Toward Robust Malware Detection: A Survey of Datasets, Techniques, and Practical Challenges

Trong-Thua Huynh
Information Security Technology Lab, Posts and Telecommunications Institute of Technology, Vietnam
thuaht@ptit.edu.vn

De-Thu Huynh
School of Computer Science & Engineering, The Saigon International University, Vietnam
huynhdethu@siu.edu.vn

Van-Quynh Trinh
Information Security Technology Lab, Posts and Telecommunications Institute of Technology, Vietnam
quynhtv@ptithcm.edu.vn (corresponding author)

Received: 12 January 2026 | Revised: 14 February 2026, 6 March 2026, 19 March 2026, and 21 March 2026 | Accepted: 23 March 2026

 DOI: https://doi.org/10.48084/etasr.17500

ABSTRACT

The increasing sophistication of malware has diminished the effectiveness of traditional signature-based detection. While Machine Learning (ML), Deep Learning (DL), and Large Language Models (LLMs) have improved malware classification, real-world systems continue to struggle with evasion attacks, temporal drift, and class imbalance. This study reviews the advancements in robust malware detection, focusing on benchmark datasets, detection methods, and operational constraints. Public datasets - EMBER2018, SOREL-20M, MalDICT, MOTIF, and EMBER2024 - are assessed for scale, label quality, and reproducibility. This paper contributes: (i) a Robust Malware Evaluation Protocol (RMEP) for consistent benchmarking under low False-Positive Rates (FPR) (≤ 0.1%) with temporal splits, and (ii) a Dataset-Task-Robustness (DTR) matrix for systematic comparison, offering practical guidance for reproducible malware-detection research. Future efforts should focus on broader multi-platform benchmark coverage, explicit analysis of robustness–accuracy trade-offs, interpretable language-assisted detection pipelines, and privacy-preserving collaborative learning frameworks.

Keywords-malware detection; robustness; benchmark datasets; Machine Learning (ML); Deep Learning (DL); Large Language Models (LLMs)

I. INTRODUCTION

Malware evolves rapidly through packing, obfuscation, and variant generation, limiting the effectiveness of traditional signature and heuristic-based detection. Machine Learning (ML) and Deep Learning (DL) approaches improve the detection accuracy by learning discriminative patterns from large corpora, with surveys demonstrating that they have been deployed across PC, mobile, IoT, and cloud environments [1]. Large Language Models (LLMs) have been explored for tasks, such as source-code understanding, system-call modeling, and malware analysis, highlighting new computational and interpretability trade-offs [2-4].

Despite recent advancements, employing consistent models remains a challenge. Specifically, generalization often fails when training data are imbalanced, outdated, or noisily labeled [2, 5]. Additionally, adversarial evasion attacks can degrade detection performance, particularly under strict low false-positive-rate requirements [6]. For these reasons, models must be periodically retrained using temporally structured splits to mitigate temporal concept drift [2]. High computational costs and limited interpretability continue to hinder the adoption of advanced detectors in resource-constrained operational contexts [7, 8].

Benchmark datasets have improved reproducibility. EMBER2018 provided 1.1M Windows Portable Executable (PE) files with curated static features [9], while SOREL-20M expanded this to 20M samples, including 10M disarmed malware PE files to support reproducible feature extraction and large-scale static analysis [10]. MalDICT contained roughly 5.5M samples with behavior tags and exploited vulnerabilities [11], and MOTIF supplied a family-labelled dataset for reproducible family classification [12]. EMBER2024 was expanded to 3.2M samples across multiple formats (Win32/64,

.NET, APK, ELF, and PDF), introducing a temporal split and an adversarial challenge set [2]. These corpora facilitate reproducible evaluation at very low False-Positive Rates (FPR) and support drift-aware benchmarking.

While previous surveys have focused on detection accuracy, robustness - encompassing adversarial resistance, temporal stability, explainability, and deployability - has received limited attention [6, 13]. For example, authors in [13], provided a broad coverage of ML-based detection and classification techniques, but they did not address robustness dimensions. This survey addresses this gap by presenting a focused, robustness-oriented synthesis. Specifically, it evaluates widely used benchmark datasets, such as DREBIN, EMBER2018, SOREL-20M, MalDICT, MOTIF, and EMBER2024, in terms of scale, labeling practices, and robustness properties [9-12, 14]; it compares signature-based, classical ML, DL, and emerging LLM-based malware detection methods [3, 6, 8, 13, 15]; and it synthesizes key challenges related to adversarial evasion, temporal concept drift, interpretability, and deployment constraints in operational settings [2, 5, 7, 8]. It further contributes a Robust Malware Evaluation Protocol (RMEP) and a Dataset-Task-Robustness (DTR) matrix for systematic benchmarking. Future directions emphasize multi-platform benchmarks, robustness-oriented training, explainable detection, and privacy-preserving collaboration.

