

A Comparative Analysis of Static Feature Extraction Techniques for PE File Malware Detection using LightGBM and Gradient Boosting Variants

Aleksandar Sandro Cvetković^{1*}, Snježana Stanišić², Saša Adamović¹

¹ Faculty of Computing and Informatics, Sinergija University, Bijeljina, Bosnia and Herzegovina

² Faculty of Business Economics, Sinergija University, Bijeljina, Bosnia and Herzegovina

*ascvetkovic@sinergija.edu.ba

Abstract – Efficiency of traditional signature-based detection has been severely diminished due to the exponential growth of sophisticated malware. For this reason, the cybersecurity industry has pivoted toward Machine Learning (ML) solutions. This paper presents a comprehensive comparative analysis of machine learning models for static malware detection using Windows Portable Executable (PE) files. By evaluating the efficiency of feature extraction methodologies, specifically targeting structural metadata, byte histograms, and import tables, we demonstrate the limitations of deep learning architectures, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), which often suffer from high computational overhead and the curse of dimensionality. In contrast, Gradient Boosted Decision Trees (GBDTs) such as XGBoost, CatBoost, and LightGBM establish a definitive performance upper bound for tabular PE metadata. Our findings emphasize that, when combined with optimized feature engineering, LightGBM achieves a superior speed-to-accuracy ratio, making it an ideal candidate for real-time, resource-constrained endpoint deployment.

Keywords – PE Malware, Static Analysis, Feature Extraction, LightGBM, Python, Gradient Boosting, EMBER 2024.

Introduction

The exponential spread of malware has rendered manual reverse engineering and traditional signature-based detection obsolete [1]. As attackers rapidly automate the generation of polymorphic and packed malware, the 'breakout time' for attackers has significantly decreased, making human-driven analysis impossible and leading the cybersecurity community to strive toward automated, predictive solutions. In order to counter these accelerated threat lifecycles, the industry is increasingly turning to Machine Learning (ML) models trained on static analysis data [2].

By extracting structural metadata (such as headers, section properties, Import Address Tables (IAT), and byte-level

entropy) from Windows Portable Executable (PE) files prior to execution, static ML models provide a critical, zero-day capable first line of defense. This approach bypasses the high computational costs and runtime risks typically associated with dynamic execution in sandbox environments, [3].

While deep learning architectures such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) have been thoroughly researched for this task, they often behave as 'black boxes' and struggle with the curse of dimensionality. Furthermore, these models impose high computational demands and latency constraints that are largely impractical for real-time endpoint security [3], [4]. Recent literature points to a distinct gap in feature engineering, when structural PE features are engineered and compressed efficiently, tree-based models offer a demonstrably superior speed-to-accuracy ratio for tabular PE files [1].

Accordingly, research has shifted toward Gradient Boosted Decision Trees (GBDTs), specifically LightGBM, XGBoost, and CatBoost as the state-of-the-art for tabular malware data [1]. This study evaluates feature extraction efficiency, focusing primarily on Python libraries (such as pefile) to avoid version-induced feature drift, and systematically benchmarks LightGBM against its GBDT counterparts to establish optimal deployment strategies for resource-constrained environments.

I. LITERATURE REVIEW

For static malware analysis, simple string matching just doesn't cut it anymore. Instead, researchers have moved toward extracting complex, multidimensional structural features. Standardized benchmarks like the EMBER dataset changed the game by introducing a 2,381-dimensional vector. This vector covers general file info, Common Object File Format (COFF) optional headers, and section entropy [5].

Table 1. Summary of existing literature on PE malware detection

Specific Static Features Extracted	Machine Learning Model Used	Performance Results	Sources
PE Header, COFF headers, import/export tables, section properties, section entropy	LightGBM, XGBoost, CatBoost, Random Forest, Extra Trees, MLP	LightGBM: 0.9852 AUC, 0.9421 F1-Score	[1]

