ISSN:2584-2706 DOI: https://doi.org/10.5281/zenodo.22025661

Machine Learning-Based Malware Detection through Static, Dynamic, and Hybrid Analysis

MD Shakhawat Hossain
CSE, World University of Bangladesh

Abstract:
Machine learning-based malware detection addresses a central limitation of signature-driven security: malicious programs can alter their syntax, packaging, and execution traces faster than manually authored rules can be updated. This review examines how static and dynamic analysis generate complementary evidence for binary detection, family classification, variant recognition, and first-time-appeared malware detection. Static analysis extracts information without executing a file, including metadata, imported functions, byte patterns, opcodes, and assembly-derived representations. Dynamic analysis observes execution in an isolated environment through API calls, system calls, runtime opcodes, and other behavioral traces. Static methods offer low latency and high throughput, whereas dynamic methods can expose behavior concealed by packing and code obfuscation, although they require controlled execution and cannot observe behavior that the sandbox fails to trigger.
The literature increasingly evaluates **hybrid analysis**, which combines code-level and behavioral evidence rather than treating the two views as independent alternatives. Integrated static and dynamic features improved reported accuracy over either individual view in several studies, while correlated fusion of static and dynamic API sequences produced detection and classification accuracies of 97.89% and 94.39%, respectively, in MalDAE. **Classical machine learning** remains useful where features are compact, interpretable, and computationally constrained. **Deep learning** reduces manual feature design by learning representations from bytes, images, opcode sequences, API traces, and heterogeneous feature spaces, but reported performance is highly sensitive to dataset construction and temporal separation.
The synthesis therefore treats accuracy as one dimension of detector quality. Computational cost, execution coverage, family and platform scope, zero-day evaluation, explainability, adversarial robustness, and resistance to **concept drift** are equally relevant to deployment. Chronologically organized benchmarks show that malware families, packers, and evolved samples induce distinct forms of distribution change, while adversarial studies demonstrate that evasion, poisoning, and backdoors can invalidate apparently strong classifiers. The review compares these evidence sources and learning strategies to identify when feature fusion, adaptive updating, and explicit robustness testing are justified.

Keywords: Malware Detection; Static Analysis; Dynamic Analysis; Hybrid Analysis; Machine Learning; Deep Learning; Adversarial Robustness; Concept Drift

ISSN:2584-2706 DOI: https://doi.org/10.5281/zenodo.22025661

Introduction:

Malware detection now operates against programs that can conceal code statically, delay execution dynamically, alter their features across variants, and appear in volumes that exceed manual analysis capacity. **Static analysis** is attractive for screening because a file need not be executed, but packing, encryption, and obfuscation can remove or distort the signals on which a classifier depends [1]. **Dynamic analysis** can recover runtime behavior and is therefore more tolerant of syntactic changes, yet sandbox execution is costly, environment-dependent, and incomplete because malware may stall, require unavailable resources, or follow an unobserved path [1]. A detector trained on either view may consequently fail when its evidence source is manipulated or when future samples differ from historical data. The research problem is thus to learn representations that preserve malicious semantics across changing syntax while controlling analysis latency and false positives. Static and dynamic evidence can constrain different failure modes: code features support rapid triage, behavioral traces expose actions after unpacking, and their correlation can connect apparent syntactic differences to common program meaning [2], [3]. This multiview objective also changes evaluation: a model must be tested on temporally separated, packed, obfuscated, and previously unseen samples rather than only on randomly divided observations.

Literature Review: Analytical Foundations and Feature Representations in Malware Research

Static Analysis Features, Program Semantics, and Low-Parameter Models

Static analysis represents a program through properties extracted without execution. Common representations in the reviewed literature include imported functions, metadata, byte sequences, opcode n-grams, assembly instructions, grayscale or coloured images of executable files, and manually designed feature vectors [4], [5]. These representations support both binary discrimination and family classification. Their main advantage is operational: feature extraction can precede execution, making static screening suitable for large repositories and early-stage filtering. Low-parameter models are relevant when memory and inference resources are limited. An evaluation of artificial neural networks, support vector machines, and gradient boosting machines for static metadata classification reported 93.44% accuracy for the neural network under extreme memory constraints [6]. That result supports the feasibility of resource-aware static detection, but it does not establish performance across platforms or against adaptive packing.

Static features often encode syntax rather than executable intent. Packers can transform the observable file while preserving the underlying payload, and benign software may also use packing or obfuscation for intellectual-property protection or license control [7]. Consequently, a model may learn packer-specific correlations that separate the training data but fail when packers change or occur in both benign and malicious files. BenchMFC makes this problem explicit by tagging family, packer, and timestamp information and separating unseen, packed, and evolved-family drift [8]. Its case study reports that packers preserve signals that can appear effective to static models, while the robustness of those signals remains unresolved [8]. Static learning therefore requires a distinction between stable semantic indicators and incidental artifacts of collection or packaging.