II. DATASETS FOR MALWARE DETECTION

A. Importance of Benchmark Datasets

The quality and diversity of training data are critical to the effectiveness of malware detection systems. Publicly available datasets enable reproducible evaluation under adversarial evasion and temporal concept drift. However, researchers regularly face restricted access to high-quality corpora, as many stay proprietary or incomplete [16]. Class imbalance, obsolete malware families, and inconsistent antivirus-based labeling are common challenges in existing datasets [2, 5].

B. Legacy Datasets

Early benchmark datasets played an important role in foundational research but are increasingly inadequate for modern evaluation. For instance, DREBIN contains over 5,000 Android malware applications represented by static features like permissions and API calls [14], while malware binaries were visualized as gray-scale images to support image-based classification tasks [17]. These datasets were valuable for foundational research but increasingly inadequate for thorough, multi-platform evaluation.

C. Large-Scale Benchmarks

Several large-scale datasets designed for reproducible evaluation have been introduced:

- EMBER2018 provides 1.1M Windows PE files with curated static feature vectors and associated metadata [9]. By standardizing static feature extraction and providing pre-computed baseline models, EMBER2018 established a widely adopted evaluation baseline for static PE classifiers.
- SOREL-20M contains over 20M Windows PE samples represented by EMBER-style static features, providing approximately 10M disarmed malware PE files to support reproducible feature extraction at scale [10]. Its unprecedented scale enables statistically meaningful evaluation at very low FPR, which is critical for operational deployment where even small values translate to significant alert volumes.
- MalDICT releases approximately 5.5M malware samples annotated with behavioral properties, exploited vulnerabilities, platforms, and packers [11]. It is unique among public benchmarks in providing fine-grained multi-label behavioral annotations, enabling research on tasks beyond binary detection, such as vulnerability exploitation prediction and packer identification.
- MOTIF supplies a curated reference dataset with ground-truth malware family labels, enabling reproducible and comparable family-classification benchmarks [12].

The most recent large-scale benchmark is EMBER2024, which contains more than 3.2M samples spanning six formats: Win32, Win64, .NET, APK, ELF, and PDF [2]. It includes an evasive challenge set composed of malware samples that initially bypassed antivirus detection, enabling realistic adversarial evaluation. EMBER2024 further enforces an explicit temporal split (52-week training / 12-week test) to evaluate temporal concept drift, and introduces feature version 3, which represents all supported formats using a fixed 696-dimensional feature schema [2].

D. Dataset-Task-Robustness (DTR) Matrix

To systematically compare benchmark datasets, this survey introduces the DTR matrix, which maps each dataset against three dimensions: (1) dataset properties (scale, platforms, benign coverage), (2) supported evaluation tasks (detection, family classification, behavior labeling), and (3) robustness-relevant attributes (temporal splits, challenge sets, multi-label support). Table I presents the DTR matrix for the major benchmark datasets reviewed in this research.

TABLE I. DTR MATRIX OF MALWARE BENCHMARK DATASETS

Dataset	Scale	Platforms	Benign	Tasks	Challenge	Robustness notes / ref.
EMBER2018	1.1M	Windows PE	Yes	Detection	No	Standardized PE features [9]
SOREL-20M	20M	Windows PE	Yes	Detection + behavior	No	Low-FPR evaluation [10]
MalDICT	5.5M	Multi (malware)	No	Behavior/CVEs/Packers	No	Multi-label tagging [11]
MOTIF	100K+	Windows PE	Limited	Family classification	No	Family ground truth [12]
EMBER2024	3.2M	Win32/64, .NET, APK, ELF, PDF	Yes	Seven tasks (incl. behavior)	Yes	Temporal split, evasive samples [2]

E. Remaining Gaps and Robust Malware Evaluation Protocol (RMEP)

Despite substantial progress, multiple challenges persist. Class imbalance - often exhibiting Zipfian distributions at the family level - significantly reduces macro-averaged performance [2, 5]. Additionally, detection performance degrades under temporal concept drift when models are evaluated weeks or months after training [2]. Label noise from antivirus consensus and limited benign samples for non-Windows ecosystems further complicate ground-truth labels at scale.

