Journal of Al-Farabi for Engineering Sciences (JFES)
Vol.5, No. 1, 2026, **pp**. 66–92
ISSN: 2617-5517 | **DOI**: https://doi.org/10.59746/asgg6t42

EMBER-Based Static Malware Detection: A Critical Review of Accuracy, Explainability, and Temporal Robustness Trade-offs

Ahmed M. Redha Abdulsattar[1], Riyadh Rahef Nuiaa Alogaili[2], Ahmed Raad Al-Sudani[1], Selvakumar Manickam[3]
[1]Software Department, College of Computer Science and Information Technology, Wasit University, Wasit, Al-Kut, 52001, Iraq
stdm.aabdulsattar@uowasit.edu.iq ; araad@uowasit.edu.iq
[2]Cybersecurity Department, College of Computer Science and Information Technology, Wasit University, Wasit, Al-Kut, 52001, Iraq
riyadh@uowasit.edu.iq
[3]Cybersecurity Research Centre, Universiti Sains Malaysia, 11800 USM, Penang, Malaysia
selva@usm.my
Received 09/01/2026, Revised 15/03/2026, Accepted 30/04/2026

Abstract: Static analysis-based malware detection of Portable Executable (PE) files has evolved remarkably since the release of the EMBER dataset in 2018. Yet evaluation methodology and model explainability continue to suffer from critical challenges that limit real-world implementation. This literature review explores five thematic clusters: Traditional Machine Learning, Deep Learning, Ensemble and Hybrid Architectures, Explainable AI (XAI), and Zero-day Detection with Concept Drift. The review analyzes 27 primary studies and 15 supporting references for a total of 42 studies published between 2018 and early 2026. The review shows that gradient boosted decision tree steadily offers better baseline performance. In comparison, ensemble and hybrid architectures show the highest accuracy overall. That being said, this comes with the cost of reduced explainability and an increase in computational overhead. Deep Learning methods make the performance gap thinner but bring up transparency and resource concerns. And the emerging Large Language Model (LLM)-based approaches remaining premature and unverified. Across all of the five clusters, six intersecting gaps are identified, the most notable being the near-universal dependence on random instead of temporal train/test splits. Other gaps include the lack of sufficient false positive rate reporting at operational thresholds, and the consistent separation between explainability and detection performance. Critically, no reviewed study achieved a successful integration of ensemble level accuracy, embedded explainability, and temporal oriented evaluation within a single framework. It's a gap that this review specifically recognizes and highlights as the most crucial priority of the research in this field. The gaps explored can be addressed with the seven future research directions presented later in this review. The most critical one of them is the incorporation of ensemble accuracy, explainability and temporal evaluation in a unified framework. This is a combination that no reviewed study has achieved yet.

Keywords: static malware detection, EMBER dataset, ensemble learning, explainable AI, windows malware.

1. Introduction

Malware threats are continuously growing in complexity and capability, industries recently report the notice of hundreds of millions of new malicious binaries yearly. This growth is visible in cumulative statistics of malware detection, as shown in Figure 1 by AV-TEST Institute [1], records show that the total number of identified Windows malware samples has increased from nearly 400 million in 2016 to nearly 1.2 billion in 2026. Signature-based detection techniques, which is the traditional way to detect malware, relied heavily on pattern matching against a database of previously detected threats, hence struggling to keep up with the speed at which new undetected and adversarial variants are released [2]. This ever growing gap between the speed of malware production and signature generation capacity has created a necessity for newer, more reliable methods to detect malware, methods that are more capable of detecting malware it didn't see before, methods that rely on data, instead of mere signatures, to identify threats [3], [4].

Static analysis, which is the practice of obtaining distinctive features from binary files without running them, offers an especially attractive mix of scalability and speed to address the need of aforementioned newer detection methods. Due to the needlessness of a sandboxed runtime environment, static methods have the ability to deployed at high capacity in endpoint protection pipelines. On Windows, the dominant type of executable files are Portable Executables (PE) serve as the primary surface for attacks and targeted malware. This results in making static PE feature engineering emerge as a highly active research area within the overall scope of malware detection methods [5], [6]. An important turning point in the field was the announcement of Endgame

 ISSN: 2617-5517 | **DOI**: https://doi.org/10.59746/asgg6t42

Malware Benchmark for Research (EMBER, now rebranded as Elastic Malware Benchmark for Empowering Researchers after Endgame's acquisition by Elastic [7]) dataset by Anderson and Roth in 2018 [8]. EMBER presented the community of researchers with a large-scale, open access benchmark comprising of about one million samples carrying a label of either benign or malicious or unlabeled. Each of those samples is represented by a high-dimensional parameter array extracted from eight distinct feature groups.

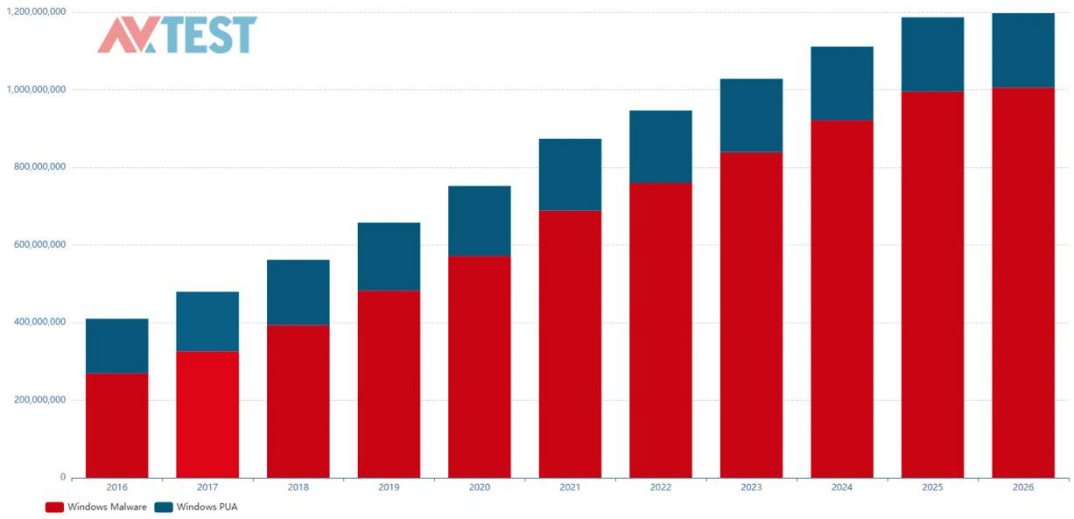

Figure 1 Total Amount of Malware and PUA under Windows (2016-2026)

The baseline model LightGBM it arrives with set a robust benchmark point of 0.9994 AUC (Area Under Curve) and quickly, EMBER became the standard for static malware detectors evaluation, this opened the opportunity for reproducible, joint-study comparisons among different continuously growing research. Recently, EMBER2024 was released [7] as an update to further develop this evaluation foundation by introducing a comprehensive, multi-family testing methodology that overcomes multiple limitations of the original version. This review examines the literature that utilizes EMBER or adopts EMBER-compatible static PE features for machine learning and deep learning-based threat detection. Three selection requirements dictate if a study is qualified to be included: the literature must (a) use EMBER dataset or a parallel static PE feature framework covering similar feature categories (byte, statistics, header etc.), (b) implement machine learning or deep learning for the identification or classification, and (c) be published in peer-reviewed venues or publicly available as citable technical reports (such as arXiv, SSRN preprints etc.). Preprints referenced from arXiv or SSRN were included selectively and only in cases where they show a sole availability of the work addressing a specific thematic criterion. Their provisional status is also noted openly at each point of citation. More expansive surveys and taxonomies were also cited to provide context for the study, but not treated as primary studies. The resulting work is structured into five thematic groups: traditional machine learning, deep learning, ensemble and hybrid architectures, explainable AI, and zero-day detection with concept drift. For each group, the methods, results and remaining gaps are synthesized before laying out overall insights, that apply across all groups, and future directions.

Despite the increase of studies conducted in this research field, the existing reviewed literature shows a critical set of persistent methodological constraints across. The large majority of studies test models against randomly selected train/test splits from EMBER, this leads to producing

artificially optimistic performance numbers, as discussed by Galen and Steele [9]. Such practices results in noticeable performance degradations over time under real world deployments. Beyond evaluation methodology, interpretability methods are applied almost across all the studies universally as a post-hoc addition to trained models, and none of the studies validated their applicability through a systematic testing procedure that ensure the role of security professionals. Most critically, no study in the existing corpus has succeeded in integrating a high accuracy ensemble architecture with embedded explainability and temporal oriented rigorous evaluation all in a single framework on EMBER's benchmark. This review addresses the gaps directly through outlining an ordered synthesis of 27 primary studies that are EMBER-based, detecting six intersecting shortcomings, and offering seven solid directions for the research to guide the field toward producible static malware detection approaches.

1.1 Review Contributions

The main contributions of this review are as listed:

1. An organized synthesis of 27 primary studies that are EMBER-based, across five theme-based clusters (traditional ML, Deep Learning, Ensemble and Hybrid Architectures, XAI, and concept drift / zero-day detection).
2. An intersecting recognition of six research gaps that go beyond individual clusters, that includes near universal use of random train/test splits instead of temporal oriented selection of splits. In addition to the consistent division between detection performance and models interpretability.
3. An overall comparison table (Table 6) evaluating 27 primary studies against four dimensions: Temporal Evaluation, Reporting FPR, Use of XAI, and Adversarial Robustness.
4. Seven solid future research directions, each with clear measurable criteria to allow for valid progress.
5. Recognition of the most critical unaddressed gap in the field which is that no existing study combines high accuracy ensemble architectures with integrated explainability and temporal evaluation on the EMBER benchmark.

The rest of this paper is structured as follows. Section II introduces EMBER dataset and places static analysis among the larger context of malware detection. Section III to VII review the five major research groups, one at a time: traditional machine learning, deep learning, ensemble and hybrid models, explainable AI, and zero-day detection under concept drift. Section VIII summarizes the six critical emerging gaps across the full corpus. Section IX presents future research directions based on those gaps, alongside a clear and quantifiable criteria for measuring the progress. Finally, Section X concludes with a summary of the main findings, discussing their applied significance and reflecting the field's future direction.

1.2 Review Methodology

1.2.1 Search Strategy

An organized literature search was done in four primary digital libraries: Google Scholar, Scopus, IEEE Xplore, and ACM Digital Library. Primarily, the search string used contained the dataset name and the detection approach: {"EMBER" AND "malware detection"}, "static malware detection using EMBER", and "EMBER malware detection". Following up with a secondary search to expand the thematic coverage using the terms: {"explainable AI" AND

 ISSN: 2617-5517 | **DOI**: https://doi.org/10.59746/asgg6t42

"malware"}, "concept drift malware detection", and "adversarial malware". The search targeted studies published between 2018 (the year which EMBER was originally released) through early 2026. Plus, only English-language publications were considered.

1.2.2 Inclusion and Exclusion Criteria

Papers were included only if they match all three requirements: (a) Utilize EMBER dataset or an EMBER compatible static PE feature framework that cover comparable feature types, (b) implement machine learning or deep learning for malware detection or classification, (c) published in venues where submissions undergo peer review or where they are available for citing as technical reports via known preprint platforms such as arXiv or SSRN.