API call frequency, Windows-based APIs, file system interactions, registry manipulation	Random Forest, XGBoost, Logistic Regression, SVM, NB, KNN	Random Forest: 96% Accuracy, 0.96 F1-Score, 0.99 Precision	[2]
47 diagnostic features: static properties, process/file/registry activity, network telemetry	CatBoost, LightGBM, XGBoost, Random Forest, TabNet, FT-Transformer	CatBoost: 99.0% Accuracy; LightGBM: 98.7% Accuracy	[3]
EMBER feature set (format-agnostic byte/entropy histograms, string info, parsed headers)	MalConv (CNN), LightGBM (via GBDT-EMBER baseline)	GBDT: 0.75 TPR @ 0.0028 FPR	[4]
2,381 EMBER features: Byte distribution, entropy, headers, imports (excluding timestamps)	Residual Neural Network (ResNet), LightGBM (Baseline)	Improved ResNet: 92.58% Accuracy, 0.9175 F1-Score	[5]
965 features: Opcodes (93), Registers (26), Windows API (794), Section attributes, File size	GIN (Graph Isomorphism Network), MLP, SVM, ResNet-50	GIN (Adv): 99.5% Accuracy; MLP: 97.9% Accuracy	[6]
Fuzzy-import hashing, permissions, opcodes, API calls, grayscale image matrices	CNN, RNN, LSTM, DNN, Transformers	Various; Fischer et al.: 99.2% ROC-AUC	[7]
Binary code, assembly instructions (.text), source code (decompiled), API call traces	LightGBM (Baseline), CodeBERT, GPT-2, stacked BiLSTM	LightGBM: Highly competitive baseline	[8]
EMBER metadata (COFF headers, data directories, imports unrolled into flat arrays)	XGBoost, Deep Learning (Binary/Multi-class), Autoencoders	XGBoost: 97.76% Accuracy, 0.9776 F1-Score	[9]
Linear/Logical opcode sequences, weighted undirected graphs, PageRank, node degree	LightGBM, XGBoost, CatBoost, Graph Neural Networks (OSGNN)	Consensus Voting: 98.29% Accuracy, 0.9746 MDR	[10]
AST-JS nodes, node combinations, lexical/host-based features, Doc2Vec	LightGBM, XGBoost, RandomForest, FastText, LSTM	XGBoost: 0.9989 Recall, 0.9909 F1-Score	[11]
470 dynamically observed behaviors (API frequencies), static sensitive APIs, permissions	LightGBM, XGBoost, Random Forest	LightGBM: 93.95% Accuracy, 94.1% Precision	[12]
79 PE metadata features: ImageBase, Subsystem, CheckSum, MajorSubsystemVersion	Random Forest, Decision Tree, XGBoost, MLP, Conv1D, LSTM	Random Forest: 95.95% Accuracy; XGBoost: 95.69% Accuracy	[13]
"Radian" features (statistical dispersion), traffic patterns, flow-based features	TabLoRA (TabNet + LoRA), Random Forest, Decision Tree, KNN	TabLoRA: 99.97% Accuracy, 99.98 F1-Score	[14]
2,381 raw static features, byte histograms (hex values like 0x7D, 0x44), string entropy	LightGBM (Target), MalConv, GAMMA (Evolutionary black-box)	Adversarial evasion: >70% success against GBDTs	[15]
EMBER 2024 histogram-based static features (float32 compression)	Stacking ensemble: LightGBM, CatBoost, XGBoost (meta-learner: LightGBM)	Ensemble: 86.99% Accuracy, 0.87 F1-Score, 0.9473 AUC	[16]
PE Optional Header Fields (PEOHF), 1323 features (Balanced) / 1877 (Unbalanced)	SMO (SVM), J48 (Decision Tree), Simple Logistic, Random Forest	DFS/MI + SMO: 98.677% Accuracy (BD), 99.308% (UBD)	[17]
Semantically extracted raw bytes: Header, Code, Data, and integrated File	CNN (MalConv-based architecture for multi-classification)	Header-only: 99.54% Accuracy; File: 98.06% Accuracy	[18]
EMBER feature vector (2,381 features), SHA digest metadata, descubrimiento timestamp	LightGBM (calibrated via Isotonic Regression)	Calibrated LightGBM: 0.99712 F1-Score	[19]
Ordering concatenation: PE header, opcode sequence, and API import features	Stacking ensemble: SVM + Random Forest (meta-learner: Logistic Regression)	Hybrid Ensemble: 98.91% Accuracy, 98.91% F1-Score	[20]
Static Joint analysis of API calls and Dynamic Link Library (DLL) data	XGBoost, CNN-GRU-3, Random Forest, KNN, Extra Trees Classifier	XGBoost: 99.81% Accuracy; CNN-GRU-3: 99.37% Accuracy	[21]

Feature Extraction Challenges

Since malware authors are constantly switching up their code to tank model accuracy over time, concept drift remains a massive headache [5]. Static analysis is also wide open to problem space attacks. These tricks involve adding padding bytes or messing with non-critical header regions (like the DOS header) just to flip the classification results [15], [4]. Reproducibility is another issue. Version gaps in LIEF libraries often cause feature drift, so it's usually better to stick with clean Python alternatives like pefile [16].