An examination of most legacy datasets indicates several important limitations. Among them, EMBER2024 uniquely provides an explicit challenge set designed to evaluate robustness against evasive malware [2]. Regarding label reliability, antivirus-consensus labeling introduces noise because individual engines disagree on borderline samples, and labels may shift over time as engines update their signatures; EMBER2024 applies a majority-vote threshold over multiple engines, but a non-trivial fraction of samples remains ambiguous [2]. Regarding dataset aging, the temporal distribution of malware families shifts substantially within months. Older datasets, such as DREBIN (collected through 2012) and EMBER2018 (collected through 2017), may no longer accurately represent current threat landscapes, thereby limiting their value in evaluating drift robustness. Regarding real-world representativeness, public benchmarks often over-represent Windows PE files and under-represent fileless malware, document-based exploits, and mobile threats relative to operational SOC telemetry. EMBER2024 partially addresses multi-platform representativeness by including APK, ELF, and PDF formats; however, cross-platform benign coverage remains uneven [2].

To standardize these requirements, the RMEP was proposed in this study. Relative to the prior framework of [13], which provides broad coverage of ML-based detection and classification techniques, RMEP differs in three key respects: (1) it mandates evaluation at operationally relevant very low FPR ($\leq 0.1\%$) rather than aggregate ROC-AUC; (2) it requires separation of evasive challenge sets from standard test sets where available, and (3) it provides an explicit practitioner-oriented deployment checklist covering latency, update cadence, and privacy constraints, extending beyond the algorithmic scope of [13].

Accordingly, this work consolidates prior robustness-oriented evaluation practices into a compact and reproducible framework. The RMEP comprises five components: (1) temporal (non-random) data splits to expose drift; (2) ROC evaluation at very low FPR ($\leq 0.1\%$), aligned with operational alert budgets; (3) PR-AUC reporting under class imbalance to capture performance across the full precision-recall trade-off; (4) separate evaluation on evasive or packed challenge subsets whenever available; and (5) a practitioner-oriented checklist covering latency and throughput targets, false-positive budgets, update cadence, explainability requirements, and privacy or federation constraints. RMEP further proposes disclosing feature schemas and versions (e.g., EMBER feature version 3) to enable cross-study comparability.

To illustrate RMEP's application, consider applying it to EMBER2024 [2]. Component (1) is satisfied by its built-in 52-week training and 12-week test temporal split. Component (2) is supported by EMBER2024's evaluation scripts, which report a True Positive Rate (TPR) at 1% FPR and 0.1% FPR; published baseline LightGBM classifiers achieve approximately 94% TPR at 1% FPR on the standard test set [2]. Component (3) is addressed by reporting PR-AUC separately for each file-type partition (Win32, Win64, .NET, APK, ELF, PDF) to capture per-class imbalance effects. Component (4) is directly enabled by the EMBER2024 challenge set, which isolates initially antivirus-bypassing samples; the same LightGBM baseline achieves substantially lower TPR on this challenge set, demonstrating the gap between standard and adversarial evaluation. Component (5) would require a practitioner to document target inference latency (e.g., <10 ms per sample), acceptable false-positive budget (e.g., 1 alert per 1,000 benign files), and the intended update cadence. This case study confirms that EMBER2024 is the only current public benchmark that satisfies all five RMEP components out of the box.

III. TECHNIQUES FOR MALWARE DETECTION

A. Signature and Heuristic-Based Detection

Signature-based detection remains the foundational layer of modern antivirus engines and Security Operations Center (SOC) workflows [6, 13]. This approach matches known byte patterns, cryptographic hashes, or structured rules against a database of previously identified threats, enabling rapid and precise identification of known malware with minimal computational cost. Heuristic-based methods extend this paradigm by applying predefined behavioral rules, such as detecting suspicious API call patterns, registry modifications, or network activity, to identify previously unseen variants [18]. While these methods offer high-confidence, low-latency defense for known threats and remain the first line of defense, they cannot generalize to novel malware families or obfuscated variants, motivating the adoption of ML and DL approaches as complementary detection layers [6, 13].

B. Classical Machine Learning Approaches

Classical malware detection approaches have widely employed conventional classifiers such as Random Forests (RF), Support Vector Machines (SVM), and k-Nearest Neighbors (kNN) [6, 13]. These models rely on manually engineered features, including byte *n*-grams, opcodes, and API call frequencies. Because ensemble methods improve generalization by aggregating diverse decision boundaries, they often outperform single classifiers, particularly RF and gradient-boosted trees [13]. These approaches are effective for real-time applications due to their low computational cost and interpretability. However, they cannot adapt to novel threats because they depend on manual feature design.