A paper would be excluded if it: (i) exclusively targets dynamic malware detection or network analysis that doesn't employ static PE features, (ii) is not published in English, (iii) found through venues with no trusted peer review process, (iv) was a surveys or taxonomy that is broad that do not present original experimental results (such studies were kept for contextual reference but not reviewed as one of the primary studies).

1.2.3 Screening Process

Initially, the conducted search led to the collection of around 120 candidate papers. After de-duplication and removing papers that were clearly outside our scope based on the title and abstract content, 65 eligible papers were selected. Further full text level checks resulted in the removal of 23 more studies due to not meeting the inclusion criteria specified. This makes it 42 papers that were used to form the foundation of this review. 27 papers out of these were treated as primary studies that provided original experiments and evaluations on EMBER or EMBER-compatible features, and the remaining 15 serve as context support references. Figure 2 shows the complete screening process which followed the flow structure of PRISMA 2020.

The review covers studies across both the EMBER 2018 and EMBER 2024 eras. EMBER 2024 [8] introduced a substantially different evaluation paradigm, consisting of a rolling 52-week training window with a 12-week test phase and a challenge split designed to expose models to severe distributional shifts, and studies from this era are examined with reference to these updated standards for evaluation where applicable.

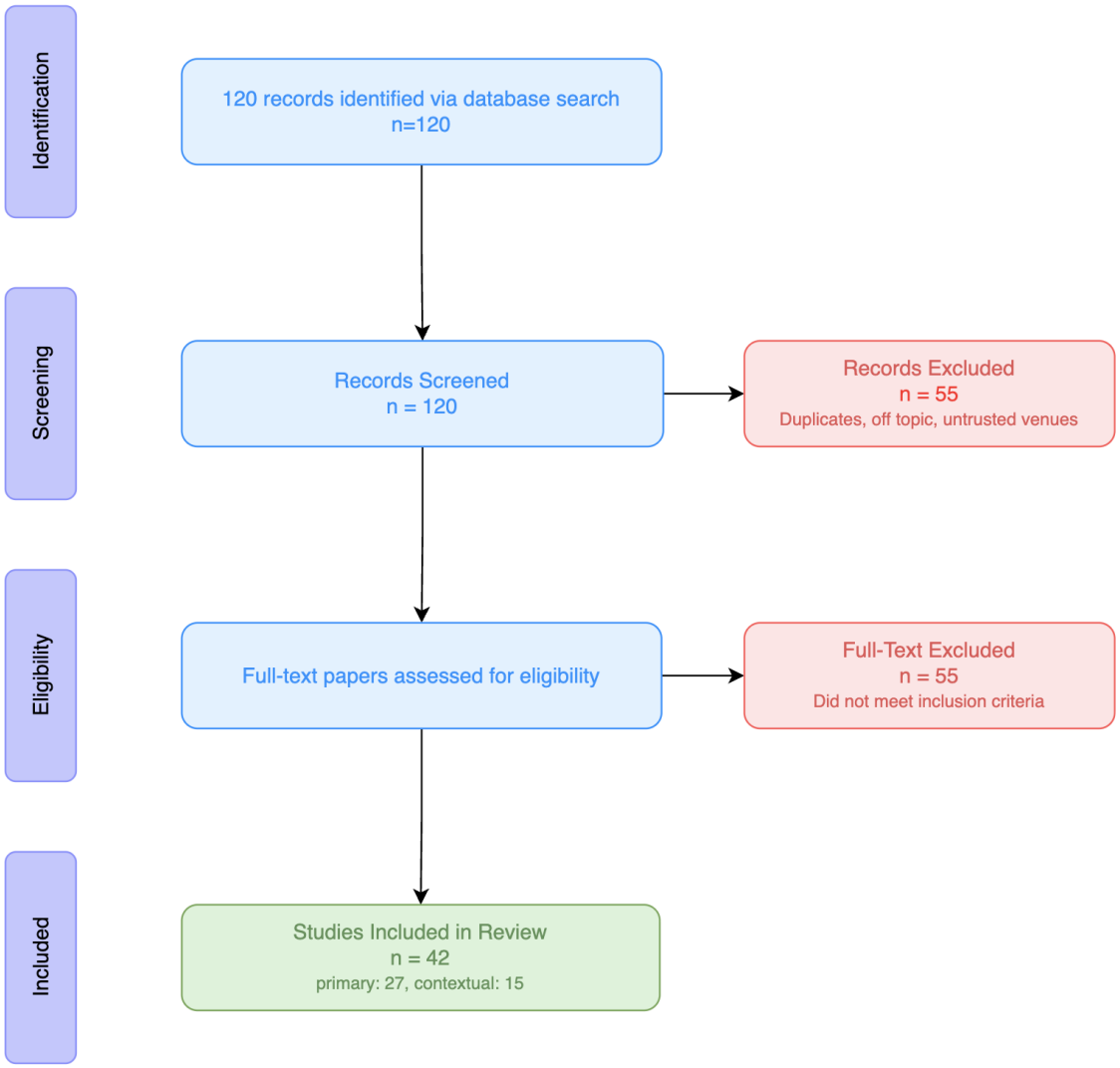

Figure 2 PRISMA Flow Diagram of Literature Selection Process

2. Background

2.1 The EMBER Dataset

Anderson and Roth [8] published the Endgame Malware BEnchmark for Research to the public as an open source, large scale benchmark designed to help achieve reproducible evaluation of static PE malware detectors. As briefly touched upon earlier, the dataset is a collection of approximately one million PE samples, partitioned into three categories based on label: malicious, benign, and unlabeled / unknown. Each sample is constructed by a feature array obtained through eight groups. **ByteHistogram** holds the global byte-value distribution, while **ByteEntropyHistogram** encodes entropy through byte windows of fixed size, and StringExtractor outlines embedded string statistics. **HeaderFileInfo** and **GeneralFileInfo** encodes PE metadata like the file size, section counts, and optional header fields. ImportsInfo and ExportsInfo represent the imported and exported API surface, while DataDirectories describe the presence and size of the PE data directory entries. With each other, these groups build an approximate of 2381 dimensional feature vectors combining low level byte statistics with organized PE metadata. As mentioned earlier, the baseline model shipped with the dataset, which is LightGBM, achieved an AUC of 0.9994 on the built-in evaluation split, setting up a strong reference score, which against subsequent work was benchmarked. The

original 2017 was expanded with an updated 2018 version, featuring additional labelled samples and introducing an altered temporal split to better match how malware emerges over time. The EMBER ecosystem have since been enhanced by multiple extensions, like EMBERSim databank [10] which was presented at NeurIPS in 2023, providing a large scale similarity based search infrastructure that allow for nearest-neightbour retrieval in the EMBER feature space which provided support for tasks including malware family classification and clustering. The benchmark of EMBER2024 [7] presented a holistic, multifamily covering a more expanded set of malware families and more robust temporal based partitioning. Švec et al. [11] built a knowledge based PE dataset that enhances flat feature vectors with richter representations on the semantic level, investigating whether or not structured domain knowledge can enhance the performance of classification and model explainability.

2.2 Static Analysis in Context

Methods of malware detection are widely grouped into static, dynamic and hybrid approaches, where static analysis inspects the static features such as file structure, metadata without executing them, which offers an advantage in both speed and scalability in comparison to dynamic methods where a controlled sandbox environment is required [3]. In actual implementation, the methods of static detection can handle thousands of potentially suspect files per second, which makes the methods highly appropriate to be used at network edges, email gateways and endpoint agents where fast response is essential.

The primary drawback of the static analysis, however, is that it is vulnerable to avoidance by such measures as packing, obfuscation, and manipulation of adversarial features. Packed binaries encode their malicious code, or compress it, making the extracted static properties of the binary deviant of the actual behavior at run-time. Gibert *et al.* [12] revealed that a significant deterioration in the precision of the detectors in the static machine learning occurs as a result of the application of packing. Assaults which alter non-functional header fields of a PE or inject benign code to malicious binaries can also further reveal the vulnerability of feature space classifiers, discussed by Ling *et al.* [13] and Aryal *et al.* [14]. These issues inspired a considerable portion of the analyzed studies, particularly the enhancement of adversarial resistance and mixed detection models.

Wider scans of malware detection, such as the one by Gorment *et al.* [4] and Ferdous *et al.* [2] cluster detection on the basis of the static, dynamic and network-levels. In this landscape, EMBER has a clear role to play, providing a standardized, large scale, and static features rich structure, that can be compared to the various detection architectures under controlled conditions in order to provide the basis on which the comparative analysis of papers published in this review can be performed.

3. Traditional Machine Learning Approaches

A. Gradient Boosting and Tree-Based Models

Gradient boosted decision trees (GBDTs) have dominated early EMBER research, mainly due to LightGBM being the baseline that came with the dataset, which showed that tree-based ensembles are a decent fit for the big, high dimensional data EMBER uses. Pham *et al.* [5] were one of the first to actually test GBDTs on Static PE Features, achieving great results without the need to modify or tune the models heavily. Their research built an important bases which is going to be extended over time, however, temporal evaluation and explainability were both not handled.

Following up, Gao *et al.* [15] built on top of the LightGBM approach by setting up a custom logistic loss function specifically created for production environments where security matters critically. False positives in such settings can cost noticeably more in comparison to a false negative, hence they focused on keeping false-positive rates at their lowest possible rate (under 0.1%) which in return made the model work noticeably better for actual deployment in production in comparison to the standard LightGBM. This low error performance is crucial in security software but it doesn't get the required attention in older / early research. All together these studies prove that tree-based models are a good fit for EMBER's static features

approach. However, continuous reliance on random or fixed train/test splits by researchers causes a limitation to exist. None of these two mentioned studies evaluate performance using time-based partitions to enhance the way a model holds up in real-world production level conditions.

B. Feature Engineering and Dimensionality Reduction

EMBER's features count of around 2381 covering byte level statistics, PE header metadata and API calls information, dimensionality reduction becomes a crucial part when looking to achieve a more efficient and improved accuracy of the model being developed. Among the earliest studies that showed the importance of implementing feature engineering was the study conducted by Oyama *et al.* [16] where they applied a different permutation and split based measures to weigh importance in order to identify the most contributing eight groups of features to the performance of classification. They found that **ImportsInfo** and **ByteEntropyHistogram** were the most valuable of all, that discovery led to the way the community now views and handles feature engineering in EMBER.

Barut *et al.* [17] took the research further by comparing multiple different classifiers such as random forests, extra trees etc. using various feature subsets from EMBER. Their results showed a recurring trade-off between efficiency and detection accuracy of the model. Heavy dimensionality cuts improved throughput of training marginally as well as inference time, but it also drops the accuracy by a few points, although few only, these points matter a lot in real world security scenarios.

Although these have improved, none of the studies mentioned in this review compared the connection between dimensionality reduction and temporal generalization; features that are optimal on a particular split are not necessarily optimal as malware feature distributions change with time and that is an issue that has not yet been tackled.

C. Reinforcement and Adaptive Learning

Instead of the fixed one-time feature selection method, an adaptive feature selection method can be adapted by reinforcement learning (RL), where the agent learns to add or remove features in small amounts based on their small impact on detection performance. Khan *et al.* [18] improved that approach, building on top with a dueling double deep Q-network (D3QN) framework which separates state-value and advantage estimation. This enabled increased stability in learning when offered the high-dimensional EMBER feature space. Adaptive selection methods presented achieved a close, or superior accuracy to the full feature sets even though that effective feature counts were reduced, which could possibly lead to less inference time in resource limited environments. The study however has two important unseen areas, which are that: (a) explainability is not integrated into the RL pipeline and this leaves the selection criteria hidden to security analysts, (b) it doesn't evaluate the architectures robustness under adversarial disruption of the feature space.