The Transition to Gradient Boosting

Early approaches leaned heavily on the Random Forest algorithm, hitting accuracy levels up to 95.95% using limited PE metadata [13]. But as datasets ballooned into the millions, the focus shifted toward Gradient Boosted Decision Trees (GBDT) type models. A prime example of this scale is the EMBER 2024 dataset. While XGBoost has proven to have a high recall for analyzing opcode sequences [10], LightGBM has emerged as the state-of-the-art choice for tabular static data. It's simply more efficient, using significantly less memory and offering faster training times [8]. Comparative benchmarks show LightGBM hitting an AUC-ROC of 0.9852, outperforming both Random Forest and Extra Trees when it comes to inference speed [1].

Deep Learning and Hybrid Methodologies

The researchers in [7] conducted a deep dive into using deep learning for malware detection. They explored how models like CNNs, LSTMs, and Transformers handle features like fuzzy-import hashing, permissions, and opcodes. Their results show that specific architectures hit an impressive 99.2% AUC-ROC across various environments. Building on the use of neural networks, the authors of [18] introduced a semantic classification approach using a MalConv-based CNN. By pulling raw bytes strictly from specific semantic locations (like header, code, and data sections) they proved that analyzing the header alone can reach an accuracy of 99.54%.

Unified Feature Evaluation and Hashing

Addressing the need to for standardized feature assessment, the study in [9] presents a unified evaluation of learning-based similarity techniques. Authors did this by parsing EMBER metadata (specifically COFF headers and data directories) into flattened arrays. The research confirms that XGBoost is highly effective for this task, hitting an accuracy of 97.76% and an equivalent F1-Score.

Network, Web-based, and IoT Attack Detection

Beyond traditional PE binaries, machine learning is also making waves in the broader landscape of cyber defense. As shown in [14], a framework called TabLoRA, which pairs TabNet with Low-Rank Adaptation (LoRA) was used to analyze "Radian" features. These features focus on statistical dispersion and traffic patterns, and the combination hit a near-perfect accuracy of 99.97% in detecting network anomalies. In a similar vein, the research in [11] proved how effective XGBoost can be for spotting web-based attacks. By looking at

AST-JS nodes and Doc2Vec features, that study recorded a 0.9989 recall and a 0.9909 F1-Score. More recently, a 2026 study [3] focused on Industry 6.0 by evaluating 47 different static and network telemetry features for IoT malware detection. The findings revealed that CatBoost and LightGBM are still the ones to beat, outperforming other models with accuracies of 99.0% and 98.7%, respectively.

Ensemble and Calibrated Approaches

To combat concept drift, recent studies highlight the superiority of complex, calibrated ensembles and the improvement of classification margins. The authors of [19] utilized Isotonic Regression to calibrate a LightGBM model using EMBER and SHA digest metadata, achieving an F1-score of 0.99712 in their fight against data drift. In a parallel advancement, a 2025 study [20] proposed a hybrid Support Vector Machine - Random Forest (SVM-RF) stacking ensemble that fused PE headers, opcode sequences, and API imports. Supervised by a Logistic Regression meta-learner, this hybrid approach reached an impressive 98.91% accuracy and F1-score.

II. METHODOLOGY AND FEATURE EXTRACTION

Data Collection

The primary data source for this research is the EMBER 2024 dataset. While the original EMBER release focused on 1.1 million files, the 2024 update has expanded significantly, now featuring over 3.2 million samples across six different file formats [5], [16]. To ensure the study covers a diverse range of current threats, additional samples were pulled from MalwareBazaar and VirusShare, focusing on recent families like Ransomware, Trojans, and Botnets [2]. These datasets provide the high-confidence labels needed to build robust classifiers that can stand up to modern obfuscation techniques [9].