C. Deep Learning Models

DL models can automatically learn discriminative representations from raw or lightly processed inputs, reducing

the need for manual feature engineering [13, 15]. Convolutional Neural Networks (CNNs) are commonly applied by transforming byte sequences or binaries into image-like representations, where they have demonstrated strong performance in malware detection tasks [17]. Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) architectures model sequential dependencies in API calls, opcodes, or execution traces, enabling temporal pattern learning [15]. Hybrid designs that combine static and dynamic information, such as byte-level features and API call sequences, can improve robustness by leveraging complementary signals [15]. DL models frequently outperform classical ML approaches in detection accuracy; however, they require large and representative training datasets as well as being vulnerable to adversarial manipulation and distribution shift [13, 15].

D. Large Language Model (LLM)-Based Approaches

LLMs, pretrained on large-scale code or execution-related corpora, can be adapted for malware detection and analysis tasks [3, 4, 8]. Two primary adaptation paradigms have emerged: fine-tuning, in which pre-trained models (e.g., CodeBERT, RoBERTa) are further trained on labeled malware samples for classification or behavioral prediction, and prompt-based approaches, in which LLMs such as GPT-4 are queried with structured prompts describing binary characteristics or system call sequences, leveraging in-context learning without weight updates [3, 8]. Pretrained LLMs have been explored for modeling system call sequences and runtime traces [4]. LLMs have also been proposed as tools for assisting human analysts through summarization and interpretation of detection outputs, such as generating natural-language threat summaries [8]. Compared to DL models, LLM-based approaches exhibit stronger contextual reasoning and transfer learning capabilities but at substantially greater computational cost. Typical DL classifiers (CNN, LSTM) process raw bytes or feature vectors with millisecond inference latency, whereas decoder-based LLMs require orders-of-magnitude more compute per query, making them impractical for real-time endpoint detection [7]. They further introduce security-specific risks, including hallucination of plausible but incorrect malware family attributions, susceptibility to prompt-injection attacks, and limited robustness under adversarial input perturbations---all of which are particularly problematic in high-stakes security decisions [8]. Operationally, deploying LLMs needs explicit compute and latency budgets, and their outputs typically require explanation surrogates (e.g., attribution) to satisfy SOC requirements.

E. Comparative Overview

The strengths and weaknesses of signature-based, classical ML, DL, and LLM-based approaches for robust malware detection are comparatively analyzed in Table II.

TABLE II. COMPARATIVE OVERVIEW OF MALWARE DETECTION TECHNIQUES

Category	Key methods	Strengths	Weaknesses	Ref.
Signature/ heuristic	Pattern matching, rule-based	Fast, precise for known threats, low cost	Cannot detect novel malware, manual rule updates	[6, 13, 18]
Classical ML	RF, SVM, kNN, ensembles	Low cost, interpretable, reproducible	Requires handcrafted features, limited adaptability	[6, 13]
DL	CNN, RNN/LSTM, hybrids	Automated feature learning, strong accuracy	High data requirements, adversarial vulnerability	[13, 15, 19]
LLM-based	BERT, GPT, LLaMA variants	Transfer learning, code/system analysis	High cost, limited explainability, robustness untested	[3, 7, 8]

Table III presents a taxonomy of malware detection techniques organized by input features, representative methods, strengths, and limitations. This categorization supports engineering decision-making, as the choice of input features strongly influences model suitability, resource requirements, and deployment cost.

TABLE III. TAXONOMY OF MALWARE DETECTION TECHNIQUES

Category	Input features	Methods	Strengths	Limitations	Ref.
Signature/ heuristic	Byte patterns, rules, hashes	Pattern matching, YARA rules	Fast, precise, low cost	No novel malware detection	[6, 13, 18]
Classical ML	Byte n-grams, opcodes, API freq.	SVM, RF, XGBoost, LightGBM	Low cost, interpretable	Feature engineering, limited adaptability	[6, 13]
DL (CNN)	Byte sequences, malware images	1D/2D CNN	Auto feature learning, strong accuracy	Adversarial vulnerability, high data demand	[15, 17], 20]
DL (RNN)	Opcode/API sequences	RNN, LSTM, GRU	Captures sequential patterns	Training cost, resource-intensive	[15]
Hybrid	Static + Dynamic features	Multimodal deep networks	Complementary views, robustness	Complexity, deployment cost	[15]
LLM-based	Code corpora, syscall traces	BERT, GPT, LLaMA variants	Transfer learning, contextual analysis	High overhead, limited explainability	[3, 7, 8]

F. Engineering Implications

In production systems, signature and heuristic-based methods continue to be the primary line of defense due to their speed and precision for known threats. Classical ML stays relevant for constructing lightweight and interpretable detection pipelines.