D. Summary and Gaps

Across this group, it is seen that GBDTs constantly exceed other traditional algorithms on EMBER benchmark. That confirms their strong fitness in the mixed categorical continuous feature space that PE metadata offers. **Table 1** lists the summary of studies each with their core method and key gap(s).

Several systematic weakpoints do exist however, for starters, the large majority of evaluations relied on random or fixed train/test data splits. This violates the chronological order that exists in malware emergence, and is likely to lead to increased reports of accuracy and AUC values. Secondly, False Positive Rates at important thresholds (e.g. below 0.1%) is only covered by Gao *et al.* [15]. Other remaining studies use overall metrics that keep the actual effectiveness of the models obscured. Third, explainability is completely unmentioned, no traditional ML in the studies covered provide explanation of which features matter for an individual prediction, and that is a gap that constraints analyst confidence and compliance with regulatory.

 ISSN: 2617-5517 | **DOI**: https://doi.org/10.59746/asgg6t42

Finally, the relationship between feature selection methods and temporal robustness remains not analyzed, which offers an opportunity for it to be covered in future work. In the perspective of the accuracy, explainability and robustness trade-off that encapsulates this review, studies in the traditional ML cluster, although showing a strong detection baseline that pushed the research towards covering more advanced aspects, interpretability and temporal resilience challenges remain unaddressed.

Table 1 Traditional ML Approaches Summary

Study	Core Method	Key Gap
Pham et al. [5]	GBDT on PE features	No XAI; no temporal eval
Gao et al. [14]	Custom-loss LightGBM	No adversarial eval
Oyama et al. [15]	Feature importance analysis	Single model only
Barut et al. [16]	Multi-classifier comparison	Fixed temporal split
Khan et al. [17]	D3QN feature selection	No XAI integration

Reading through the table, two patterns are seen at the row level. The column **Key Gap** show no overlapping between studies in what each of them fails to address, every study mentions a different shortcoming, that means the cluster covers a broad research surface but no individual weakpoint has been covered with determined attention from multiple authors. In the same way, the **Core Method** column shows that all five studies follow the same path of tree-based or feature-engineering pipelines, and no return to non-tree classical approaches like SVMs or naïve bayes once GBDTs entered the scene. Combined, this indicates a settled methodological consensus paired with a fragmented gap landscape, a mix that makes follow up work easy to place among others, but difficult to compare directly across studies.

4. Deep Learning Approaches

A. Feedforward and Convolutional Networks

Deep learning enables the extraction of layered, non-linear representations from raw or partially modified PE features directly, highlighting interaction patterns where a manual feature design might be unable to capture, although offering high accuracy, deep learning models have an issue due to the computational overhead sparking the question to whether they are worth using over tree-based approaches. ALGorain and Alnaeem [19] analyzed this by merging cover arrays with grid search which produced a combinatorial evaluation technique that reduced the hyperparameter configuration space while preserving interaction coverage. The yielding optimized DNN architecture outperformed LightGBM baseline on EMBER in general accuracy, showing that deep architectures can extract additional distinctive signals from the same feature set. The study however did not cover details on training period, model sophistication level, which leaves the cost performance trade-off unresolved.

Lai *et al.* [20] examined the relationship between deep learning and feature reduction strategies, they showed that implementing dimensionality reduction prior to training can compensate the architectural complexity increase without giving up detection effectiveness. Their results showed that PCA-reduced features retain decent distinctive information for deep learning models to be on the same level or surpass full feature baseline models, while noticeably reducing the time required for training and resource consumption. Practically, this means that deep learning can achieve more efficiency in resource usage for datasets that are of scales like EMBER's without the need for large simplification of the architecture used that could possibly lead to performance degradation.

Together, these studies suggest that deep learning can outperform GBDT benchmarks mentioned in Section 3, though the enhancements are somewhat limited and are accompanied with some drawbacks such as reduced explainability and higher resource consumption, leading to higher costs. Additionally, studies in this section, like previously mentioned studies, do not offer feature level explanation capabilities for individual predictions nor do they assess robustness in case of adversarial malware.

B. Automated Machine Learning for Deep Models

Designing effective and performant deep learning architectures for the purpose of malware detection needs decent proficiency in both neural networks and cybersecurity, creating a difficult obstacle for practitioners without expert knowledge. This issue was addressed by Brown *et al*. [21] by implementing automated machine learning (AutoML) to deep learning based malware detection, exploring neural architecture designs, activation functions and learning rate schedules. Their tests on feature vectors like EMBER's showed that architectures generated by AutoML can match or outperform the performance of manually designed models, which effectively lowers the barrier of expertise required to deploy deep learning in security contexts. Yet, the cost of computational work of the searching process is still a concern, because even significant GPU resources are still required in guided architecture searches. Reproducibility is also reliant on randomness control and the complete documentation of the search setup, which the study only partially touches. Besides, the selected architectures were benchmarked solely on fixed EMBER data splits, not confirming whether models optimized by AutoML generalize to temporally developing malware distributions or offer enhanced adversarial attacks resistance in comparison to manually constructed models.

C. LLM-Native Static Analysis

An emerging and more research-exploratory body of work uses large language models (LLMs) as a reasoning system over static PE metadata, shifting beyond merely classifying numerical features to natural language inference. Sun and Masum [22] introduced PE2Prompt that is an approach that converts structured PE feature vectors into natural language prompts and queries LLM to generate not only the results of the classification, but also the explanations. This method generates human interpretable explanations along with predictions that may possibly serve to close the interpretability gap observed in conventional deep learning models. Despite this, as the work is an SSRN preprint, its conclusions are to be considered tentative. The approach also comes with important and still unaddressed challenges including the reliability of LLM generated explanations (which could sound true and convincing but actually inaccurate), the latency added to inference in comparison to lightweight models, and the dependence of results on prompt design.

Gill *et al.* [23] proposed LLM-FS which is a similar framework that implements feature selection by using zero-shot LLM in place of full classification. Providing feature descriptions and instructing the LLMs to identify the most applicable parameters for malware detection, the method allows to transfer feature engineering to the model's pre-trained knowledge. Without needing data that is labelled for the selection phase, it was able to produce subsets of features on EMBER like datasets. However, it is yet to have been tested under concept drift or adversarial strategies, nor has its ability to scale to real world systems handling millions of binaries per day been verified.

In general, these LLM based methods show an evolving research borderline, which combines both natural language processing (NLP) and security analysis. However, the evidence supporting is this approach remains negligible, as only two studies, which are both preprints, address this topic and neither outline results on data that's temporally split or against adversarial strategies. Their practicality over time will depend on reducing hallucination risks which are inherent in generative models and on lowering inference requirements and costs to suitable levels that fit real time deployment.

D. Summary and Gaps

The studies, their architectures alongside the key gaps of each one are all summarized in **Table 2**. Deep learning methods reduce the performance difference between EMBER based feature representations and static detection's theoretical constraints. These gains however come with clear tradeoffs in two vital areas. Coming first, interpretability deteriorates as the complexity of the model grows, and neither of both the feedforward or convolutional models reviewed provided explanation for each individual prediction. This lack of transparency

intensifies even more in the study introducing AutoML, where even model design process is automated. Secondly, the computational requirements spike significantly in comparison to tree-based approaches discussed in Section 3, with AutoML searches and LLM based inference being particularly higher in costs.

LLM focused methods which Sun and Masum [22] and Gill *et al.* [23] point toward a capable yet still under development research direction. For these methods to have practice viability, more progress is required in hallucination effects reduction, enhancing prompt reliability, and elevating inference efficiency (which is being actively studied in the general NLP field but still lack enough exploration in malware detection).

Temporal evaluation being missing within this cluster shows the same trend covered in Section 3 and originates from an identical implicit issue. Many studies depend on EMBER's random data splits instead of preserving the chronological structure [8]. In this manner leading to enlarged AUC values that conceals the performance loss under temporal shift as presented by Galen and Steele [9]. Totally, deep learning research's reported accuracy figures are likely to remain highly optimistic and their real-world reliability is still uncertain, until temporal evaluation turns into a standard practice.

Table 2 Summary of Deep Learning Approaches

Study	Architecture	Key Gap
ALGorain & Alnaeem [18]	DNN + grid search	No XAI; fixed split
Lai et al. [19]	DL + feature reduction	No adversarial evaluation
Brown et al. [20]	AutoML / NAS	High computational cost
Sun & Masum [21]	LLM-native (PE2Prompt)	SSRN preprint; latency
Gill et al. [22]	LLM zero-shot FS	No drift evaluation

As visible, only three of five entries (Brown et al., Sun and Masum, Gill et al.) suffer from gaps that are tied to the immaturity of the approach itself, rather than to a missed evaluation, which separates them from the first two studies where the gaps are conventional methodological omissions. This division suggests that the deep learning cluster comprises two parallel sub-streams operating at distinct stages of validation. One stream is exploration-oriented, testing whether AutoML and LLM-based reasoning can be made viable at all, and the other is an established branch that refines known architectures. It's reasonable to review both studies under the same heading for scope reasons, but it shouldn't be interpreted as a comparison of studies with equal evidentiary bases.

5. Ensemble and Hybrid Model Architectures

A. Homogeneous and Heterogeneous Ensembles

Ensemble learning mixes multiple base models in order to achieve reduced variance, address bias and improve generalization further than the capability of any individual model. Among the EMBER framework, such strategies have been applied using both identical configurations where identical algorithms learn using varying initial conditions or subsets, and heterogeneous setups which combine a variety of algorithms by using strategies like voting or stacking.

Early contributions focus on different methodologies to heterogeneous ensemble construction. For example, Azeez *et al.* [6] merged decision trees, naïve bayes, and support vector machines (SVM) through a majority voting structure. Their findings demonstrated that combining distinct models can help lower variance and in return lead to an accuracy increase in comparison to any single PE malware detection classifier. Developing into this direction, Damaševičius *et al.* [24] merged neural network parts into an ensemble framework. This had the effect of improving detection rates, especially in the case of obfuscated or packed types of malwares, cases where classical tree-based approaches often struggled because of the alteration that can be done to static features. However, the emergence of neural network

yielded an increase in resource requirements, whether during the training phase or inference. Neither of studies addressed explainability or evaluated performance over time to address temporal evaluation, allowing uncertainty to be present around model transparency and long-term effectivity.

B. Stacking and Meta-Learning

Stacking (or stacked generalization) involves the process of training a higher-level model (meta learner) on predictions produced by multiple base classifiers through a practice known as cross-validation. This method enables the system to combine outputs in a supervised way by learning optimal ways to carry out this procedure. The robustness of stacking largely depends on the diversity of the base models and the choice of the meta learner, both of the aforementioned have an impact on predictive performance and the computation efficiency.

