing-review and editing, M.Z.A. and A.H.S.; supervision, M.Z.A. and A.H.S.

References

- [1] R. J. Joyce, D. Park, E. Raff, J. Holt, R. Harang and C. K. Nicholas, "EMBER2024: A benchmark dataset for holistic evaluation of malware classifiers", *arXiv preprint arXiv:2506.05074*, 2025.
- [2] A. Afianian, S. Niksefat, B. Sadeghiyan and D. Baptiste, "Malware dynamic analysis evasion techniques: A survey", *ACM Computing Surveys*, Vol. 52, No. 6, Art. 126, pp. 1–39, 2019.
- [3] M. Christodorescu and S. Jha, "Static analysis of executables to detect malicious patterns", In: *Proc. of 12th USENIX Security Symposium*, pp. 169–186, 2003.
- [4] H. S. Anderson and P. Roth, "EMBER: An open dataset for training static PE malware machine learning models", *arXiv preprint arXiv:1804.04637*, 2018.

- [5] M. Z. Abedin and T. A. Mehrub, "Evaluating ensemble and deep learning models for static malware detection with dimensionality reduction using the EMBER dataset", *arXiv preprint arXiv:2507.16952*, 2025.
- [6] A. Shalaginov, S. Banin, A. Dehghantanha and K. Franke, "Machine learning aided static malware analysis: A survey and tutorial", *arXiv preprint arXiv:1808.01201*, 2018.
- [7] M. Yousuf, I. Anwer, A. Riasat, K. Zia and S. Kim, "Windows malware detection based on static analysis with multiple features", *PeerJ Computer Science*, Vol. 9, Art. e1319, 2023.
- [8] M. Miraoui and M. Ben Belgacem, "Binary and multiclass malware classification of Windows portable executable using classic machine learning and deep learning", *Frontiers in Computer Science*, Vol. 7, Art. 1539519, doi: 10.3389/fcomp.2025.1539519, 2025.
- [9] E. Raff, J. Barker, J. Sylvester, R. Brandon, B. Catanzaro and C. K. Nicholas, "Malware detection by eating a whole EXE", In: *Proc. of AAAI Workshop on Artificial Intelligence for Cyber Security (AICS)*, 2018.
- [10] H. D. Pham, T. D. Le and V. H. Nguyen, "Static PE malware detection using gradient boosting decision trees algorithm", In: *Proc. of Future Data and Security Engineering (FDSE)*, pp. 228–236, 2018.
- [11] M. Martina Jose Mary, S. Usharani, P. Manju and S. G. Sandhya, "Detection of ransomware in static analysis by using gradient tree boosting algorithm", In: *Proc. of IEEE International Conference on Smart Computing, Analytics and Networks (ICSCAN)*, pp. 1–5, doi: 10.1109/icscan49426.2020.9262315, 2020.
- [12] P. Qi, W. Wang, L. Zhu and S. K. Ng, "Unsupervised domain adaptation for static malware detection based on gradient boosting trees", In: *Proc. of 30th ACM International Conference on Information and Knowledge Management (CIKM)*, pp. 4264–4268, 2021.
- [13] S. Kattamuri, R. Penmatsa, S. Chakravarty and V. Madabathula, "Swarm optimization and machine learning applied to PE malware detection towards cyber threat intelligence", *Electronics*, Vol. 12, No. 2, Art. 342, 2023.
- [14] S. S. Lad and A. C. Adamuthe, "Improved deep learning model for static PE files malware detection and classification", *International Journal of Computer Network and Information Security*, Vol. 14, No. 2, pp. 14–26, doi: 10.5815/ijcnis.2022.02.02, 2022.
- [15] B. Abdrakhmanov, "Hybrid AI-based static analysis for malware detection: A feature engineering and model optimization approach", In: *Proc. of IEEE Soft Intelligence and Systems in Science and Technology (SIST)*, pp. 1–6, 2025.
- [16] W. Hardy, L. Chen, S. Hou, Y. Ye and X. Li, "DL4MD: A deep learning framework for intelligent malware detection", In: *Proc. of International Conference on Data Mining (DMIN)*, pp. 61–67, 2016.
- [17] J. Saxe and K. Berlin, "Deep neural network based malware detection using two dimensional binary program features", In: *Proc. of 10th International Conference on Malicious and Unwanted Software (MALWARE)*, pp. 11–20, 2015.
- [18] P. Hosahalli Dayananda and Z. Chen, "Re-evaluating Android malware detection: Tabular features, vision models, and ensembles", *Electronics*, Vol. 15, No. 3, Art. 544, doi: 10.3390/electronics15030544, 2026.
- [19] A. Alotaibi and M. Frikha, "Enhancing malware detection using machine learning models on static features", *International Journal of Advanced Computer Science and Applications*, Vol. 17, No. 2, pp. 1–8, 2026.
- [20] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions", In: *Proc. of Advances in Neural Information Processing Systems (NeurIPS)*, pp. 4765–4774, 2017.
- [21] D. Gibert, C. Mateu and J. Planes, "The rise of machine learning for detection and classification of malware: Research developments, trends and challenges", *Journal of Network and Computer Applications*, Vol. 153, Art. 102526, pp. 1–22, 2020.
- [22] Elastic Security Labs, "2024 global threat report", Elastic N.V., Technical Report, 2024, [Online]. Available: <https://www.elastic.co/security-labs/>
- [23] R. Shwartz-Ziv and A. Armon, "Tabular data: Deep learning is not all you need", *Information Fusion*, Vol. 81, pp. 84–90, 2022.
- [24] B. Kolosnjaji, A. Zarras, T. Weidinger, C. Yannikos, G. Webster, X. Eckert and C. Rossow, "Adversarial malware binaries: Evading deep learning for malware detection", In: *Proc. of European Signal Processing Conference (EUSIPCO)*, pp. 533–537, 2018.
- [25] Y. Sun and M. Masum, "PE2Prompt: A large language model-native framework for interpretable static malware analysis", SSRN preprint 5933533, 2025.
- [26] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye and T.-Y. Liu, "LightGBM: A highly efficient gradient boosting decision tree",