Program semantics offers one route beyond surface syntax. Dynamic system-call detectors can be disrupted by injected calls that alter observed sequences without changing the malicious objective. A semantics-oriented method used information-theoretic characterization to extract information-rich call sequences, reducing exposure to direct call-injection manipulation [9]. The broader implication is that feature design should represent relations among actions, rather than treating every token as equally informative. Such representations may improve interpretability and resistance, but their extraction assumptions and computational requirements must be measured alongside accuracy.

Dynamic Behavioral Features, API Calls, Opcodes, and Controlled Execution

Dynamic analysis executes a sample in an isolated environment and records its runtime behavior. API calls are a dominant feature source because malware relies on operating-system interfaces to create processes, modify files, communicate, and perform other actions. Generalizing raw traces into higher-level actions produced useful behavior-based features for detecting variants whose syntactic structures differ [10]. Dynamic analysis can also record system calls and runtime opcodes. Cryptomining research, for example, compared static opcode sequences with dynamically captured system-call

events, obtaining 95% static accuracy and 99% dynamic accuracy for its study task [11]. These results concern one specialized threat and dataset, so they should not be interpreted as universal superiority of dynamic evidence.[12], [13]

Sequence structure matters because malicious behavior is often expressed through ordered actions rather than isolated tokens. A dynamic architecture that hashes API arguments, applies gated convolutional networks to individual calls, and uses bidirectional recurrent layers for call-sequence correlations addresses information lost when models retain API names but discard arguments [14]. Runtime opcode analysis similarly requires attention to data growth and retraining cost, not only classification accuracy. A cost study comparing 23 algorithms on an expanding run-trace corpus found parallelized random forest with four cores to offer the preferred balance of accuracy and training and testing time in that setting [15]. Dynamic evidence thus increases behavioral sensitivity, but it also increases dependence on trace collection, storage, execution scheduling, and model-update resources.[16], [17]

Controlled execution introduces a coverage problem. A sandbox observes only the paths and environmental conditions reached during analysis, and dynamic malware analysis surveys report that no single tool covers every aspect of behavior [1]. Malware may detect virtualization, delay harmful actions, require user interaction, or activate only after a particular network response. Dynamic analysis is therefore more resistant to several static evasion techniques without being immune to evasion. The appropriate comparison is not whether dynamic data are inherently better, but whether the additional behavioral information justifies execution cost for the decision being made.

Feature Engineering, Feature Selection, and Deep Representation Learning

Feature engineering determines what a classifier can observe, while feature selection controls redundancy, dimensionality, and computation. Traditional workflows rely on expert choices about imports, calls, n-grams, metadata, and behavior categories; surveys identify feature construction and dataset design as persistent determinants of malware-model performance [18], [19]. A study using PCA, linear discriminant analysis, scaling, and twelve classifiers found that LightGBM reached 97.16% accuracy with PCA and either min-max scaling or normalization on a tabular dataset of 11,598 samples and 139 features [20]. The finding supports evaluating preprocessing as part of the model, although the binary tabular setting limits direct comparison with sequence and execution-based detectors.

Evolutionary selection can model interactions that univariate filtering misses. A multi-objective genetic approach reduced a behavior-derived feature set from 335 to 200 features and reported 93.33% peak accuracy, with a reported 40% performance enhancement after reduction [21]. The study used Cuckoo Sandbox data and several classifiers, illustrating how feature selection can reduce analysis and inference burden. Yet selection performed before a strictly separated test procedure is necessary to avoid optimistic estimates. Feature selection must also be reassessed under temporal drift because a feature that identifies historical families may encode a transient collection artifact.

Deep representation learning changes the division between feature construction and classification. DeepAM used API-call data with an autoencoder, stacked restricted Boltzmann machines, and associative memory, pretraining on labelled and unlabelled files before supervised fine-tuning [22]. Other systems transform executable bytes into images, learn visual features, and pass them to conventional classifiers; one such framework combined deep features with SVM and reported 99.06% accuracy on Malimg [23]. These results show that deep models can learn useful representations from raw or lightly processed inputs, but high benchmark accuracy does not by itself establish semantic understanding, cross-dataset transfer, or resistance to adversarial modification. Fusion of expert and learned features remains attractive because assembly, bytes, images, and structural entropy can encode partially distinct information [24].