Barut *et al.* [17] compared stacking to other ensemble strategies like bagging and boosting using features obtained/derived from EMBER. Their results indicated that stacking with heterogenous base learners constantly achieved higher AUC and F1 scores in comparison to homogeneous methods. In spite of that, they also noted that stacking presents a significant overhead to the training process due to the requirement of cross validated predictions from each base model making the training phase turn into a much more computationally heavy process when compared to boosting approaches such as LightGBM. In practice, a stacking setup with three base models and fivefold cross validation can demand nearly ten times the duration of training required by an individual LightGBM model. Although this can be acceptable for the development of an offline model it quickly becomes impractical in contexts where frequent retraining is a critical requirement, such as those targeting concept drift.

C. Hybrid Models: Integrating Static and Structural Features

Hybrid systems aim to group different types of features or learning methods within a unified detection pipeline, which enables the systems to capture information that an individual method does not have the capability to do. This augmentation can yield noticeable enhancements generating better performance. For instance, in the “Quo Vadis” framework by Trizna [25], static contextual features (EMBER compatible) were combined with behavioral data taken from dynamic analysis. This combination provided higher AUC scores and showed a reduction in false positive rates when compared to models depending individually on static or dynamic inputs.

However, the benefits that come from incorporating dynamic analysis also have trade-offs due to the requirement of sandbox environments capable of executing suspected malware, which in itself introduces additional expenses, increase in latency, and challenges related to scalability. Consequently, such systems may not be a good fit for real time or high throughput scanning. Although that organizations that are already equipped with infrastructure capable of running sandbox environments may benefit from the enhanced accuracy, others may find it difficult to justify the needed investment. Vuran Sarı and Acı [26] proposed a hybrid CNN-GRU model that processes raw byte sequences and structured PE features both. The architecture is comprised of a convolutional neural network (CNN) that extracts local patterns through byte level data, while a gated recurrent unit (GRU) picks up sequential relationships throughout different PE sections. These representations of features are then merged for final classification. An outstanding aspect of their presented approach is the simultaneous use of LIME and SHAP for explainability or interpretability, enabling one to one comparison of attribution methods on identical predictions. Nonetheless, producing explanations from both techniques at inference time presents a new challenge, which is the additional computational overhead, raising concerns about practicality in deployment environments where time is a sensitive aspect of the operation.

 ISSN: 2617-5517 | **DOI**: https://doi.org/10.59746/asgg6t42

D. OPTISTACK: Ensemble + XAI for Compressed Files

Sujon *et al.* [27] presented OPTISTACK, which is a composite ensemble framework that integrates explainable artificial intelligence (XAI) for malware detection in compressed file types like ZIP and RAR archives. Their approach mixes a variety of base models: LightGBM, random forest, XGBoost, using a stacked architecture, in which a meta learner is trained to accumulate the resulting outputs. In aims of enhancing interpretability, a SHAP based post-hoc analysis is employed to highlight the features that have the highest effect the decision of each classification.

By addressing a relatively neglected attack surface, this method generalizes the scope of the EMBER framework. Compressed files are regularly exploited to drop malware and malicious payloads while circumventing introductory security checks of email gateways and web proxies. in spite of this improvement, a crucial concern arises linked to feature compatibility The parameters obtained from the compressed archives (such as file headers, internal directory layouts and compression ratios) vary substantially in comparison to PE based features employed in EMBER, that includes section tables and imported packages/libraries. As a result, it remains unclear if patterns that were learned from PE datasets can be transferred effectively to individual feature domain, and this belief has not yet been analytically validated.

Even when considering these constraints, OPTISTACK's integration of explainable AI into the ensemble system denotes a meaningful impact. It indicates a development focus on transparent and explainable malware detection systems, where the decisions of the model are understood, is becoming as much of value as accomplishing high predictive performance.

Table 3 Summary of Ensemble and Hybrid Approaches

Study	Architecture	Key Gap
Azeez et al. [6]	Hetero. voting ensemble	No XAI; random split
Damasevičius et al. [24]	NN + ML ensemble	No temporal eval
Barut et al. [16]	Stacking vs. boosting	High training overhead
Trizna [25]	Static + dynamic hybrid	Requires dynamic infra.
Vuran Sarı & Acı [26]	CNN-GRU + LIME/SHAP	High inference cost
Sujon et al. [27]	OPTISTACK (stack + XAI)	Compressed-file scope

A row-level analysis of Table 3 reveals that the Key Gap column is evenly distributed across infrastructure cost, scope constraint, and missing evaluation dimensions. Notably, no two studies share the same shortcoming. This contrasts with the more pronounced convergence observed in the deep learning cluster, suggesting that ensemble work has progressed by exploring diverse architectural strategies rather than refining a singular template. This leads to complicating direct comparisons and partly explains the absence of a unified evaluation across this cluster.

E. Summary and Gaps

Summarized in **Table 3** are the studies of these sections with the architecture of each and the key gap(s) they come with. Ensemble and hybrid models provide the highest accuracy reported in the work reviewed, with configurations of stacking and heterogeneous voting being constantly superior single classifiers on the EMBER benchmark. However, this advantage of accuracy comes with three significant compromises. First, both model complexity and the latency of inference get a substantial increase, specifically in case of stacking frameworks, where outputs from multiple base models must be produced before a meta-learner can make a final prediction result, in addition to the case of hybrid systems where both analysis techniques of static and dynamic are combined. Second, explainability is

reduced in comparison to more simple individual model methods. Although single tree-based models are often fairly interpretable, the decision-making process turns into a less clear one when a combination of multiple models is made through voting or stacking, hiding the path that the model takes from input features to the final outcome. OPTISTACK [27] and CNN-GRU model [26] integrate explainable AI (XAI) directly into their ensemble framework, but neither assesses how stable these explanations stay with the passage of time. Third, there is a continuous lack of temporal validation and adversarial robustness evaluation across this class. No studies mentioned that accommodating ensemble frameworks investigate a decline in performance to changes in temporal distributions, or whether their resilience to adversarial features space assaults can be validated. This disjury gives a wide discrepancy between evaluative accuracy and real reliability. Due to these considerations, ensemble methods rank top in a range of tradeoffs of accuracy, albeit at the cost of no interpretability and no proven time robustness. This problem is also well highlighted in **Table 6**, as it is stated that all six datasets of ensemble or hybrid models do not address the issue of temporal testing, and only two incorporate XAI in any way. This field is not a pure incidental case of the lack of XAI adoption, it indeed has its structural challenges roots. Techniques like SHAP are designed explain sole prediction functions, which is when applied to ensembles, they interpret each base model independently instead of illustrating how the outputs are to be assembled. For example, OPTISTACK [27] generates explanations using SHAP for its foundational models but does not cover how the meta learner weights these predictions. Thus, the stacking or voting mechanisms, where much of the accuracy increase comes from, stay mostly opaque, even in cases when the individual components are explainable. A study that successfully incorporates strong ensemble performance with a full explainability pipeline (including transparency in the combination stage) with temporal evaluation would fill one of the most apparent gaps in the current field of research.

6. Explainable AI in EMBER-Based Detection

A. SHAP-Based Explanations

SHAP (Shapley Additive exPlanations), which is rooted in cooperative game theory, emerged as the top XAI technique for the use in malware detection research that is EMBER-based. It assigns an additive contribution to each feature for a certain prediction; this allows both local explainability (at the level of each sample) and global explainability (throughout the dataset). A constantly appearing limitation in this area is the absence of governed user evaluations that involve security analysts. Al Siam *et al.* [28] demonstrate this trend in their LightGBM approach enhanced by SHAP, which produced explanations at the feature level for each prediction and underlines the most influential variables behind decisions, uncovering practically relevant indicators such as improved import table entropy and inconsistent sizes of data directory. However, it is still unclear whether these insights are actually of enhancement to analysts' real-world decision-making scenarios.

Two additional studies apply SHAP to various settings in which the attributions of features are very dependent on the settings. Kumar and Subbiah [29] have also integrated SHAP as a part of an ensemble model to detect zero day malware, and the feature importance rankings are used to determine the best PE parameters to use in the identification of a malware family never before observed. This constitutes a remarkable example where explainability functions effectively to generalize rather than a post-hoc utility. It still is in any case, since it was evaluating with a random divided set of data, uncertain whether these attributions are isolated through time as malwares change and become more sophisticated.

Comparatively, Manthena *et al.* [30] applied SHAP to a streaming context, in which models successively generate predictions without looking at the entire dataset. They showed that rankings of feature-importances vary between batch and online environments, indicating that feature-importances explained offline may not be directly transferred to settings involving production. Just a wide-ranging study was conducted by Baghirov [31] on the strength of XAI explanations when it comes to adversarial manipulation. In the work especially, whether small and functionality preserved changes to PE files had the potential to alter SHAP attributions

greatly. The results confirmed a significant lack of stability, with even slight perturbations causing a significant change in the explanation, raising insecurities about their drawbacks with adversarial malware. This has dire consequences of trust, as the fact that explanations that change radically under small input changes gives misleading directions as opposed to ameliorating the security analysts. Throughout all these, an identifiable gap is the absence of a set of standardized reliability measures to test SHAP explanations in malware detection. While there are well established metrics like monotonicity, faithfulness and stability in the general XAI field, they aren't systematically implemented in the studies reviewed. Such lack of standardized testing principles complicates the comparison of explanation effectivity across works or describing baseline standards on how the real-world works.

B. LIME, LEN and Alternative XAI Methods

Several studies beyond SHAP explore different explainability techniques that vary in their core methodology and the explanation type they produce. Anthony *et al.* [32] presented Logic Explained Networks (LENs) as ante-hoc alternative for post-hoc attribution methods. In comparison to SHAP and LIME, which interpret the outcome of previously trained black box models, LENs integrate explainability directly within the structure of the model by producing logical rules that are human readable, alongside predictions. This ante-hoc technique evades the accuracy issues related to post-hoc methods, as the explanation shows the model's reasoning process instead of an approximate. However, it remains unclear whether LENs can scale well to high dimensional datasets like EMBER with roughly 2381 features, knowing that the count of possible logical rules grows in a combinatorial way with feature size. Svec *et al.* [33] advances further into the ante-hoc paradigm through developing semantic representations of data that encode PE features in a way intrinsically appropriate for interpretable modeling. By converging raw feature values into profound categories (e.g. high import diversify or unusual section entropy), their methods enable models to work on hypotheses that are directly reasonable to cybersecurity professionals, in that way achieving a reduction in reliance on isolated post-hoc explanation techniques.

Generally broader perspectives are provided through surveys by Manthena *et al.* [34] and Saqib *et al.* [35], participating in the overall XAI terrain in malware detection. As a review does, EMBER positions larger methodological trends targeted studies in the middle as such places and gains attention to the, in general, stability in the disjunctive position between the superiority of the available XAI methods and their practical implementation. To be more precise, it is observed that the majority of operational systems are still reliant on post-hoc explanations like SHAP or LIME, rather than switching towards the one that possesses more theoretical base of ante-hoc methods that are on the rise of research. The separation between post-hoc and ante-hoc approaches illustrates an imperative trade-off in decipherable maliciousness evaluation. Post-hoc methods like SHAP and LIME are model agnostic and flexible, so will apply to potentially any classifier, but they provide explanations at the approximate level that can be short of fully capturing the true decision rationale. By contrast, networks to which the ante-hoc methods, such as LENs and semantic illustrations, are applied provide naturalistically realistic explanations, but they have structural restrictions that may limit scalability and model expressiveness. Nonetheless, none of the reviewed studies explicitly considered the interaction of analysts and benefit of such explanation types that are distinct based on controlled user experiments. That puts a lot undecided on their actual performance on security operations centers (SOC).