Representative published results illustrate the current performance landscape:

- Classical ML: LightGBM and XGBoost classifiers trained on EMBER-style static features achieved ROC-AUC values above 0.999 and TPR above 94% at a 1% FPR. However, this performance dropped sharply (55-65%) when evaluated on evasive challenge sets [2].

- DL Models: Architectures like CNNs (on byte sequences) and LSTM (on API calls) frequently achieved high detection rates on clean test sets (ROC-AUC 0.98-0.999) but were similarly vulnerable under adversarial evaluation [13, 15].
- LLMs: While LLMs have been evaluated primarily on source-code analysis tasks with reported accuracy above 90% for binary malicious/benign classification, they have not yet been standardized under RMEP-aligned protocols at low FPR [3, 7].

Although DL models dominate contemporary research benchmarks, they require explicit hardening against adversarial evasion and should be trained and evaluated using temporally structured datasets to mitigate temporal concept drift. LLM-based approaches are promising for tasks integrating malware detection with code understanding, but their high computational cost and limited interpretability restrict widespread operational adoption. Consequently, operational evaluation should emphasize metrics aligned with real-world alerting requirements, including ROC performance at very low FPR ($\leq 0.1\%$) and PR-AUC under class imbalance, consistent with the RMEP framework described above.

IV. CHALLENGES IN ROBUST MALWARE DETECTION

A. Adversarial Evasion

This study assumes a limited-knowledge adversary that should preserve malicious functionality while applying packing, obfuscation, or small feature-level perturbations to evade detection. The defender operates under strict budgets, aiming for high TPR at very low FPR ($\leq 0.1\%$). Even modest modifications to file structure or byte patterns can mislead DL models without altering program semantics, whereas hand-crafted ML features persist in their susceptibility to manipulation [18]. EMBER2024 partially addresses this challenge by providing an explicit evasive test set consisting of malware samples that initially bypassed antivirus detection, enabling controlled evaluation of adversarial robustness [2]. Representative adversarial attacks in the malware domain include gradient-based perturbation methods (e.g., FGSM, PGD) adapted to discrete binary domains [6], content-injection attacks that append benign bytes to malicious binaries without altering executable semantics [13], and problem-space attacks that synthesize fully functional adversarial PE files [21]. Libraries, such as the IBM Adversarial Robustness Toolbox (ART) and Foolbox, provide reference implementations for evaluating classifier robustness against these attack families. Defenses include adversarial training with augmented perturbation samples, ensemble-based randomized smoothing, feature-space input transformations (e.g., byte squeezing, header normalization), as well as certified defenses that provide provable robustness bounds. Defense continues to focus on adversarial training, demonstrating improved robustness on challenge sets while incurring moderate accuracy trade-offs on clean samples [6, 22].

B. Temporal Concept Drift

Models trained on previous data rapidly degrade on more recent samples as the malware landscape undergoes rapid evolution with the introduction of new families and techniques [2]. Authors in [23] proposed the Transcend framework, which uses conformal evaluators to flag aging classifier decisions in real time, enabling practitioners to detect when a deployed model requires retraining. Similarly, the TESSERACT evaluation methodology was introduced in [24], which enforces temporal train-test splits to prevent time-travel bias while providing realistic performance estimates of classifiers under distribution shift. EMBER2024 explicitly exposes temporal concept drift by organizing training data over 52 weeks and evaluating on a subsequent 12-week test window [2]. Maintaining performance under drift typically requires periodic retraining and evaluation protocols that explicitly account for temporal effects. Mitigation strategies studied in the literature include continual learning with selective sample replay, online learning frameworks that adapt decision boundaries incrementally, scheduled model updates combined with incremental retraining, and semi-supervised relabeling using newly collected samples to adapt to emerging threats.