Methods: Evidence Synthesis and Comparative Evaluation Framework

Corpus Organization Across Static, Dynamic, and Hybrid Detection Studies

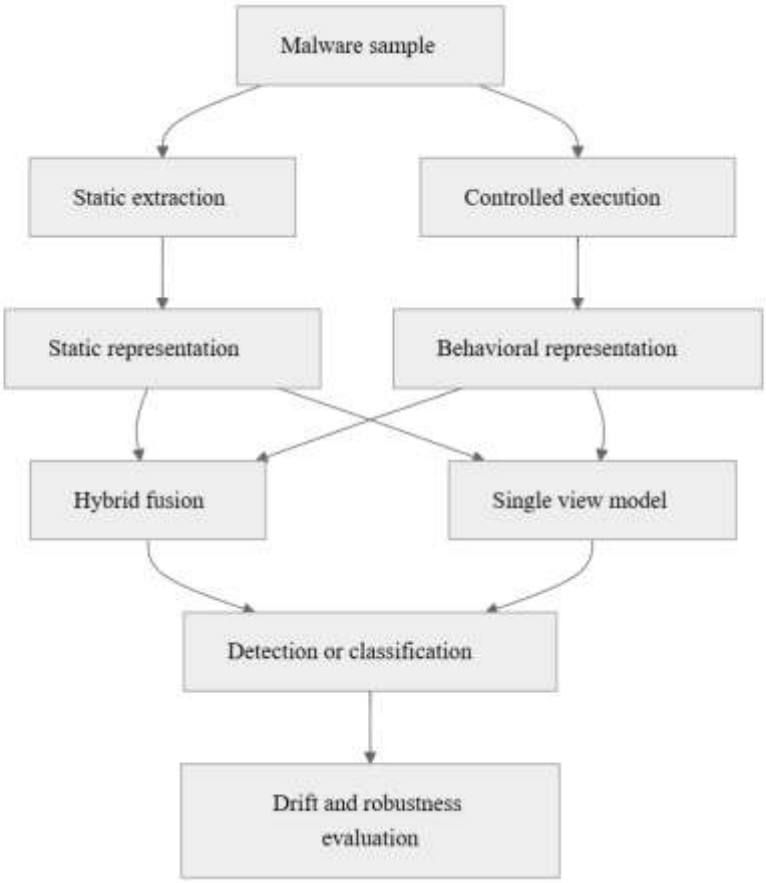
The evidence was organized by analysis view, learning strategy, task, and deployment concern. Static studies were grouped when they used executable metadata, bytes, imports, opcodes, assembly, or image encodings without requiring sample execution. Dynamic studies were grouped when they collected API calls, system calls, runtime opcodes, network behavior, or sandbox traces. Hybrid

ISSN:2584-2706 DOI: https://doi.org/10.5281/zenodo.22025661

studies were retained as a separate category when they concatenated, fused, or jointly modelled static and dynamic evidence, since their computational requirements and failure modes differ from those of either standalone view.

A second organization separated binary detection from family classification, new-family discovery, first-time-appeared malware recognition, and specialized-threat detection. A third recorded whether evaluation used random, temporally separated, packed, obfuscated, or adversarial samples. This distinction follows evidence that random splits can conceal temporal degradation and that packers create a distinct source of distribution change [8], [25]. Reported accuracy values were retained with their task and dataset context rather than treated as directly interchangeable.

The synthesis framework can be represented as follows:



Comparison Dimensions: Detection, Family Classification, Zero-Day Recognition, and Computational Cost

The primary comparison dimensions were detection effectiveness, family classification, recognition of unseen or first-time-appeared samples, and computational cost. Detection concerns malware-versus-benign discrimination, whereas family classification concerns attribution among malicious groups; a method can perform well on one and poorly on the other. Zero-day claims were interpreted narrowly, requiring temporal, family-disjoint, anomaly-based, or explicitly first-time-appeared evaluation. Deep-learning surveys distinguish unsupervised, semi-supervised, few-shot, and adversarial-resistant approaches to zero-day detection, confirming that "zero-day" is not a single experimental condition [26].

Cost was assessed conceptually through extraction latency, sandbox execution, memory, inference, retraining, and storage. Dynamic runtime-opcode research directly examined computational cost under expanding datasets [15]. Static low-parameter work addressed memory constraints [6], while adaptive and federated studies addressed update frequency, annotation, and training-set growth [27],

ISSN:2584-2706 DOI: https://doi.org/10.5281/zenodo.22025661

[28]. These dimensions prevent a high accuracy obtained through expensive offline execution from being compared uncritically with a low-latency static screen.