C. Metric Embeddings and Similarity Based Explanation

Pathway to achieve explainability not only goes through feature attribution. The replaceable framework that Rudd *et al.* [36] describe is metric embedding spaces. Their method, which projects samples into a learned embedding space, assigns each single feature importance scores, instead of using single features, which conform to behavioral resemblance in distances calculated by quantifying similarity (e.g. Euclidean similarity or cosine similarity). Different samples in that environment can be visualized based on which known

malware or benign clusters are resembling it most (e.g. “this sample is closely related to the cluster X of ransomware).

This explanation based on semblance is more intuitive to analysts in security operations centers (SOCs) than are abstract numerical attributions. Despite this fact, this approach also has its shortcomings such as loss of details at feature level. Although embedding methods show what a sample is identical to, they do not give an explanation to why that similarity exists, which is an important characteristic in tasks like investigating root cause and incident response.

Forthcoming research that incorporates metric embedding strategies with feature attribution approaches could possibly provide a standard explanation framework, capturing both “what” sample is similar to and “why”.

D. Summary and Gaps

Table 4 summarizes each study in this group, with their key gaps and XAI methods they used. Three cross cutting observations appear from this cluster. One, in EMBER-based detection, XAI landscape is dominated by SHAP, yet explanation effectiveness is rarely confirmed with professionals or evaluated against unified accuracy metrics. The topic lacks a consensus evaluations framework similar to those founded in other domains such as medical imaging or natural language processing (NLP). Two, approaches that are ante-hoc such as LENs and semantic representations remain substantially underexplored in relation to their potential for integrated interpretability, largely due to the fact that they impose architectural limitations that make adoption among existing high accuracy classifiers complex. Three, studies covered in this review don’t combine multiple explanatory models (feature attribution, logic rules, similarity-based explanation) into a standardized pipeline that could have the ability to provide analysts with complementary perspectives on an individual detection decision. The persistence of unexplained explanations has an initial source that cuts across the study design. In comparison with other areas like medical imaging or NLP, the research on malware detection is yet to have the standard framework on how to assess the quality of XAI. Saqib *et al.* [35] underline that most studies do not put any interpretability metrics to use, while Manthena *et al.* [34] recognize that quantitative testing of interpretable models remains an open problem. Without shared benchmarks that the community can all agree to use, researchers lack a clear way to test or compare explanation quality, which hinders motivation to invest in detailed validation of XAI. Although explainability has seen enhancement, it still lacks the required accuracy and temporal robustness that the research in malware detection leans into prioritizing.

Table 4 Summary of XAI Approaches

Study	XAI Method	Key Gap
Al Siam et al. [28]	SHAP + LightGBM	No analyst user study
Kumar & Subbiah [29]	SHAP + ensemble	Random split only
Manthena et al. [30]	SHAP (black box)	Online setting only
Baghirov [31]	XAI + Robustness	No concept drift eval
Anthony et al. [32] & Švec et al. [33]	Ante-hoc methods (LEN; semantic rep.)	Early stage limited scalability
Rudd et al. [36]	Metric embeddings	No attribution-level XAI

7. Zero-Day Detection and Concept Drift

A. Zero-Day Generalization

A malware is only actually useful if it is able to generalize farther than the malware families observed during the training phase to catch new, genuinely novel threats. This zero-day detection challenge differs from standard generalization, because the test data may contain entirely new malware families with different qualitative behaviors and structures.

Two main methods have been analyzed for addressing zero-day challenges, each providing a different set of advantages. Kumar and Subbiah [29] utilized a mix of bagging and boosting with SHapley based ensemble methods, together with SHAP (Shapley Additive exPlanations) to underline the features most linked to unknown malware families. This does not only improve detection overall, but it also gives professionals, whom are tasked with analyzing samples, the ability to gain useful insight into emerging threat types.

Comparatively, Jurečková *et al.* [37] introduced an online strategy for clustering, where input samples are either assigned to previously created groups, or used to form new groups if not close enough match is achieved. This pushes continuous identification of new malware types in flowing environments.

The first approach stands out for combining explainability with generalization, when the second is better for real world settings, in cases where data is continuously being fed into the system. However, neither of the two use temporal data splits or tests that utilize different time gaps between training and testing, which an important limitation that both approaches suffer from. This can be tricky in a real-world scenario where months or even years can cause the model training to drift away with the emergence of new types of malwares in the world, and this is where it is difficult to determine how performance declines with time. It would be of useful assistance to evaluate the effect on the accuracy of zero-day detection as this gap increases and in doing so ascertain when models need to be retrained when using them in a production system.

B. Concept Drift Under Temporal Shift

Malware feature distributions change over the course of time as malicious actors evolve and develop their strategies, introducing new evasion methods, and letting go of old methods. This phenomenon which is known as concept drift, pushes the model's performance towards degradation unless systems are routinely retrained or built to adapt without human interference.

Galen and Steele [9] presented one of the very early analytical works regarding performance degradation on temporal-bases for ML models trained on the EMBER dataset's PE. Experiments they carried showed significant AUC decline when models trained on older data is tested against new current/later ones, with performance taking hits of several percentage points down even over periods as short as six months. It underlines the importance of temporal based testing and represents that random data splits have a tendency for over estimating real world performance.

Maillet and Marais [38] proposed optimized deep learning strategies that mitigate the effects of concept drift, their method include a routine retraining with refreshed labelled data and architecture selections, like wider layers and dropout. Although the modifications under consideration resulted in slower decrease in performance, they nevertheless did not wipe the latter out, which is why it can be concluded that retraining still has to be implemented, yet not that frequently.

A lightweight ensemble technique, which is suggested by Manikandaraja *et al.* [39] to change the paradigm of drift, is referred to as RAPidDrift, where certain components of the model are changed instead of re-training the whole ensemble on everything. Being able to actively control a group of learners, and performing a selective replacement of poor performing learners, RapidDrift decreases the cost of the computational operation whilst enhancing responsiveness to changing data. Its capacity to cope with sudden changes was not provided, though.

The major limitation throughout these works is the absence of a single time benchmark, with their different temporal splits, and different evaluation configurations, it is difficult to directly compare the results of methods and determine who has the best trade-off between accuracy retention with computational cost.

C. Continual Learning for Malware

Ongoing or gradual learning emphasizes the importance of updating the models with new data and retaining the earlier learnt knowledge. It would be necessary in malware detection when systems are required to identify new fresh threats without forgetting old, potentially still relevant threats. Rahman *et al.* [40] analyzed the problems of conducting continuous learning on malware classification. In their study, they discovered that the simple attempts of experience replay involving the storage and reutilization of previous samples are ineffective in real-life circumstances. Compared to other areas such as computer vision, malware evolves quickly, forming significant changes of distribution that cannot be effectively tackled by replay.

This issue was addressed by using distribution aware reiteration in the following framework MADAR [41] Instead of sampling old data randomly, it picks out samples that best model previous disturb (disruptions) enabling the model to retain valuable knowledge. This method, as compared to other standard replay methods, exhibits a considerable decrease in forgetting with EMBER like databases. But MADAR, too, is limited in practice. The maintenance of a representative replay buffer across multiple time periods necessitates data management and extra storage resources, potentially complicating its use in resource constrained systems, such as endpoints and edge devices. Whether this overhead can be reduced or not remains unclear and an unanswered question, or if substitute approaches like parameter regularization or model expansion could obtain similar outcomes without depending on stored data.

D. Adversarial Robustness

Attacks of adversarial nature expose core weakness points in static detectors, by showing that minimal, strategically designed modifications to Portable Executable (PE) files can lead to improper identification while preserving the binary's malicious behavior. Demetrio *et al.* [42] demonstrated the feasibility of functionality-retaining black box adversarial attacks directed at Windows PE malware detectors, showing that gradient free enhancement can detect disruptions that have the capability to bypass detection techniques without the need to access internal model attributes. Their following survey [42] and earlier analysis of vulnerabilities [43] offer a complete mapping of the attack surface of deep learning based static malware detectors, recognizing the range of strategies, from the manipulation of headers to the level of injecting sections and the addition of overlay data. Advancing contributions by Ling *et al.* [13] and Aryal *et al.* [14] expand on this work by encompassing both feature space attacks, where the feature representation extracted is directly manipulated, and problem space attacks, which involve the modification of the binary itself, all while keeping its operational integrity. Gibert *et al.* [12] explored the puzzling effect of packing on static ML detectors, showing that packing functions as a natural non adversarial alternative to adversarial disruption. The representation of the static features in both the scenarios is misshapen without interfering with the runtime semantics of the program.

An extensive literature review indicates that the adversarial resistance of the model has not received much attention beyond this particular area of study. The only study of the detection oriented studies discussed in Section 3 to 5, which has taken the stability of model explanations to adversarial conditions, was by Baghirov [31], and none of the studies reviewed in this paper has reported its detection accuracy under adversarial conditions. This separation highlights a critical divide: the resulting lack of integration has allowed adversarial approaches and detection systems to develop separately, leaving even high-performance EMBER-based classifiers with insufficient robustness in practice.

 ISSN: 2617-5517 | **DOI**: https://doi.org/10.59746/asgg6t42

Table 5 Summary of Zero-Day and Concept Drift Approaches

Study	Core Method	Key Gap
Kumar & Subbiah [29]	SHAP + ensemble (zero-day)	Random split only
Jurečková et al. [37]	Online clustering	Distance metric sensitivity
Galen & Steele [23]	Temporal decay study	Single model type
Maillet & Marais [38]	Deep learning under concept drift	Partial drift mitigation
Manikandaraja et al. [39]	RapidDrift ensemble	No abrupt-drift evaluation
Rahman et al. [40]	Continual learning limits	Replay inadequacy
Rahman et al. [41]	MADAR (distribution-aware replay)	Buffer overhead
Demetrio et al. [42]	Black-box adversarial attacks	Attack-focused; no defense
Gibert et al. [11]	Packing impact on machine learning	No adversarial intent

A useful observation emerges from the comparison of this table with the previous four. Five of the nine rows here report temporal evaluation, whereas across Tables 1 to 3, not a single one of the studies do. This asymmetry reflects the methodological divide discussed earlier in this section. The table also reveals that none of all the nine studies record explainability or adversarial robustness, except for one, which is Demetrio et al.'s, where it itself is adversarial in focus rather than in defense. Consequently, the drift cluster inherits the temporal firmness of the detection clusters but it also lacks their focus on interpretability. This leaves the three review dimensions distributed throughout separate research communities rather than addressed within any single study.

E. Summary and Gaps

A summary of studies, core method and core gaps is given in **Table 5**. Two can be observed throughout this stream; first, temporal evaluation is prevailing over the studies on concept drift but occurs virtually never in the detector-oriented work reviewed in the previous sections with a resultant division between who quantifies the problem and who designs detection models.

Second, the concept of drift aware and adversarial robustness have emerged primarily as research directions in their own right, and the two have been limitedly incorporated. This difference seems to be a consequence of different priorities in research, instead of an incompatibility scheme. Static detectors with the adversarial field are predominantly examined as targets for avoidance: e.g., Demetrio *et al.* [44] provide an in depth exploration of strategies used for evasion while deferring robustness advancements to future work. In comparison, the studies that focus on detection usually treat adversarial robustness as far from their intended scope. Ling *et al.* [13] underlines the consequences of this division, stating that there is scarce adversarial defense methods for developing more robust PE models for malware detection.