The Extraction Pipeline

The pipeline uses a pure Python implementation to keep things stable in research environments. While the LIEF library handles the heavy lifting for advanced binary instrumentation [16], the pefile library is the go-to tool for pulling structural metadata. This choice specifically avoids the version-heavy "feature drift" that usually plagues C++ based parsers [16]. Raw files are then crunched into flattened numerical arrays, which keeps byte-level histograms and structural flags intact without the massive overhead of a full disassembly.

Feature Categories

The feature engineering process focuses on specific domains of the PE structure:

General File Info and Structural Metadata

This layer includes basic metrics like file size, global entropy, and digital signature status. High global entropy is a classic red flag, often pointing to packed or encrypted payloads designed to hide malicious code [15].

The extraction also deep-dives into the `IMAGE_FILE_HEADER` (tracking machine types and section counts) and the `IMAGE_OPTIONAL_HEADER` (capturing the `AddressOfEntryPoint`, `ImageBase`, and `Subsystem`). Key fields like `Checksum` and `MajorSubsystemVersion` are prioritized because they have high "discriminative power", meaning they are very effective at helping the model tell a legitimate file from a fake one [13].

Content and Organization (Section Data and IAT)

This involves digging into the internal sections of the file, how many there are, their permissions (Read, Write, Execute), and their individual entropy levels. Malicious files frequently use odd section names or show high entropy in the `.rsrc` (resource) or `.reloc` (relocation) sections to hide data [16]. The

pipeline also pulls the list of DLLs and Windows API calls from the Import Address Table (IAT). The frequency of sensitive APIs, especially those used for registry manipulation or network communication, is a major tell for malicious intent [2].

Statistical and Format-Agnostic Data

Finally, the system extracts fixed-length vectors, such as 256-byte histograms and sliding-window byte-entropy histograms, along with string statistics like the count of URLs and IP addresses. These features provide a strong statistical baseline that works regardless of the specific file format. Best of all, they allow the model to make highly accurate decisions without the heavy processing power needed for a full code disassembly [5], [4].

Table 2. Proposed static PE feature set for LightGBM classification

Category	Specific Features	Justification	Sources
Structural Metadata (Headers)	DOS Header (e_lfanew), COFF/File Header (NumberOfSections, Characteristics), Optional Header (ImageBase, AddressOfEntryPoint, Subsystem).	GBDT ensembles are statistically efficient on tabular metadata; they excel at modeling high-order, non-additive feature interactions through hierarchical partitioning (e.g., specific linker flags combined with entry point anomalies) that traditional linear models miss.	[1], [13], [17]
Content and Organization	Section Headers (.text, .data names/sizes), Data Directories (DataDirectories 8), Opcode sequences, Import/Export tables (DLL names, Windows API calls like <code>NtAllocateVirtualMemory</code>).	Tree-based models naturally handle heterogeneous feature groups (numeric counts vs. categorical API names) without extensive preprocessing; they are robust to code changes because they learn decision paths where the predictive meaning of a feature is highly conditional on others.	[10], [2]
Statistical and Format-Agnostic	256-Byte Histograms, Byte-Entropy Histograms (sliding window), String information (counts of URLs/IPs), String Entropy.	These models excel at modeling non-linear thresholds in high-dimensional tabular data (e.g., thresholding global entropy to identify packing) and can achieve inference speeds up to 64 times faster than deep learning models while capturing sparse, high-entropy signals.	[5], [15], [4], [9]

LightGBM Algorithm Specification

To manage high-dimensional vectors (like the 2,381 features found in EMBER), we rely on LightGBM. This model stands out because it uses two specific techniques to keep things running fast without sacrificing precision. Its leaf-wise tree growth naturally prioritizes features with the highest variance. This makes it incredibly efficient at thresholding high-dimensional, format-agnostic data, such as byte histograms. Plus, its built-in categorical handling processes structural header flags seamlessly, avoiding the massive memory drain usually caused by one-hot encoding [14], [1].

Key LightGBM Optimizations

Gradient-based One-Side Sampling (GOSS)

Instead of chewing through every single data point to calculate information gain, GOSS focuses on instances with large gradients, essentially the "under-trained" data. It then takes a random sample of the instances with small gradients.

This massively cuts down on the computational heavy lifting while keeping the model's accuracy sharp [5], [8].