C. Explainability and Trustworthiness

Model interpretability is critical for adoption in security operations, as analysts require explicit explanations to prioritize alerts and reduce false positives [8]. Techniques such as SHapley Additive exPlanations (SHAP) assign per-feature importance scores [25] revealing that import-table characteristics, section entropy, and header fields are primary discriminators [19], while Local Interpretable Model-agnostic Explanations (LIME) provide instance-level justifications for classification decisions [26]. For DL models, Grad-CAM highlights binary regions driving predictions [20], and LEMNA supports interpretable explanations over API-call and opcode sequences [8]. However, explainability methods are not widely deployed in production systems. Beyond interpretability, trustworthy detection systems should also demonstrate accountability and robustness to distributional change [8]. In operational settings, SHAP-based triage of import tables, entropy values, and PE header fields can assist analyst workflows, while API-call attributions can surface behavioral indicators such as process injection or credential access patterns.

D. Deployment Constraints

Modern deployments operate under strict limits on latency, throughput, memory, and updating cadence, especially at the edge. Such constraints make heavy models impractical without pruning, distillation, or classical-ML fallbacks, and require explicit false-positive budgets ($\leq 0.1\%$) with drift monitoring. These demands motivate lightweight detection pipelines and explicit cost‑accuracy trade-offs in security-critical systems [7].

E. Engineering Implications

Collectively, these challenges highlight the need for evaluation methodologies that prioritize robustness rather than aggregate accuracy alone. Effective studies report performance at very low FPR, evaluate models under adversarial and temporal splits, and provide outputs that support analyst decision-making. At extremely low FPR ($\leq 0.1\%$), sufficiently

large benign corpora are required, and reporting the scale of benign data and associated alerting costs improves transparency. From a deployment perspective, robustness further depends on lightweight models, timely update mechanisms, and privacy-aware architectures suitable for distributed environments.

V. FUTURE DIRECTIONS

A. Standardized Multi-Platform Benchmarks

Large-scale datasets have improved reproducibility; however, most public benchmarks remain concentrated on Windows PE files [2, 10]. EMBER2024 advances the field by supporting multiple file formats within a unified evaluation framework [2]. Future benchmarks would benefit from broader coverage of Linux, macOS, IoT, and industrial environments, provided that data quality and labeling consistency are maintained.

B. Balancing Robustness and Accuracy

Most studies focus on detection accuracy, whereas robustness to adversarial evasion and temporal concept drift has received comparatively less emphasis [6, 13]. Adversarial training and regularization can improve robustness but may reduce clean-data accuracy, while drift-aware approaches, such as continual learning or domain adaptation, mitigate temporal degradation at the cost of increased system complexity [8].

C. Explainable Language-Assisted Detection

LLMs seem promising for code analysis, system call modeling, and analyst-facing report generation [3, 4, 8]. However, their high computational cost and limited interpretability currently constrain their direct use in real-time malware detection. A promising direction lies in hybrid architectures that leverage LLMs for contextual analysis while retaining lightweight classifiers for time-critical detection [19, 20].

D. Lightweight and Privacy-Preserving Frameworks

Emerging IoT and edge environments generate the need for compact and efficient malware detection models [7, 18].

TABLE IV. RESEARCH GAPS IN ROBUST MALWARE DETECTION

Area	Existing practice	Gaps identified	Future research	Ref.
Datasets	Large PE-focused datasets	Limited non-PE coverage, few challenge sets	Multi-platform benchmarks with temporal splits	[2, 10]
Evaluation	Overall accuracy or ROC-AUC	Rarely report PR-AUC or metrics at ≤0.1% FPR	Robustness-centered protocols (RMEP)	[2, 21]
Techniques	DL dominates, LLMs emerging	Adversarial training underexplored, LLM robustness untested	Robustness-oriented training, hybrid models	[6, 13, 15]
Explainability	Feature attribution explored	Limited operational integration	Practical SOC-facing interpretable outputs	[8, 25-26]
Deployment	Benchmark-focused	Lightweight/IoT constraints underexplored	Compact models, federated frameworks	[7, 18]

Techniques, such as model reduction, knowledge distillation, and lightweight architectural design, have not been examined yet. Privacy considerations further highlight the importance of collaborative learning frameworks. While federated learning and secure aggregation have shown potential, their application to malware detection is at an early stage [2]. Despite progress, substantial challenges remain, as summarized in Table IV.