Consequently, community of each research group effectively describe their contribution in a manner that excludes the essential concerns of the other. A result of that is the lack of studies that integrate temporal evaluation with adversarial robustness analysis, leading to an unresolved matter of whether that adversarial susceptibility grows, decreases or remains stable as detection models develop over time.

8. Critical Analysis and Research Gaps

Six intersecting gaps emerge from the reviewing of studies covered that transcend individual thematic clusters and limit the maturity of the research covering EMBER based static malware detection. They are listed as follows.

(1) Evaluation Methodology: As shown in **Table 6**, only **5** of the primary **27** studies implement temporal evaluation, and all of those five are of the concept drift group. The rest 22 studies use random splits of train/test from the EMBER dataset, which is a practice violating the temporal ordering bound to malware emergence and artificially fills reported performance benchmarks [9]. Due to the fact that malware feature distributions evolve over time, evaluating models using randomly sampled test sets that are highly identical to what they were trained on results in very optimistic evaluation results, making models look very accurate and performant. This results in an unrealistic set of positive results that do not reflect real world scenarios where the model has to perform well against new not previously seen data. A better approach is to split the data by time so that the model is trained on samples of an older time frame, then tested on samples from a more recent one. This grants realistic measurement of the model's performance in production.

(2) False Positive Rate (FPR) Reporting: In real world, malware detection systems must maintain a low false positive (typically below 0.1) rate of reports since false detection of a legitimate file can result in interference with user processes and undermine the confidence of user in the system. With that said, this practical need is not widely displayed in the studies provided. Among 27 primary studies, only 3 work (as indicated in Table 6) reveal FPR results that are consistent with operational requirements. This is because, among these few studies, Gao *et al.* [15] is the only one that research specifically is looking to low FPR conditions, whereas Trizna [25] and Pham *et al.* [5] also give threshold based FPR. Other studies (most of them) however are based upon summary metrics such as accuracy, F1 score etc., these metrics are likely to encompass whether the models are applicable to real world deployment or not.

(3) Reproducibility: Although EMBER dataset is open access, required key components that are used to replicate tests (like data processing, hyperparameter settings, pipelines, etc.) are rarely available. This lack of accessibility constraints the ability to confirm results. The challenge gets intensified by the inclusion of preprints from platforms such as arXiv and SSRN [22], [32] as well as studies published in less reputable venues such as IJCNIS, as mentioned earlier in this review. These studies often bypass rigorous peer review, and without the ability to access the original experimental framework, their results cannot be replicated and hence cannot be independently confirmed.

(4) Explainability and Performance: Among the reviewed papers, ensemble and hybrid models constantly provide the best accuracy in detection. In spite of that, this performance advantage often causes a reduction in model's interpretability. As covered in Section 6, explainable AI (XAI) approaches are applied typically post model training, using SHAP, LIME or other similar techniques for producing explanations. Still, none of the studies assess the practicality of these explanations via structured testings with the participation of security professionals. As a result, a gap persists between the development of XAI methods at the academic level, and their applied integration into security operation environments.

(5) Adversarial Robustness: Whether in feature or problem spaces, adversarial attacks are comprehensively analyzed in specialized research [13], [42], yet they rarely are put into consideration when testing malware detection systems. As covered in **Table 6**, out of the total of 27 primary studies, only one study by Baghirov [31] addresses adversarial aspects,

 ISSN: 2617-5517 | **DOI**: https://doi.org/10.59746/asgg6t42

and even in that case, the attention is limited to the stability of explanations rather than the model's detection performance when under attack. This gap underlines a clear distinction between adversarial research and detection focused studies, resulting in a major barrier in the path of EMBER-based systems deployment in environments where active attempts to evade detection are done by attackers.

(6) Dataset Monoculture: Heavy reliance on a single dataset like EMBER in this case, as the only testing benchmark limits how general the findings of the reviewed work can be applied. None of the studies incorporate cross-dataset assertion using substitute benchmarks such as SOREL-20M [45], MOTIF or VirusShare. This limited focus results in a "monoculture" in other words, it results in the overreliance on a single source, which in return could make models become implicitly tailored to the specific attributes of EMBER, such as its distribution of data, labelling criteria, time span, etc.

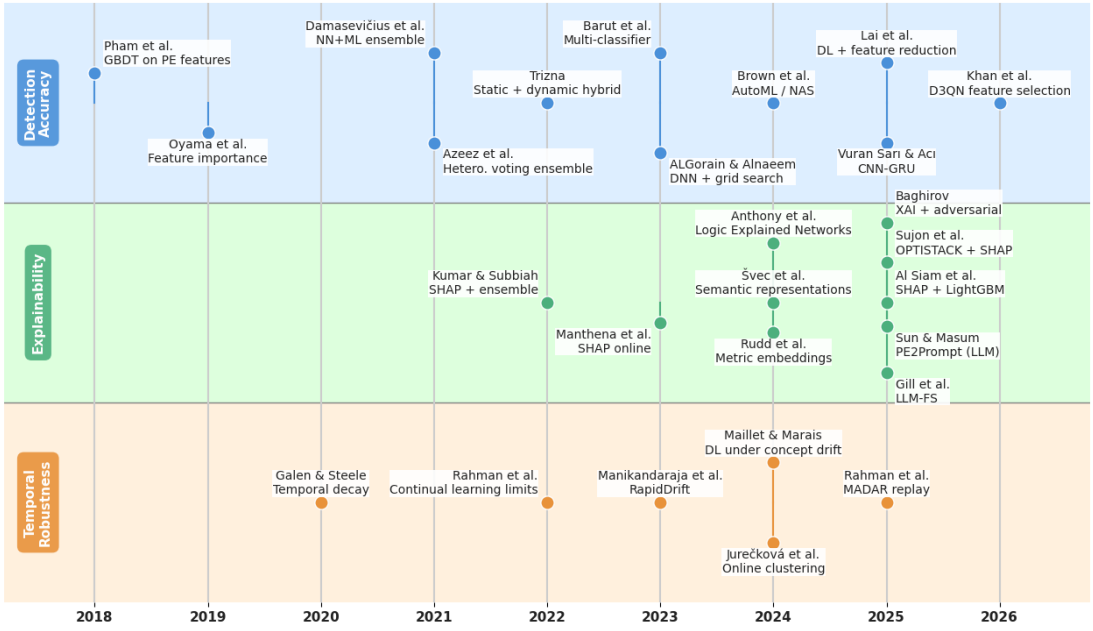

Figure 3 Research Evolution Along the Three Trade-off Dimensions

Table 6 Master Comparison of Primary-Corpus Studies

Study	Cluster	Core Method	Temp.	FPR	XAI	Adv.	Venue
Pham et al. [5]	Trad. ML	GBDT on PE features	N	Y	N	N	Peer-rev.
Gao et al. [14]	Trad. ML	Custom-loss LightGBM	N	Y	N	N	Peer-rev.
Oyama et al. [15]	Trad. ML	Feature importance	N	N	N	N	Peer-rev.
Barut et al. [16]	Trad. ML	Multi-classifier	N	N	N	N	Peer-rev.
Khan et al. [17]	Trad. ML	D3QN feature sel.	N	N	N	N	Peer-rev.
ALGorain & Alnaeem [18]	Deep L.	DNN + grid search	N	N	N	N	Peer-rev.
Lai et al. [19]	Deep L.	DL + feature red.	N	N	N	N	Peer-rev.
Brown et al. [20]	Deep L.	AutoML / NAS	N	N	N	N	Peer-rev.

Sun & Masum [21]	Deep L.	LLM (PE2Prompt)	N	N	Y	N	Preprint
Gill et al. [22]	Deep L.	LLM zero-shot FS	N	N	N	N	Peer-rev.
Azeez et al. [6]	Ensemble	Hetero. voting	N	N	N	N	Peer-rev.
Damasevičius et al. [24]	Ensemble	NN+ML ensemble	N	N	N	N	Peer-rev.
Barut et al. [16]	Ensemble	Stacking vs. boosting	N	N	N	N	Peer-rev.
Trizna [25]	Hybrid	Static + dynamic	N	Y	N	N	Peer-rev.
Vuran Sarı & Acı [26]	Hybrid	CNN-GRU + XAI	N	N	Y	N	Peer-rev.
Sujon et al. [27]	Hybrid	OPTISTACK (stack)	N	N	Y	N	Peer-rev.
Al Siam et al. [28]	XAI	SHAP + LightGBM	N	N	Y	N	Peer-rev.
Kumar & Subbiah [29]	XAI	SHAP + ensemble	N	N	Y	N	Peer-rev.
Manthena et al. [30]	XAI	SHAP (online)	N	N	Y	N	Peer-rev.
Baghirov [31]	XAI	XAI + robustness	N	N	Y	Y	Peer-rev.
Anthony et al. [32] & Švec et al. [33]	XAI	Ante-hoc (LEN; semantic repr.)	N	N	Y	N	Preprint
Rudd et al. [36]	XAI	Metric embeddings	N	N	N	N	Peer-rev.
Galen & Steele [23]	Drift	Temporal decay	Y	N	N	N	Peer-rev.
Maillet & Marais [38]	Drift	DL under drift	Y	N	N	N	Preprint
Manikandaraja et al. [39]	Drift	RapidDrift	Y	N	N	N	Peer-rev.
Rahman et al. [40]	Drift	Continual learning	Y	N	N	N	Preprint
Rahman et al. [41]	Drift	MADAR (replay)	Y	N	N	N	Preprint
Jurečková et al. [37]	Drift	Online clustering	N	N	N	N	Peer-rev.

Temp. = temporal evaluation; FPR = false-positive rate at operational threshold reported; XAI = explainability method included; Adv. = adversarial robustness evaluated. Venue: Peer-rev. = peer-reviewed; Preprint = arXiv/SSRN; Weak = unestablished venue.

 ISSN: 2617-5517 | **DOI**: https://doi.org/10.59746/asgg6t42

9. Future Research Directions

Considering the gaps identified in the previous section, seven solid research directions are presented to advance the field.