Evaluation Concerns: Dataset Scope, Feature Leakage, Generalization, and Reproducibility

Evaluation quality depends first on dataset scope. Public benchmarks often concentrate on one platform, one task, or one period, while malware families may share near-duplicate code and labels may reflect vendor or packer decisions. EMBER2024 responds to this limitation with more than 3.2 million files across six formats, labels for seven classification tasks, and a challenge set containing malicious files initially missed by selected antivirus products [29]. BenchMFC similarly supplies 223,000 samples from 526 evolving families with timestamps and packer tags [8]. These resources support stronger temporal and evasive-sample evaluation than undifferentiated random partitions.[30], [31]

Feature leakage can arise when near-duplicates cross training and test sets, when selection uses the test partition, or when packer and family artifacts provide shortcuts. Temporal separation and family-disjoint testing are therefore more informative for generalization than a single random split, as shown by work that explicitly removed dataset bias through disjoint timescale splits [25]. Reproducibility also requires reporting extraction tools, sandbox conditions, preprocessing, class balance, thresholds, update policies, and hardware. Without these details, nominal accuracy cannot establish operational equivalence across studies, and the comparative results below are interpreted as conditional findings rather than a universal ranking.

Results I: Comparative Performance of Static, Dynamic, and Hybrid Malware Detection

Static Analysis: Speed, Scalability, and Vulnerability to Packing and Obfuscation

Static analysis consistently offers the lower operational barrier because it avoids executing the suspicious file. Metadata, imports, bytes, and opcode or image representations can be extracted before a sample enters a sandbox, supporting high-volume triage and reducing exposure to active malware during analysis [4]. Low-parameter models extend this advantage to constrained environments: a static study found that an ANN achieved 93.44% accuracy despite extreme memory constraints [6]. Static representations also support several model families, including SVM, gradient boosting, neural networks, and ensemble classifiers, so deployment can trade representational capacity against memory and latency.

The limitation is that static evidence is vulnerable when observable syntax is decoupled from malicious behavior. Packing, encryption, code morphing, and obfuscation can change bytes, imports, or instruction patterns without eliminating the payload. The dynamic-analysis survey characterizes dynamic methods as more resistant to these transformations, whereas static methods remain exposed to them [1]. A study of static classifiers further emphasizes that benign applications may also be packed or obfuscated, making packer detection an ambiguous feature rather than a reliable maliciousness indicator [7]. Static models can therefore learn useful packer signals while confusing implementation style with threat status.

The problem is clearest under temporal evaluation. BenchMFC distinguishes unseen families, packed families, and evolved families and reports that packers preserve signals that appear effective for static machine learning, while the durability of those signals requires further study [8]. This finding reframes packing from a simple preprocessing nuisance to a source of concept drift. A model may retain high accuracy on samples produced by familiar packers yet fail when a new packer changes the feature distribution. Static detection is consequently strongest as a rapid first decision when its training corpus spans relevant packers, periods, and benign software practices.

Static representations can still support unseen-malware recognition when the learning task is designed around novelty rather than closed-set family labels. A framework that converted PE files into coloured images, extracted deep features, selected influential dimensions, and applied a one-class classifier reported 99.30% accuracy on Malimg while targeting evolving, polymorphic, and zero-day samples [32]. The result is promising but dataset-specific, and the reported benchmark does not demonstrate that all future malware lies outside the learned benign boundary. Static methods therefore provide scalable evidence, but their claims require challenge sets and transformations that were absent from training.

ISSN:2584-2706 DOI: https://doi.org/10.5281/zenodo.22025661

Dynamic Analysis: Behavioral Coverage, Zero-Day Potential, and Sandbox Dependence

Dynamic analysis gains its principal advantage from observing what a program does after execution rather than inferring behavior solely from its stored representation. API traces can be generalized into actions, allowing classifiers to recognize variants with different syntax but similar behavior [10]. Dynamic analysis also exposes code after unpacking and can retain discriminative information when static features have been altered. In a comparative HMM study across malware families, the fully dynamic approach generally achieved the best detection rates among the evaluated static, dynamic, and hybrid configurations [33]. That result supports the behavioral view, although it reflects the particular HMM features and execution protocol used in the study.

Behavioral sequences become more informative when arguments and order are retained. API-name-only representations discard parameters that may distinguish benign file access from malicious persistence or communication. A feature-hashing method encoded API arguments and used gated CNNs followed by bidirectional LSTMs to learn local call features and sequential correlations; experiments on a large real dataset reported superiority over its baselines [14]. Semantic compression offers another response to trace manipulation. The AEP-based method extracted information-rich call sequences to reduce vulnerability to system-call injection, because the model did not depend directly on every observable call [9].