Exclusive Feature Bundling (EFB)

Since PE feature sets are often "sparse", meaning many features are mutually exclusive and rarely active at the same time, EFB bundles these features into a single column. This reduces the total feature count and speeds up the training process without losing any critical information [8].

III. COMPARATIVE ANALYSIS

Overview of the Benchmarking Landscape

Evaluation of modern malware classifiers requires a multi-dataset approach to account for the growing number of complex threats. Established benchmarks such as EMBER 2018 [1], BODMAS [5], and the Kaggle PE dataset [13] have provided the foundational corpus for static analysis, focusing primarily on structural metadata and byte-level histograms.

Table 3. Benchmark results of gradient boosting and neural models for PE malware detection

Dataset	Algorithm	Accuracy (%)	Precision	Recall	PR AUC / F1	Ref.
EMBER 2018	LightGBM	—	—	—	0.9421	[1]
EMBER 2024	Stacking Ensemble (LGBM/XGB/CB)	86.99	0.87	0.87	0.87	[16]
BODMAS	Improved ResNet (DRBCE)	91.28	—	—	0.9056	[5]
EMBER (Binary)	XGBoost	97.76	0.9776	0.9776	0.9776	[9]
OSG Academic	LightGBM	96.48	—	0.9803	—	[10]
C-Prot PE	XGBoost	99.81	99.62	100.00	99.81	[21]
IoT Malware (Any.Run)	CatBoost	99.00	0.990	0.979	0.989	[3]
Big-15	MLP (Warm)	97.90	—	—	—	[6]
Kaggle PE	Random Forest	98.91	—	99.02	98.91	[13]
AST-JS	XGBoost	—	0.9698	0.9981	0.9838	[11]

Table 3 shows that transitioning to the high-complexity EMBER 2024 version 3 framework [16] introduces a rigorous environment characterized by adversarial samples designed to mimic benign software. Robust model evaluation extends across various architectures, including Internet of Things (IoT) specific malware samples from the Any.Run [3] environment to large-scale binary classifications [9], [10]. Specialized datasets focusing on specific feature domains, such as the opcode-sequence based Big-15 [21] and Abstract Syntax Tree (AST-JS) representations [10], are essential to ensure that models maintain resilience against obfuscation across both Windows PE and web-based attack vectors.

Performance Metrics and the EMBER v3 Challenge Gap

A critical review of the metrics in Table 3 reveals a significant performance gap when transitioning from legacy distributions to modern, evasive datasets. While earlier iterations of the EMBER dataset allowed models like LightGBM to achieve a PR AUC of 0.9421 [1] and XGBoost to reach an accuracy of 97.67% in binary tasks [9], these figures drop sharply when faced with current threats.

The Stacking Ensemble evaluated on EMBER 2024 achieved an accuracy of only 86.99% [16]. This degradation

highlights the impact of concept drift, where the statistical properties of malware features change over time.

Concept drift makes older models less efficient. The F1-Score of 0.87 [16] serves as a realistic baseline for hardened static detectors, which stands in sharp contrast to the F1-Score of 0.9776 seen in simpler binary variants [9]. This gap underscores the difficulty of detecting samples that are specifically hardened to bypass traditional byte-histogram features.

Algorithmic Comparison: GBDT vs. Neural Architectures

The data consistently support the effectiveness of Gradient Boosted Decision Trees (GBDT) over Deep Learning architectures for tabular PE metadata. In Table 3, the XGBoost and CatBoost variants achieve excellent results across multiple tests. XGBoost reached an accuracy of 99.81% on the C-Prot PE dataset [21] and an accuracy of 99.00% on the IoT Malware dataset [3]. The AST-JS feature learning approach using XGBoost yielded an F1-Score of 0.9838 [11].

In contrast, neural network implementations show lower relative performance on these feature sets, an improved ResNet model on BODMAS recorded an accuracy of 91.28% [5], while Warm MLP on Big-15 achieved 97.90% [6]. These results indicate that GBDT variants are superior in handling the high-dimensional, non-linear, and heterogeneous feature sets typical of PE headers and metadata, without the massive computational overhead that neural architectures require.