1. **Temporally Aware Evaluation Protocols:** To address **Gap 1 (Evaluation Methodology)**, research community should unify the use of time-bound splits for training and testing. Actual applications of this approach already exist. For example, Galen and Steele [9] presented a month by month testing scheme on EMBER2018, where models are trained on data covering months of January and February, then tested on each one of the following months till December. EMBER2024 [7] further enhances this idea, which implements a rolling framework comprising of a 52-week training duration followed by a 12 week long test phase drawn from a dataset that exceed 3.2 million samples. Beyond that standard partition of temporal data, EMBER 2024 also presents a challenge split, which is a harder benchmarking protocol designed to expose models to distributional shifts that are more severe than the ones encountered in the standard temporal duration. Due to the fact that the challenge split picks test samples that are ultimately distant from the training distribution, it makes models that perform well against it provide a more realistic and credible estimate of real-world production level reliability than those tested on the standard partition (train / test) only. Consequently, future research should present results across both partitions of data, which are the standard temporal split to support consistent comparison across studies, and the challenge split which, as mentioned earlier, provides realistic estimates of performance under real world deployment conditions. Precisely, researchers should report AUC and detection rates at several temporal intervals (such as 1 month, 2 month and 6 months gaps) for the standard split, while discretely presenting results on the challenge split to cover the scope of performance degradation in case of the most demanding evaluation settings. A clear adoption criterion can also be outlined explicitly. A protocol may be considered recognized when studies enable direct comparison at regular temporal intervals, and when AUC decay curves (improved with the outcomes of the challenge split) take place of single value accuracy measurements as the foundational base for performance evaluation.
2. **Integrated XAI Pipelines:** In aims of addressing Gap 4 (Explainability vs Performance) future work should shift away from post-hoc techniques of explanation, towards inherently explainable ante-hoc models embedded within the architecture itself. Anthony *et al.* [32] present a useful background with their Tailored Logic Explained Networks, generating explainable sets of rules through the use of an accuracy score that is used to determine the similarity of these rules to the model predictions. In parallel, Švec *et al.* [33] present semantic representations that convert PE features into concepts that are more accessible by human analysts. However, a key gap exists, which is the lack of assertion that involve security professionals. This can be addressed with the help of controlled user studies that can take place in the Security Operations Center (SOC) environments. Those tests can evaluate both, quality of technical explanations (with metrics such as monotonicity, faithfulness and stability), and actual performance, such as the average screening time per alert, accuracy on new malware families, and a calibrated trust point. A clear success benchmark can be expressed. An XAI pipeline may be considered operationally validated when the addition of explanations lead to a significant statistical enhancement in the decision accuracy of analysts in comparison to settings where no explanations are offered, within a unified screening task.

3. **Continual and Distrubution-Aware Learning:** To jointly address gaps 1 and 4, future work should expand the distribution aware replay technique introduced in MADAR [41]. This method shows that choosing replay samples based on how good they represent the basic data distribution, instead of using a uniform random sampling, can result in a significant reduction in catastrophic forgetting. For example, MADAR achieves 85.8% average accuracy with a sample buffer of 20K, in comparison to 66.8% with prior methods on EMBER. The next practical move is to reduce the memory overhead of the replay buffer to make the method of value in edge deployment. This could be realized by knowledge distillation, in which, distributional information from the buffer is encoded into the model's parameters, or through parameter regularization strategies (like elast weight consolidation) that omit the need for an explicit buffer entirely. A well-defined evaluation criterion can be established. A continual learning system may be considered ready for production if it keeps average accuracy within 2 percentage points of full retraining, while working under a fixed memory limit (e.g. stored samples of no more than 10k) across at least four sequential temporal intervals on the EMBER2024 12 week test window [7].

4. **Adversarial-Robust Feature Engineering:** Addressing gap 5 (Adversarial Robustness), future research should focus on designing EMBER-compatible feature representations that are intrinsically strong against functionality-maintaining adversarial modifications. Previous research by Demetrio *et al.* [44] identifies typical evasion mechanisms like header padding, injection of sections and appending of overlays. While Aryal *et al.* [14] offer a more general classification of such attacks. A good research direction encircles 2 steps. To determine the vulnerability of one of the eight categories of features in EMBER, first, analyze the performance of detection in the isolated attack conditions of each category. Second, recode the most vulnerable features to more robust replacements, like invariants of call-graph hashes, or control flow-based hashes, and evaluate whether these replacements preserve detection quality with a better adversarial resistance to manipulation. Quantitative success benchmarks the quantitative success benchmark can be summarized as follows: On temporally partitioned test data, a set of feature can be said to be robust when its AUC (computed on clean data) remains at least 95% even to the most successful known attacks which do not impair functionality at different levels of disruption.

5. **Multi-Dataset Validation:** To counter Gap 6 (Dataset Monoculture), future work must add at least one additional benchmark to each EMBER based evaluation, including SOREL-20M [45], MOTIF or even VirusShare. The most convenient approach is cross dataset transfer evaluation: models are trained on one dataset (e.g. EMBER) and evaluated on another (e.g. SOREL-20M), and vice versa, and the difference in the AUC result is reported between the two performances in the same dataset and cross-dataset. Simulation tools like EMBERSim [10] facilitate this process by enabling nearest-neighbour matching analysis across datasets, enabling scientists to quantify distributional alignment, and find the malware families or feature regions do not decently generalize. A model may be considered generalizable when its AUC under cross dataset evaluation score, in at least two independently collected benchmarks it occurs that it achieves shows only a small decrease when compared with its performance on the same data, within a predefined range (say not exceeding 3 percentage points).

 ISSN: 2617-5517 | **DOI**: https://doi.org/10.59746/asgg6t42

6. **Lightweight Deployable Model**: The usage of models in edge and endpoint applications entails a delicate tradeoff between the high-detection capability and strict limitations on response time and memory consumption. To achieve these limitations, future studies must put the emphasis of priority to reduce high performing models, particularly the ensemble-based models that are emphasized in Section 5 to less resource consuming and efficient versions. A number of methods can be used to achieve this goal including lowering accuracy of gradient boosted decision trees (GBDTs), dropping redundant components inside ensemble pipelines, and knowledge distillation can be exploited optimally toward compressing multi model systems into simplified, single architectures. The improved models must be systematically compared to their original versions when the same performance is tested under a comparable temporal analysis background, and not only the AUC and detection rate are posted, but also the speed of inference, the storage consumption (in megabytes) and the highest utilization of memory. One pragmatic measure of achievement can be pursued as follows: a model can be said to be fit to be used in production settings when it retains at least 99% of the performance of the base model in terms of AUC at a given false positive rate (e.g. 0.1) and achieves a given latency target in parallel (e.g. not over 1 ms per sample on typical user equipment).

7. **LLM-Asssisted Analysis Pipelines:** Common approaches (or approaches propelled by LLM) are being developed, including PE2Prompt [22] and LLM-FS [23] propose a new set of practical research problems about hallucination control, prompt reliability, and the effectiveness of inference and the importance of Natural Langauge Explanation in Security Workflows. One of the gaps in the current research is lack of a robust assessment of explanation faithfulness. For example, PE2Prompt incorporates SHAP chosen features into prompts but does not confirm if the generated explanations are consistent with the underlying SHAP attributions, and none of both studies quantifies hallucination frequency. A solid advancing step is to develop the faithfulness evaluation framework that matches each LLM-produced explanation with a ground truth SHAP attribution vector. Agreement can then be evaluated using metrics such as rank correlation between features mentioned in the explanation and the top-k SHAP features, as well as the factual accuracy of the stated feature values. Evaluation thresholds can be clearly established as followed: an LLM-assisted pipeline may be considered dependable if its explanations exhibit a high level of alignment with SHAP attributions (e.g. spearman $\rho \geq 0.7$ for top 10 features) and if the rate of unverifiable or hallucinated feature claims remains below 5%, when evaluated on temporally held out samples of EMBER.

From the seven directions presented, the most critical is the integration of explainability and temporal evaluation within ensemble models of high performance. As shown in the comprehensive comparison in Table 6, prior studies have enhanced accuracy, interpretability, and temporal robustness in isolation, but none successfully integrated all three dimensions. combines all three. Bridging this gap would precisely address the central trade off described in this review, and mark a significant advancement toward deploying EMBER-based malware detection systems in real world contexts.

10. Conclusion

This review has synthesized 27 primary studies on static malware detection that are EMBER-based, published between years 2018 to early 2026, spread across five thematic clusters: traditional machine learning, deep learning, ensemble and hybrid architectures, explainable AI, and zero-day detection with concept drift. The work provides three contributions: an organized comparison of approaches and their results across all five clusters, a recognition of six research gaps that crosses all of them regardless of the method used, and seven future research directions each of which provided with a clear criterion for the purpose of progress measurement.

Three patterns constantly show up across the studies that were reviewed. First of which, gradient boosted models (particularly LightGBM) constantly produced the strongest classification performance numbers on EMBER, this is majorly due to tree-based methods ability to handle EMBER's large and type heterogeneous feature space. Second, ensemble and hybrid architectures report the best accuracy in total, although at the price of reduced explainability and increased computation resources, two tradeoffs that most studies address but do not resolve. Third, and most crucially, 22 of the 27 studies treated as primary used randomly selected train/test splits, instead of chronologically ordered splits that would guarantee more rigorous and generalizable performance results. This means that the performance reported using random splits would lead to reporting higher performance figures than what the built models could really achieve in a real-world production environment.

These findings come with practical consequences that are worth being mentioned. Security teams and vendors choosing models based on published accuracy should do so with caution, as it's possible that those figures do not hold once tested against new malware of a pattern it hasn't been trained on. Galen and Steele [46] displayed that performance can face severe drops within just six months when models are tested against data that was collected after the model's training window. Separately, fewer than three of the 27 primary studies reviewed reported false positive rates at low thresholds that real security systems need, this makes it difficult to decide whether any of these models could be used for dependable operation in production sets. On the side of explainable AI, the gap between what the researchers have built and what security analysts can actually put to use remain largely unaddressed, since no study of the reviewed tested its explanations with real analysts in a production environment.

The most notable gap this review highlights is that no study has yet mixed strong ensemble detection performance with built in explainability and time-based detection and evaluation on EMBER all within the same work. The foundation to build such framework is already available. EMBER 2024 provides both the standard split and the more demanding challenge temporal split, SHAP can be applied at ensemble level, and evaluation methods for explanation performance and quality have been developed in other fields already and are possible to be applied in this field. The only thing missing is a study that addresses all these points together and, (ideally) encourages other researchers to adopt the same method. That single step would result in moving this field toward practical, trustworthy and reliable malware detection systems in comparison to improvements limited to increasing accuracy on random data splits.

REFERENCES

[1] AV-TEST Institute, "Malware Statistics." AV-ATLAS. Accessed: Apr. 20, 2026. [Online]. Available: https://portal.av-atlas.org/malware/statistics

[2] J. Ferdous, R. Islam, A. Mahboubi, and Md. Z. Islam, "A Review of State-of-the-Art Malware Attack Trends and Defense Mechanisms," *IEEE Access*, vol. 11, pp. 121118–121141, 2023, doi: 10.1109/ACCESS.2023.3328351.

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

[4] N. Z. Gorment, A. Selamat, L. K. Cheng, and O. Krejcar, "Machine Learning Algorithm for Malware Detection: Taxonomy, Current Challenges, and Future Directions," *IEEE Access*, vol. 11, pp. 141045–141089, 2023, doi: 10.1109/ACCESS.2023.3256979.

[5] H.-D. Pham, T. D. Le, and T. N. Vu, "Static PE Malware Detection Using Gradient Boosting Decision Trees Algorithm," in *Future Data and Security Engineering*, T. K. Dang, J. Küng, R. Wagner, N. Thoai, and M. Takizawa, Eds., Cham: Springer International Publishing, 2018, pp. 228–236. doi: 10.1007/978-3-030-03192-3_17.

[6] N. A. Azeez, O. E. Odufuwa, S. Misra, J. Oluranti, and R. Damaševičius, "Windows PE Malware Detection Using Ensemble Learning," *Informatics*, vol. 8, no. 1, p. 10, Feb. 2021, doi: 10.3390/informatics8010010.

[7] R. J. Joyce *et al.*, "EMBER2024 - A Benchmark Dataset for Holistic Evaluation of Malware Classifiers," in *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, Toronto ON Canada: ACM, Aug. 2025, pp. 5516–5526. doi: 10.1145/3711896.3737431.