Dynamic evidence has a stronger connection to zero-day potential than to guaranteed zero-day detection. Early behavior-based work used clustering and classification to discover novel behavioral classes and assign unknown binaries to those classes, with an incremental process intended to handle thousands of binaries daily [34]. A later adaptive behavioral model sorted samples by first appearance, detected concept drift, and incrementally introduced new data with historical samples to limit catastrophic forgetting. It reported 99.41% average detection accuracy for new and variant malware and an average update frequency of 1.35 times per month [27]. These findings suggest that behavioral models can adapt to changing variants when chronology is built into training and updating, rather than assuming a stationary feature-label relationship.

Execution remains the central constraint. Dynamic collection requires an isolated environment, and the malware must execute sufficiently to reveal its behavior [35]. Sandboxes may miss delayed, environment-sensitive, user-triggered, or network-dependent actions, while execution increases processing time and infrastructure demands. Runtime-opcode research demonstrates why cost must be measured over growing corpora: fixed-size experiments do not show how retraining scales as new traces accumulate [15]. Dynamic analysis therefore offers richer evidence and often stronger resistance to syntactic evasion, but its coverage and cost determine whether the evidence is available before a security decision.

Hybrid Fusion: Correlating Static and Dynamic Characteristics for More Complete Detection

Hybrid analysis addresses complementary blind spots by combining stored code evidence with observed behavior. The rationale is strongest when an attacker can disguise a sample in one view but cannot preserve malicious intent while consistently disguising every view. Multiple-kernel learning formalized this idea by assigning separate similarity measures to static and dynamic views and learning their weighted combination in an SVM classifier [3]. A simpler integrated classifier combining static and dynamic features reduced the time required to test the features separately by half and showed greater robustness across malware collections from different periods [35]. These studies differ in architecture and task, but both treat the views as complementary information sources rather than redundant measurements.

Reported accuracy comparisons generally favor fusion, although the magnitude depends on data and implementation. One integrated-analysis study reported 95.8% accuracy for static features, 97.1% for dynamic features, and 98.7% for the integrated method [36]. MalDAE moved beyond concatenation by mapping static and dynamic API sequences according to semantic relationships, producing a hybrid feature space and explanations based on malicious behavior types. Its reported detection and classification accuracies reached 97.89% and 94.39%, respectively [2]. The claims are not directly comparable because the datasets, labels, and evaluation protocols differ, but they consistently indicate that correlated views can add information.

The relation between static and dynamic traces is especially useful when their syntax differs but their semantics converge. A static sequence may contain calls that are never executed, while a dynamic

ISSN:2584-2706 DOI: https://doi.org/10.5281/zenodo.22025661

sequence may include calls introduced through unpacking or conditional paths. Mapping both to behavior-level representations can distinguish potential capability from realized action and can support explanations that a black-box byte model cannot readily provide [2]. Hybrid fusion also permits staged deployment: static features can triage all files, and dynamic execution can be reserved for uncertain or high-risk cases. Such staging is an operational inference from the complementary costs, rather than a directly tested universal architecture.[37], [38]

Fusion does not remove error sources. If both views inherit the same family label leakage, duplicate samples, or packer artifact, their combination may amplify rather than correct the shortcut. It also requires synchronization between static extraction and runtime collection, increasing storage and engineering demands. The comparison is summarized below.

Analysis view	Main evidence	Principal strength	Principal limitation	Representative result
Static	Metadata, bytes, imports, opcodes, images	Rapid screening and low execution cost	Packing and obfuscation can alter observable features [7]	93.44% under memory constraints [6]
Dynamic	API calls, system calls, runtime opcodes	Behavioral information after execution	Sandbox dependence and retraining cost [15]	Fully dynamic HMM generally led the comparison [33]
Hybrid	Correlated static and dynamic features	Complementary evidence and explanation	Higher pipeline complexity	98.7% integrated accuracy in one study [36]

The quantitative pattern is illustrative rather than a pooled estimate because the studies use different corpora and endpoints.

{"type":"bar","title":"Reported accuracy in selected malware detection studies","xLabel":"Study condition","yLabel":"Accuracy(%)","data":[{"x":"Staticintegrated-analysis study","y":95.8},{"x":"Dynamic integrated-analysis study","y":97.1},{"x":"Hybrid integrated-analysis study","y":98.7},{"x":"Static cryptomining study","y":95},{"x":"Dynamic cryptomining study","y":99}]}

Hybrid methods therefore offer the most complete evidence when execution resources and feature alignment are available. Their value should be tested under temporally separated and adversarial conditions, where an apparent gain from feature concatenation can be distinguished from genuine cross-view generalization.