VI. CONCLUSION

Machine Learning (ML), Deep Learning (DL), and more recent language-based models have substantially advanced malware detection capabilities. However, critical operational challenges persist, including adversarial evasion, temporal concept drift, class imbalance, and limited interpretability. This study examined robustness-related aspects of widely used benchmark datasets, such as EMBER2018, SOREL-20M, MalDICT, MOTIF, and EMBER2024, highlighting their differences in scale, labeling practices, platform coverage, temporal structure, and support for evasive evaluation. Furthermore, signature-based, classical ML, DL, and LLM-based detection approaches were compared regarding evasion resistance, drift-aware evaluation, transparency, and deployment efficiency.

Two structured evaluation instruments were introduced in this study: the Robust Malware Evaluation Protocol (RMEP), a five-component framework for consistent robustness benchmarking, and the Dataset-Task-Robustness (DTR) matrix for systematic dataset comparison. Rather than proposing new detection methods, this work synthesized existing datasets, techniques, and evaluation practices to position EMBER2024 within a broader robustness-oriented evaluation context [2].

Regarding future steps, malware detection research would benefit from moving beyond accuracy-centric evaluation toward more holistic robustness assessment. Key directions include broader multi-platform benchmark coverage, explicit analysis of robustness–accuracy trade-offs, interpretable language-assisted detection pipelines, and privacy-preserving collaborative learning frameworks. Progress along these dimensions is essential for developing detection systems that are not only accurate, but also resilient, transparent, and aligned with real-world Security Operations Center (SOC) constraints.

DECLARATION OF COMPETING INTERESTS

The authors declare no competing interests.

ACKNOWLEDGMENT

This research is supported by Posts and Telecommunications Institute of Technology (PTIT).

DATA AVAILABILITY

The datasets used in this study (e.g., EMBER2018, SOREL-20M, MalDICT, MOTIF, and EMBER2024) are publicly available and can be accessed through the corresponding references cited in the manuscript [2, 9-12].

AI USE AND DECLARATION OF GENERATIVE AI USE

During the preparation of this work, the authors used generative AI tools (ChatGPT) for language refinement. After using these tools, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

REFERENCES

[1] J. Ferdous, R. Islam, A. Mahboubi, and M. Z. Islam, "A Survey on ML Techniques for Multi-Platform Malware Detection: Securing PC, Mobile Devices, IoT, and Cloud Environments," *Sensors*, vol. 25, no. 4, Feb. 2025, Art. no. 1153, https://doi.org/10.3390/s25041153.

[2] R. J. Joyce *et al.*, "EMBER2024 - A Benchmark Dataset for Holistic Evaluation of Malware Classifiers," in *31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, Toronto, Canada, Aug. 2025, https://doi.org/10.1145/3711896.3737431.

[3] S. Altamimi and M. Ababneh, "Detecting Spam and Malware Using BERT and LLMs," in *2024 25th International Arab Conference on Information Technology (ACIT)*, Zarqa, Jordan, Dec. 2024, https://doi.org/10.1109/ACIT62805.2024.10877264.

[4] P. M. Sánchez Sánchez, A. H. Celdrán, G. Bovet, and G. M. Pérez, "Transfer Learning in Pre-Trained Large Language Models for Malware Detection Based on System Calls," in *MILCOM 2024 - 2024 IEEE Military Communications Conference (MILCOM)*, Washington, DC, USA, Oct. 2024, https://doi.org/10.1109/MILCOM61039.2024.10773857.

[5] H. Almajed, A. Alsaqer, and M. Frikha, "Imbalance Datasets in Malware Detection: A Review of Current Solutions and Future Directions.," *International Journal of Advanced Computer Science & Applications*, vol. 16, no. 1, 2025, Art. no. 1323, https://doi.org/10.14569/ijacsa.2025.01601126.

[6] M. G. Gaber, M. Ahmed, and H. Janicke, "Malware Detection with Artificial Intelligence: A Systematic Literature Review," *ACM Computing Surveys*, vol. 56, no. 6, pp. 1–33, Jan. 2024, https://doi.org/10.1145/3638552.

[7] C. Rondanini, B. Carminati, E. Ferrari, A. Kundu, and A. Gaudiano, "Malware Detection at the Edge with Lightweight LLMs: A Performance Evaluation," *ACM Transactions on Internet Technology*, vol. 26, no. 1, Jan. 2026, Art. no. 15, https://doi.org/10.1145/3769681.

[8] J. Al-Karaki, M. A.-Z. Khan, and M. Omar, "Exploring LLMs for Malware Detection: Review, Framework Design, and Countermeasure Approaches." arXiv, Sept. 11, 2024, https://doi.org/10.48550/arXiv.2409.07587.

[9] H. S. Anderson and P. Roth, "EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models. arXiv, Apr. 2018, https://doi.org/10.48550/arXiv.1804.04637.