[8] H. S. Anderson and P. Roth, "EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models," Apr. 16, 2018, *arXiv*: arXiv:1804.04637. doi: 10.48550/arXiv.1804.04637.

[9] C. Galen and R. Steele, "Evaluating Performance Maintenance and Deterioration Over Time of Machine Learning-based Malware Detection Models on the EMBER PE Dataset," in *2020 Seventh International Conference on Social Networks Analysis, Management and Security (SNAMS)*, Dec. 2020, pp. 1–7. doi: 10.1109/SNAMS52053.2020.9336538.

[10] D. G. Corlatescu, A. Dinu, M. P. Gaman, and P. Sumedrea, "EMBERSim: A Large-Scale Databank for Boosting Similarity Search in Malware Analysis," *Adv. Neural Inf. Process. Syst.*, vol. 36, pp. 26722–26743, Dec. 2023.

[11] P. Švec, Š. Balogh, M. Homola, and J. Kľuka, "Knowledge-Based Dataset for Training PE Malware Detection Models," Dec. 31, 2022, *arXiv*: arXiv:2301.00153. doi: 10.48550/arXiv.2301.00153.

[12] D. Gibert, N. Totosis, C. Patsakis, Q. Le, and G. Zizzo, "Assessing the impact of packing on static machine learning-based malware detection and classification systems," *Comput. Secur.*, vol. 156, p. 104495, Sep. 2025, doi: 10.1016/j.cose.2025.104495.

[13] X. Ling *et al.*, "Adversarial attacks against Windows PE malware detection: A survey of the state-of-the-art," *Comput. Secur.*, vol. 128, p. 103134, May 2023, doi: 10.1016/j.cose.2023.103134.

[14] K. Aryal, M. Gupta, M. Abdelsalam, P. Kunwar, and B. Thuraisingham, "A Survey on Adversarial Attacks for Malware Analysis," *IEEE Access*, vol. 13, pp. 428–459, 2025, doi: 10.1109/ACCESS.2024.3519524.

[15] Y. Gao, H. Hasegawa, Y. Yamaguchi, and H. Shimada, "Malware Detection Using LightGBM With a Custom Logistic Loss Function," *IEEE Access*, vol. 10, pp. 47792–47804, 2022, doi: 10.1109/ACCESS.2022.3171912.

[16] Y. Oyama, T. Miyashita, and H. Kokubo, "Identifying Useful Features for Malware Detection in the Ember Dataset," in *2019 Seventh International Symposium on Computing and Networking Workshops (CANDARW)*, Nov. 2019, pp. 360–366. doi: 10.1109/CANDARW.2019.00069.

[17] O. Barut, T. Zhang, Y. Luo, and P. Li, "A Comprehensive Study on Efficient and Accurate Machine Learning-Based Malicious PE Detection," in *2023 IEEE 20th Consumer Communications & Networking Conference (CCNC)*, Jan. 2023, pp. 632–635. doi: 10.1109/CCNC51644.2023.10060214.

[18] N. Khan, A. Al-Tamimi, A. Bermak, and I. Khalil, "Adaptive malware detection using sequential feature selection: A dueling double deep Q-Network framework for intelligent classification," *J. Inf. Secur. Appl.*, vol. 99, p. 104407, Jun. 2026, doi: 10.1016/j.jisa.2026.104407.

[19] F. T. ALGorain and A. S. Alnaeem, "Deep Learning Optimisation of Static Malware Detection with Grid Search and Covering Arrays," *Telecom*, vol. 4, no. 2, pp. 249–264, May 2023, doi: 10.3390/telecom4020015.

[20] T.-H. Lai, Y.-J. Tsai, and C.-L. Liu, "Improving the Performance of Static Malware Classification Using Deep Learning Models and Feature Reduction Strategies," *Mathematics*, vol. 13, no. 23, p. 3753, Nov. 2025, doi: 10.3390/math13233753.

[21] A. Brown, M. Gupta, and M. Abdelsalam, "Automated machine learning for deep learning based malware detection," *Comput. Secur.*, vol. 137, p. 103582, Feb. 2024, doi: 10.1016/j.cose.2023.103582.
[22] Y. Sun and M. Masum, "PE2Prompt: A Large Language Model-Native Framework for Interpretable Static Malware Analysis," Dec. 17, 2025, *Social Science Research Network, Rochester, NY*: 5933533. doi: 10.2139/ssrn.5933533.
[23] N. Gill, A. H. K, and S. D. M. Kumar, "LLM-FS: Zero-Shot Feature Selection for Effective and Interpretable Malware Detection," in *2025 Conference on Building a Secure & Empowered Cyberspace (BuildSEC)*, Dec. 2025, pp. 91–99. doi: 10.1109/BuildSEC68439.2025.00022.
[24] R. Damaševičius, A. Venčkauskas, J. Toldinas, and Š. Grigaliūnas, "Ensemble-Based Classification Using Neural Networks and Machine Learning Models for Windows PE Malware Detection," *Electronics*, vol. 10, no. 4, p. 485, Feb. 2021, doi: 10.3390/electronics10040485.
[25] D. Trizna, "Quo Vadis: Hybrid Machine Learning Meta-Model Based on Contextual and Behavioral Malware Representations," in *Proceedings of the 15th ACM Workshop on Artificial Intelligence and Security*, Los Angeles CA USA: ACM, Nov. 2022, pp. 127–136. doi: 10.1145/3560830.3563726.
[26] N. Vuran Sarı and M. Acı, "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.
[27] K. Mahmud Sujon, R. Binti Hassan, M. Abdullah-Al-Wadud, and J. Uddin, "OPTISTACK: A Hybrid Ensemble Learning and XAI-Based Approach for Malware Detection in Compressed Files," *IEEE Access*, vol. 13, pp. 104992–105026, 2025, doi: 10.1109/ACCESS.2025.3579880.
[28] A. Al Siam, H. Abu-Adaiq, M. Nafis, F. A. Anik, S. Mahmud, and M. M. Hassan, "Explainable Machine Learning for Malware Detection: A SHAP-Based LightGBM Framework," in *2026 IEEE 5th International Conference on AI in Cybersecurity (ICAIC)*, Feb. 2026, pp. 1–6. doi: 10.1109/ICAIC67076.2026.11395690.
[29] R. Kumar and G. Subbiah, "Zero-Day Malware Detection and Effective Malware Analysis Using Shapley Ensemble Boosting and Bagging Approach," *Sensors*, vol. 22, no. 7, p. 2798, Apr. 2022, doi: 10.3390/s22072798.
[30] H. Manthena, J. C. Kimmel, M. Abdelsalam, and M. Gupta, "Analyzing and Explaining Black-Box Models for Online Malware Detection," *IEEE Access*, vol. 11, pp. 25237–25252, 2023, doi: 10.1109/ACCESS.2023.3255176.
[31] E. Baghirov, "A comprehensive investigation into robust malware detection with explainable AI," *Cyber Secur. Appl.*, vol. 3, p. 100072, Dec. 2025, doi: 10.1016/j.csa.2024.100072.
[32] P. Anthony *et al.*, "Explainable Malware Detection with Tailored Logic Explained Networks," May 05, 2024, *arXiv*: arXiv:2405.03009. doi: 10.48550/arXiv.2405.03009.
[33] P. Švec, Š. Balogh, M. Homola, J. Kľuka, T. Bisták, and P. Anthony, "Semantic Data Representation for Explainable Windows Malware Detection Models," 2024, *arXiv*. doi: 10.48550/ARXIV.2403.11669.
[34] H. Manthena, S. Shajarian, J. C. Kimmell, M. Abdelsalam, S. Khorsandroo, and M. Gupta, "Explainable Artificial Intelligence (XAI) for Malware Analysis: A Survey of Techniques, Applications, and Open Challenges," *IEEE Access*, vol. 13, pp. 61611–61640, 2025, doi: 10.1109/ACCESS.2025.3555926.
[35] M. Saqib, S. Mahdavifar, B. C. M. Fung, and P. Charland, "A Comprehensive Analysis of Explainable AI for Malware Hunting," *ACM Comput Surv*, vol. 56, no. 12, p. 314:1-314:40, Oct. 2024, doi: 10.1145/3677374.
[36] E. M. Rudd, D. Krisiloff, S. Coull, D. Olszewski, E. Raff, and J. Holt, "Efficient Malware Analysis Using Metric Embeddings," *Digit. Threats Res. Pract.*, vol. 5, no. 1, pp. 1–20, Mar. 2024, doi: 10.1145/3615669.
[37] O. Jurečková, M. Jureček, M. Stamp, F. Di Troia, and R. Lórencz, "Classification and online clustering of zero-day malware," *J. Comput. Virol. Hacking Tech.*, vol. 20, no. 4, pp. 579–592, Nov. 2024, doi: 10.1007/s11416-024-00513-5.
[38] W. Maillet and B. Marais, "Optimized Deep Learning Models for Malware Detection under Concept Drift," Aug. 01, 2024, *arXiv*: arXiv:2308.10821. doi: 10.48550/arXiv.2308.10821.
[39] A. Manikandaraja, P. Aaby, and N. Pitropakis, "Rapidrift: Elementary Techniques to Improve Machine Learning-Based Malware Detection," *Computers*, vol. 12, no. 10, Sep. 2023, doi: 10.3390/computers12100195.
[40] M. S. Rahman, S. E. Coull, and M. Wright, "On the Limitations of Continual Learning for Malware Classification," Aug. 13, 2022, *arXiv*: arXiv:2208.06568. doi: 10.48550/arXiv.2208.06568.
[41] M. S. Rahman, S. Coull, Q. Yu, and M. Wright, "MADAR: Efficient Continual Learning for Malware Analysis with Distribution-Aware Replay," Sep. 17, 2025, *arXiv*: arXiv:2502.05760. doi: 10.48550/arXiv.2502.05760.

[42] L. Demetrio, B. Biggio, G. Lagorio, F. Roli, and A. Armando, "Functionality-Preserving Black-Box Optimization of Adversarial Windows Malware," *IEEE Trans. Inf. Forensics Secur.*, vol. 16, pp. 3469–3478, 2021, doi: 10.1109/TIFS.2021.3082330.

[43] L. Demetrio, B. Biggio, G. Lagorio, F. Roli, and A. Armando, "Explaining Vulnerabilities of Deep Learning to Adversarial Malware Binaries," Jan. 24, 2019, *arXiv*: arXiv:1901.03583. doi: 10.48550/arXiv.1901.03583.

[44] L. Demetrio, S. E. Coull, B. Biggio, G. Lagorio, A. Armando, and F. Roli, "Adversarial EXEmples: A Survey and Experimental Evaluation of Practical Attacks on Machine Learning for Windows Malware Detection," *ACM Trans Priv Secur*, vol. 24, no. 4, p. 27:1-27:31, Sep. 2021, doi: 10.1145/3473039.

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

[46] C. Galen and R. Steele, "Evaluating Performance Maintenance and Deterioration Over Time of Machine Learning-based Malware Detection Models on the EMBER PE Dataset," in *2020 Seventh International Conference on Social Networks Analysis, Management and Security (SNAMS)*, Dec. 2020, pp. 1–7. doi: 10.1109/SNAMS52053.2020.9336538.