Results II: Machine Learning and Deep Learning Strategies for Detection and Classification

Classical Machine Learning for Binary Detection and Malware Family Classification

Classical machine learning remains competitive when representations are compact and the task is well specified. Decision trees, random forests, support vector machines, boosting methods, nearest-neighbor models, naïve Bayes, and linear methods have been applied to static metadata, API behavior, network traffic, and engineered sequence features [4]. Their advantages include lower training complexity, clearer feature attribution in some models, and easier deployment on constrained systems. A study of Android static features found high detection accuracy using only a small subset of features, supporting the practical value of feature reduction rather than indiscriminate expansion [39]. Classical models also provide useful baselines against which deep architectures should be evaluated.

ISSN:2584-2706 DOI: https://doi.org/10.5281/zenodo.22025661

Ensemble methods often perform well on tabular or sparse behavioral data. In one feature-selection and scaling comparison, LGBM achieved 97.16% accuracy after PCA and scaling, and ensemble methods consistently exceeded the evaluated traditional alternatives [20]. A dynamic behavior study reported 100% accuracy for random forest, stochastic gradient descent, extra trees, and Gaussian naïve Bayes on its test data [40]. Such perfect test results should be interpreted cautiously because the abstract does not establish temporal separation, family disjointness, or resistance to duplicates. The contrast between high random-split scores and weaker drift-aware evaluations demonstrates why classifier rankings cannot be separated from corpus design.

Family classification introduces a different requirement: models must preserve similarities within a family while separating related families. Feature grouping and per-class weighting produced approximately 0.998 accuracy on the Microsoft Malware Classification Challenge dataset [41]. A system combining grayscale images, opcode n-grams, and imports classified unknown malware with a best accuracy of 98.9% and detected 86.7% of new malware in its evaluation [5]. These results indicate that engineered representations can support both known-family attribution and new-family discovery, but a closed family benchmark may reward memorization of family-specific artifacts. Clustering or one-class formulations are better aligned with novelty, provided they are assessed chronologically.

Deep Learning, Autoencoders, GANs, and Heterogeneous Feature Learning

Deep learning reduces reliance on manually selected features by learning transformations jointly with prediction. It has been applied to bytes, executable images, opcodes, API calls, and heterogeneous inputs, with CNNs, recurrent networks, autoencoders, restricted Boltzmann machines, and attention mechanisms appearing across the literature [42]. DeepAM used unsupervised pretraining on labelled and unlabelled files before supervised fine-tuning, allowing representation learning to use information beyond the labelled subset [22]. This design is relevant where labels are scarce, although unlabelled data can also carry collection bias.

Sequence models are suited to dynamic traces because they can represent local call patterns and longer dependencies. The gated-CNN and bidirectional-LSTM architecture for API calls and arguments outperformed its baselines in a large real dataset [14]. For cryptomining, CNN, LSTM, and attention-based LSTM models classified static opcode and dynamic system-call sequences, with the dynamic condition reaching 99% accuracy compared with 95% for static analysis [11]. The specialized threat and controlled corpus limit generalization, but the comparison shows how the same architecture family can process distinct sequential evidence.

Heterogeneous models combine learned representations from different formats. One approach extracted deep n-gram-like features from assembly and bytes, texture and shapelet features from grayscale images and structural entropy, then fused them with expert-designed features in a gradient-boosting model; it outperformed the compared gradient-boosting and deep-learning methods on the Microsoft benchmark [24]. Image-based PE representations also produced 99.06% accuracy on Malimg when deep features were passed to SVM [23]. These results favor hybrid representation learning, but image similarity may reflect layout or compiler artifacts rather than executable semantics. Explanations and cross-platform validity therefore remain open evaluation requirements.

Autoencoders and GANs have also been used to address novelty and robustness. A transferred deep-convolutional GAN trained with deep-autoencoder representations achieved 95.74% average classification accuracy and was reported as the most robust among compared models against modelled zero-day attacks [43]. GAN-generated adversarial examples can enrich training, but GAN quality, mode collapse, and transferability affect the resulting defense. LSGAN-AT was designed to generate smoother adversarial examples and improve adversarial training against black-box attacks [44]. These methods expand the training distribution, yet generated samples cannot replace evaluation on malware that evolves through real development and deployment processes.

Automated Model Selection, Feature Scaling, and Evolutionary Feature Selection

Model selection and preprocessing can materially change reported performance. AutoML addresses neural architecture search and hyperparameter optimization for static and online detection, reducing manual trial-and-error. Experiments on SOREL-20M, EMBER-2018, and an online cloud dataset found AutoML models to be at least comparable to, and in some cases better than, hand-designed

alternatives with limited architecture-search overhead [45]. This finding supports automated selection when datasets and deployment conditions are heterogeneous. It does not eliminate the need to constrain search by latency, memory, explainability, and temporal validation.