Feature Extraction Complexity and Evasion

The deciding factor in maintaining high detection rates amidst adversarial evolution is the sophistication of the feature engineering process. The high accuracy of LightGBM on the OSG Academic dataset (96.48%) [10] and the Random Forest model on Kaggle PE (98.91% F1-Score) [13] demonstrate that moving beyond raw byte histograms is crucial.

Sophisticated extraction techniques allow models to maintain exceptional sensitivity, as evidenced by the 100.00% recall achieved by XGBoost on the C-Prot PE dataset [21] and the 0.9981 recall on AST-JS features [11]. These high recall values prove that advanced static feature extraction, incorporating graph-based opcode sequences or semantic code structures, remains the primary defense against adversarial evasion. By providing the classifier with a nuanced representation of the file's intent, these methodologies bridge the gap between static efficiency and dynamic-level detection depth.

IV. DISCUSSION AND RESULTS

Synthesis of Comparative Findings

Empirical evidence from comparative analysis highlights a key reality in static malware detection, algorithmic complexity cannot compensate for inadequate feature engineering when dealing with adversarial evolution. As demonstrated by the performance degradation between the legacy EMBER 2018 dataset [1] and the evasive payloads in the EMBER 2024

framework [16], static metadata is highly susceptible to concept drift [5].

Despite this aggressive evolution, Gradient Boosted Decision Trees (GBDTs) consistently establish the upper performance bound for tabular PE metadata. Models such as XGBoost [9], [21] and CatBoost [3] dramatically outperform deep neural architectures like Multilayer Perceptrons (MLP) [6] because tree-based ensembles natively isolate non-linear structural anomalies without the massive computational overhead required to map sparse, heterogeneous data in a neural network.

Algorithmic Efficiency: Leaf-wise vs. Level-wise Growth

A key observation from the comparative data is the superior speed-to-accuracy ratio of LightGBM, which enables it to maintain highly competitive metrics (e.g., an accuracy of 96.48% on the OSG Academic dataset [10]) while achieving a drastic reduction in training and inference time compared to other tree-based models.

This processing superiority is fundamentally rooted in its tree growth strategy. Traditional algorithms, such as Random Forest [13] and standard XGBoost implementations, typically utilize level-wise (depth-first) tree growth. Level-wise growth expands all nodes at the same depth indiscriminately, calculating unnecessary splits for low-variance leaves, which wastes computational resources on uninformative data.

In contrast, LightGBM employs leaf-wise (best-first) tree growth. It strictly expands the specific leaf that yields the maximum delta loss (variance reduction), regardless of its depth. By focusing exclusively on the most informative splits, LightGBM converges toward optimal decision boundaries much faster. When this leaf-wise expansion is paired with Gradient-based One-Side Sampling (GOSS), which filters out well-trained instances, and Exclusive Feature Bundling (EFB) for compressing sparse PE metrics, LightGBM effectively bypasses the memory and latency bottlenecks that hinder both traditional GBDTs and Deep Learning models.

Feature Interpretation: Benign vs. Malicious Profiles

The true strength of GBDT architectures lies in their ability to dynamically determine the threshold of structural anomalies generated by modern feature extraction pipelines. To illustrate why sophisticated extraction techniques reliably flag evasive malware (yielding recall rates as high as 0.9981 [11] and 100.00% [21]), it is useful to examine a conceptual snippet comparing a benign PE header with a malicious one:

Standard Benign PE Profile

A legitimate executable typically exhibits conventional section names (such as .text, .data, .rsrc), average global entropy in the range of 4.5 to 5.5, and .rsrc (resource) section entropy around 3.0. Its Import Address Table (IAT) is usually densely populated with standard user-interface and system execution APIs (such as user32.dll and advapi32.dll).

Evasive Malicious PE Profile

A packed or encrypted malware variant often displays anomalous, randomized, or missing section names (such as

.upx0 and .xgf1). Its global entropy frequently approaches 7.8 (nearing the theoretical maximum of 8.0 for compressed/encrypted data), with suspiciously high entropy localized specifically within the .rsrc or .reloc sections to conceal the true payload. The IAT is often artificially thinned, utilizing only a few dynamic loading APIs (such as LoadLibraryA and GetProcAddress) to reconstruct the malicious payload in memory post-execution.

Models like LightGBM and XGBoost excel precisely because their leaf-wise decision paths naturally form logical gates around these specific anomalies (such as branching when .rsrc entropy > 7.2 and total IAT imports < 5). By bridging the gap between raw statistical data and structural intent, advanced static feature extraction remains the most robust defense against adversarial evasion.