[10] R. Harang and E. M. Rudd, "SOREL-20M: A Large Scale Benchmark Dataset for Malicious PE Detection." arXiv, Dec. 14, 2020, https://doi.org/10.48550/arXiv.2012.07634.

[11] R. J. Joyce, E. Raff, C. Nicholas, and J. Holt, "MalDICT: Benchmark Datasets on Malware Behaviors, Platforms, Exploitation, and Packers." arXiv, Oct. 18, 2023, https://doi.org/10.48550/arXiv.2310.11706.

[12] R. J. Joyce, D. Amlani, C. Nicholas, and E. Raff, "MOTIF: A Malware Reference Dataset with Ground Truth Family Labels," *Computers & Security*, vol. 124, Jan. 2023, Art. no. 102921, https://doi.org/10.1016/j.cose.2022.102921.

[13] 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, Mar. 2020, Art. no. 102526, https://doi.org/10.1016/j.jnca.2019.102526.

[14] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck, and C. Siemens, "Drebin: Effective and explainable detection of android malware in your pocket.," in *Network and Distributed System Security Symposium (NDSS 2014)*, San Diego, CA, USA, Feb. 2014, https://doi.org/10.14722/ndss.2014.23247.

[15] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, and S. Venkatraman, "Robust Intelligent Malware Detection Using Deep Learning," *IEEE Access*, vol. 7, pp. 46717–46738, Apr. 2019, https://doi.org/10.1109/ACCESS.2019.2906934.

[16] M. Ganesamoorthi, K. Subramanian, and B. D, "A Comprehensive Review on Machine Learning and Deep Learning Based Malware Detection Methods," in *2024 International Conference on Emerging Research in Computational Science (ICERCS)*, Coimbatore, India, Dec. 2024, https://doi.org/10.1109/ICERCS63125.2024.10894949.

[17] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, "Malware images: visualization and automatic classification," in *Proceedings of the 8th International Symposium on Visualization for Cyber Security*, Pittsburgh, PA, USA, July 2011, https://doi.org/10.1145/2016904.2016908.

[18] A. Al-Marghilani, "Comprehensive Analysis of IoT Malware Evasion Techniques," *Engineering, Technology & Applied Science Research*, vol. 11, no. 4, pp. 7495–7500, Aug. 2021, https://doi.org/10.48084/etasr.4296.

[19] G. M. and S. C. Sethuraman, "A comprehensive survey on deep learning based malware detection techniques," *Computer Science Review*, vol. 47, Feb. 2023, Art. no. 100529, https://doi.org/10.1016/j.cosrev.2022.100529.

[20] X. Liu, Y. Lin, H. Li, and J. Zhang, "A novel method for malware detection on ML-based visualization technique," *Computers & Security*, vol. 89, Feb. 2020, Art. no. 101682, https://doi.org/10.1016/j.cose.2019.101682.

[21] F. Pierazzi, F. Pendlebury, J. Cortellazzi, and L. Cavallaro, "Intriguing Properties of Adversarial ML Attacks in the Problem Space," in *2020 IEEE Symposium on Security and Privacy (SP)*, San Francisco, CA, USA, May 2020, https://doi.org/10.1109/SP40000.2020.00073.

[22] D. Arp *et al.*, "Dos and Don'ts of Machine Learning in Computer Security," in *31st USENIX Security Symposium (USENIX Security 22)*, Boston, MA, USA, 2022.

[23] R. Jordaney, K. Sharad, S. K. Dash, Z. Wang, D. Papini, I. Nouretdinov, and L. Cavallaro, "Transcend: Detecting Concept Drift in Malware Classification Models," in *26th USENIX Security Symposium (USENIX Security 17)*, Vancouver, BC, Canada, 2017.

[24] F. Pendlebury, F. Pierazzi, R. Jordaney, J. Kinder, and L. Cavallaro, "TESSERACT: Eliminating Experimental Bias in Malware Classification across Space and Time," in *28th USENIX Security Symposium (USENIX Security 19)*, Santa Clara, CA, USA, 2019.

[25] S. M. Lundberg and S.-I. Lee, "A Unified Approach to Interpreting Model Predictions," in *Advances in Neural Information Processing Systems*, 2017, vol. 30, pp. 4756–4774.

[26] M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why Should I Trust You?': Explaining the Predictions of Any Classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco, CA, USA, Aug. 2016, https://doi.org/10.1145/2939672.2939778.