Scaling and dimensionality reduction should be fitted within the training partition. PCA and normalization can improve optimization for tabular models, while evolutionary selection can reduce features and expose interactions among behavior variables [20], [21]. The benefit is therefore both statistical and operational: fewer features can reduce memory, extraction time, and update cost. Yet selection may become unstable under drift, and features selected on historical families may fail on new packers or platforms. A credible comparison should report selection stability across time, not only the best held-out score.

The strongest strategy in this evidence base is conditional rather than universal. Classical models suit compact, interpretable, and resource-limited representations; deep models suit raw sequences and heterogeneous inputs; automated search can reduce architecture dependence; and evolutionary selection can control feature burden. Each strategy still inherits the validity of its split, labels, execution traces, and threat model. Results on generalization must therefore determine whether these methods learn durable malware properties or merely optimize a stationary benchmark.

Results III: Generalization to Malware Variants, Zero-Day Samples, and Specialized Threats

Detecting Previously Unseen and First-Time-Appeared Malware

Generalization is tested most directly when samples are separated by time, family, or first appearance. Randomly held-out samples can share family structure and collection artifacts with training data, producing an optimistic estimate of future detection. A deep-learning evaluation addressed this concern by using disjoint dataset splits across different timescales and reported that its visual and hybrid architectures outperformed classical approaches in zero-day experiments [25]. The study supports deep representations under its protocol, while also demonstrating that dataset bias must be removed before comparing architectures.

One-class learning changes the objective from recognizing known malicious classes to modelling benign space. A PE-image framework extracted deep features, selected influential dimensions, and enclosed benign data with a one-class classifier; samples outside that boundary were treated as anomalous, and the method reported 99.30% accuracy on Malimg [32]. This design reduces dependence on complete malware-family labels and can address evolving threats, but its performance depends on how broadly benign software is represented. A boundary learned from one platform or software ecosystem may reject legitimate future programs or miss malware that resembles benign files.[46], [47]

Unsupervised and generative methods provide additional routes. Early incremental clustering discovered novel behavioral classes and assigned later samples to them, supporting analysis of previously unlabelled malware [34]. The transferred GAN and deep-autoencoder approach generated modified malware-like features and achieved 95.74% average accuracy against modelled zero-day attacks [43]. Reviews classify zero-day methods into unsupervised, semi-supervised, few-shot, and adversarial-resistant groups, indicating that the term covers different assumptions about labels and novelty [26]. Results should therefore identify whether “unseen” means a new sample from a known family, a new family, a future time period, or an adversarially modified file.

First-time appearance can also trigger adaptive updating. The AIBL-MVD system ordered behavioral samples by first appearance, used statistical process control for drift detection, and incrementally mixed new data with historical samples to reduce catastrophic forgetting. It reported 99.41% average accuracy for new and variant malware with updates averaging 1.35 times per month [27]. This is evidence for chronological adaptation, not proof that a fixed model can detect all zero-day samples. The operational question becomes how rapidly a detector can identify uncertainty, obtain labels, and update without absorbing poisoned data.

Variant Detection Under Polymorphism, Metamorphism, and Obfuscation

Polymorphic and metamorphic malware can preserve malicious behavior while modifying code structure. Behavior-based features address this mismatch by mapping API traces into higher-level actions, and experiments with decision trees, random forests, and SVMs reported effective detection of variants [10]. Dynamic analysis is therefore well matched to transformations that alter static syntax

but leave runtime actions partly intact. Its protection weakens when malware changes execution paths, injects irrelevant calls, or suppresses behavior in the sandbox. Semantics-based sequence characterization was proposed specifically to reduce sensitivity to system-call injection [9].

Static learning is particularly exposed to packing. BenchMFC treats packed families as a distinct drift category and reports that packers preserve machine-learning signals whose robustness remains uncertain [8]. ERMDS addresses evaluation by combining binary, source-code, and packing obfuscations into diverse compositions; its ERMDS-X instance reduced average accuracy by 20% for evaluated learning-based detectors and 32% for evaluated commercial antivirus products [48]. These reductions show that robustness cannot be inferred from testing only the obfuscation methods seen during training. They also support reporting performance across transformations rather than presenting one aggregate score.