V. CONCLUSION AND FUTURE WORK

Conclusion

The rapid escalation of polymorphic and packed malware has necessitated a transition from traditional signature-based detection to machine learning models based on static analysis. This research presented a comprehensive evaluation of feature extraction techniques for Windows Portable Executable (PE) files, comparing the performance of Gradient Boosted Decision Trees (GBDT), specifically LightGBM, XGBoost, and CatBoost against Deep Learning architectures.

Comparative analysis definitively shows that for high-dimensional, structured tabular metadata (such as EMBER feature sets), tree-based ensembles significantly outperform neural networks in both predictive accuracy and computational efficiency. Models like XGBoost and LightGBM consistently achieved near-ceiling accuracy across multiple benchmarks. LightGBM proved exceptionally suitable for real-time endpoint implementation. Its use of leaf-wise tree growth, Gradient-based One-Side Sampling (GOSS), and Exclusive Feature Bundling (EFB) enables rapid processing of sparse, heterogeneous data, such as byte histograms, header flags, and section entropy without the massive memory overhead required by deep learning counterparts.

The evaluation highlighted the critical challenges of concept drift and adversarial evasion. The observed performance degradation between legacy datasets (such as EMBER 2018) and modern, evasion-heavy environments (such as EMBER 2024) proves that raw algorithmic power cannot compensate for outdated feature representations. Sophisticated, semantically aware static feature extraction remains the most robust defense against attackers who manipulate non-critical headers or inject padding to bypass detection.

Future Work

While optimized GBDT models operating on static PE features offer an excellent speed-to-accuracy ratio for pre-execution filtering, highly obfuscated or packed payloads can still conceal their true malicious intent from static parsers. To address this limitation, future research must pivot toward hybrid detection frameworks.

The logical next step is combining highly efficient static structured features (such as COFF headers and global entropy) with dynamic behavioral features extracted during sandbox detonation or emulation. The integration of dynamic telemetry, such as API call frequencies, file system interactions, and registry manipulations can provide the classifier with a definitive blueprint of the malware's execution logic.

Future architectures should explore advanced stacking ensembles or TabNet-based models that merge LightGBM's rapid static triage capabilities with Recurrent Neural Networks (RNNs) or Random Forests trained on sequential dynamic behaviors. This hybrid approach would maximize both the rapid pre-execution efficiency of static analysis and the post-execution depth of behavioral profiling. Ultimately, this would provide a more resilient defense against zero-day attacks and adversarial concept drift.