- In: *Proc. of Advances in Neural Information Processing Systems (NeurIPS)*, pp. 3146–3154, 2017.
- [27] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system", In: *Proc. of 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 785–794, 2016.
 - [28] L. Breiman, "Random forests", *Machine Learning*, Vol. 45, No. 1, pp. 5–32, 2001.
 - [29] M. Ahmadi, D. Ulyanov, S. Semenov, M. Trofimov and G. Giacinto, "Novel feature extraction, selection and fusion for effective malware family classification", In: *Proc. of ACM Conference on Data and Application Security and Privacy (CODASPY)*, pp. 183–194, 2016.
 - [30] Y. Ye, T. Li, D. Adjeroh and S. S. Iyengar, "A survey on malware detection using data mining techniques", *ACM Computing Surveys*, Vol. 50, No. 3, Art. 41, pp. 1–40, 2017.
 - [31] B. Athiwaratkun and J. W. Stokes, "Malware classification with LSTM and GRU language models", In: *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2481–2485, 2017.
 - [32] MITRE, "ATT&CK enterprise matrix", Accessed: Jan. 2024, [Online]. Available: <https://attack.mitre.org/>
 - [33] Microsoft, "PE format (Windows developer documentation)", Accessed: Jan. 2024, [Online]. Available: <https://learn.microsoft.com/en-us/windows/win32/debug/pe-format>
 - [34] C. E. Shannon, "A mathematical theory of communication", *Bell System Technical Journal*, Vol. 27, No. 3, pp. 379–423, 1948.
 - [35] T. G. Dietterich, "Approximate statistical tests for comparing supervised classification learning algorithms", *Neural Computation*, Vol. 10, No. 7, pp. 1895–1923, 1998.
 - [36] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj and D. J. Inman, "1D convolutional neural networks and applications: A survey", *Mechanical Systems and Signal Processing*, Vol. 151, Art. 107398, pp. 1–21, 2021.
 - [37] M. Imran, A. Appice and D. Malerba, "Evaluating realistic adversarial attacks against machine learning models for Windows PE malware detection", *Future Internet*, Vol. 16, No. 5, Art. 168, 2024.
 - [38] R. Yumlembam, B. Issac and S. M. Jacob, "Enhancing decision-making in Windows PE malware classification during dataset shifts with uncertainty estimation", *Knowledge-Based Systems*, Vol. 331, Art. 114723, 2025.