REFERENCES

- [1] D.-C. Horvath and I. Zsigmond, "A Comparative Benchmark of Machine Learning Models for Static Malware Analysis: From EMBER 2018 to the Challenges of 2024," in *Proceedings of the 18th International Conference on Agents and Artificial Intelligence*, Marbella, Spain: SCITEPRESS - Science and Technology Publications, 2026, pp. 2284–2291. doi: 10.5220/0014218700004052.
- [2] D. Z. Syeda and M. N. Asghar, "Dynamic Malware Classification and API Categorisation of Windows Portable Executable Files Using Machine Learning," *Appl. Sci.*, vol. 14, no. 3, p. 1015, Jan. 2024, doi: 10.3390/app14031015.
- [3] M. Maghanaki, S. Keramati, F. F. Chen, and M. Shahin, "Systematic Evaluation of Machine Learning and Deep Learning Models for IoT Malware Detection Across Ransomware, Rootkit, Spyware, Trojan, Botnet, Worm, Virus, and Keylogger," *Sensors*, vol. 26, no. 6, p. 1750, Mar. 2026, doi: 10.3390/s26061750.
- [4] A. Ponte, D. Trizna, L. Demetrio, B. Biggio, I. T. Ogbu, and F. Roli, "SLIFER: Investigating performance and robustness of malware detection pipelines," *Comput. Secur.*, vol. 150, p. 104264, Mar. 2025, doi: 10.1016/j.cose.2024.104264.
- [5] W. Mailliet and B. Marais, "Optimized Deep Learning Models for Malware Detection under Concept Drift".
- [6] A. S. Li, A. Iyengar, A. Kundu, E. Bertino, and A. Iyengar, "Revisiting Concept Drift in Windows Malware Detection: Adaptation to Real Drifted Malware with Minimal Samples," in *Proceedings 2025 Network and Distributed System Security Symposium*, San Diego, CA, USA: Internet Society, 2025. doi: 10.14722/ndss.2025.240830.
- [7] A. Bensaoud, J. Kalita, and M. Bensaoud, "A Survey of Malware Detection Using Deep Learning".
- [8] H. Jelodar, M. Meymani, S. Bai, R. Razavi-Far, and A. A. Ghorbani, "SBAN: A Framework & Multi-Dimensional Dataset for Large Language Model Pre-Training and Software Code Mining," 2025, *arXiv*. doi: 10.48550/ARXIV.2510.18936.
- [9] U. Prasad and A. Chawla, "A Unified Evaluation of Learning-Based Similarity Techniques for Malware Detection," 2026, *arXiv*. doi: 10.48550/ARXIV.2602.15376.
- [10] A. Mello, V. G. Lemos, F. Barbosa, and E. Simoni, "Malware identification on Portable Executable files using Opcodes Sequence," in *Anais do XVI Congresso Brasileiro de Inteligência Computacional*, SBIC, Dec. 2023, pp. 1–8. doi: 10.21528/CBIC2023-006.
- [11] S. N. Wangari, "A Machine Learning Approach to Web-based CyberAttack Detection," Kobe University (Graduate School of Engineering), 2021. [Online]. Available: <https://da.lib.kobe-u.ac.jp/da/kernel/D1008170/D1008170.pdf>
- [12] E. G. Villarreal Enriquez and J. Gutiérrez-Cárdenas, "Dynamic Malware Analysis Using Machine Learning-Based Detection Algorithms," *Interfaces*, no. 019, pp. 119–138, Jul. 2024, doi: 10.26439/interfaces2024.n19.7097.
- [13] G. Mutlu, "Comparative Evaluation of Machine and Deep Learning Models for PE File Malware Detection".
- [14] J. Banerjee, "Feature Selection and Transfer Learning in Network Data: Enhancing Anomaly Detection with Zero-Shot and Few-Shot Learning".
- [15] L. De Rose, G. Andresini, A. Appice, and D. Malerba, "OLIVANDER: a counterfactual-based method to generate adversarial Windows PE malware," *Data Min. Knowl. Discov.*, vol. 39, no. 5, p. 46, Sep. 2025, doi: 10.1007/s10618-025-01127-1.
- [16] A. Alotaibi, "Enhancing Malware Detection Using Machine Learning Models on Static Features," *Int. J. Adv. Comput. Sci. Appl.*, vol. 17, no. 2, 2026.
- [17] S. L. Shiva Darshan and C. D. Jaidhar, "Performance Evaluation of Filter-based Feature Selection Techniques in Classifying Portable Executable Files," *Procedia Comput. Sci.*, vol. 125, pp. 346–356, 2018, no: 10.1016/j.procs.2017.12.046.
- [18] E. Martins, R. Sant'Ana, J. R. B. Higuera, J. A. S. Montalvo, J. B. Higuera, and D. P. Castillo, "Semantic Malware Classification Using Artificial Intelligence Techniques," *Comput. Model. Eng. Sci.*, vol. 142, no. 3, pp. 3031–3067, 2025, doi: 10.32604/cmesci.2025.061080.
- [19] A. Manikandaraja, P. Aaby, and N. Pitropakis, "Rapidrift: Elementary Techniques to Improve Machine Learning-Based Malware Detection," *Computers*, vol. 12, no. 10, p. 195, Sep. 2023, doi: 10.3390/computers12100195.
- [20] Dr. W. Rose Varuna, Naveen S, Shruthi K, "Hybrid SVM-Random Forest Ensemble Superiority for Static Malware Detection: A Comparative Study," *Int. J. Recent Trends Multidiscip. Res.*, doi: <https://doi.org/10.59256/ijrtmr.20260602001>.
- [21] N. Vuran Sarı and M. Aci, "A hybrid CNN-GRU model with XAI-Driven interpretability using LIME and SHAP for static analysis in malware detection," *PeerJ Comput. Sci.*, vol. 11, p. e3258, Oct. 2025, doi: 10.7717/peerj-cs.3258.