Adversarial modification adds a functional constraint: changes must preserve malicious capability. Studies of binary-encoded malware developed functionally preserved adversarial examples and incorporated them into saddle-point training [49]. A multi-attack adversarial-training method reduced average evasion rates to 12% on VirusShare and 18% on VXHeaven, compared with higher rates for selected single-attack training conditions [50]. These values are tied to particular datasets and attacks, but they demonstrate that robustness is threat-model dependent. A model hardened against one perturbation may remain vulnerable to another, so variant evaluation should include transformations that were not used for training.

Static and Dynamic Learning for Cryptomining and Mobile Malware

Specialized threats show how feature choice follows behavior. Cryptomining malware seeks sustained use of CPU or GPU resources, and a deep-learning study evaluated both static PE opcodes and dynamically captured system-call events. Static analysis reached 95% accuracy, while dynamic analysis reached 99% in that corpus [11]. The higher dynamic result is consistent with the visibility of resource-related runtime behavior, but it does not establish that every cryptominer exhibits the same trace or that dynamic analysis is cost-effective for pre-execution screening. Specialized evaluations should therefore report execution conditions and compare false positives from legitimate high-resource applications.

Mobile malware introduces platform-specific features and adaptive attacks. Static Android analysis on the Drebin dataset found that supervised classifiers could achieve high accuracy using a small feature subset [39]. Network-traffic features produced a 99.97% true-positive rate for Bayes network and random forest on MalGenome, while the same study reported an 84.57% true-positive rate for k-nearest neighbors on newer Android malware [51]. The difference across datasets and classifier settings illustrates why platform, feature source, and sample period must remain attached to every performance claim.

Mobile deployment also exposes poisoning and backdoor risks. Carefully crafted training-data injections reduced the accuracy of mobile malware classifiers, while KuafuDet's self-adaptive camouflage detection improved reported accuracy by at least 15% on more than 250,000 applications [52]. A separate Android backdoor study achieved up to 99% evasion across 750 malware samples using a trigger involving four features and a poisoning rate of 0.3% [53]. These findings mean that specialized detectors require more than high in-distribution accuracy. Their evaluation must combine platform-aware static and dynamic evidence with chronological drift, benign-resource confounders, obfuscation, and attacks on both inference and retraining.[54]

Discussion: Security, Robustness, and Deployment Challenges in Learning-Based Detectors
Evasion, Adversarial Examples, Backdoors, and Training-Data Poisoning

Learning-based detectors operate in an adversarial setting in which malware authors can modify observable features while preserving malicious functionality. Binary-domain experiments generated functionally preserved adversarial examples and used saddle-point training to improve resistance [49]. Robustness remains conditional on the attack model: multi-attack training reduced average evasion rates to 12% on VirusShare and 18% on VXHeaven, yet detectors trained against individual attacks performed less consistently [50]. Ensemble defenses also remain vulnerable when attackers combine manipulation methods, including against Android detectors and VirusTotal-like services [55]. Obfuscation therefore requires evaluation with transformations withheld from training, since

ISSN:2584-2706 DOI: https://doi.org/10.5281/zenodo.22025661

ERMDS-X reduced learning-based detector accuracy by an average of 20% [48]. Training integrity presents a separate threat. In Android experiments, a trigger involving four features and a poisoning rate of 0.3% produced up to 99% evasion across 750 malware samples [53]. Mobile poisoning studies likewise show that crafted training samples can reduce classifier accuracy, although self-adaptive camouflage detection improved reported accuracy by at least 15% [52]. Robust deployment thus requires provenance controls, attack-aware validation, and monitoring of both inference inputs and retraining data.

Concept Drift, Malware Evolution, and Continual or Federated Adaptation

Malware evolution changes the relationship between observed features and labels, so a detector trained on historical samples cannot be treated as stationary. BenchMFC formalizes drift through unseen families, packed families, and evolved families across 223,000 samples from 526 families with timestamps and packer labels [8]. Dynamic incremental learning offers one response: AIBL-MVD detects drift in sandbox API traces and introduces new samples in adaptive batches while retaining older data to limit catastrophic forgetting [27]. Its reported average detection accuracy for new and variant malware was 99.41%, with updates occurring 1.35 times per month, although this result remains tied to its corpus and behavioral representation [27]. Federated adaptation adds privacy and heterogeneity constraints because clients observe different, asynchronous drift patterns. FL-MalDrift combines local drift detection with selective participation and reports 92.4% accuracy on a chronological AndroZoo split, illustrating that temporal evaluation can produce lower performance than conventional benchmark splits [56]. FALCON further combines active learning, federated aggregation, uncertainty, diversity, and data pruning, reporting improvements in F1-AUT and reductions in cumulative undetected malware on two longitudinal datasets [28]. These approaches shift the research question from static accuracy toward update frequency, annotation demand, storage growth, privacy, and recovery after drift.

Benchmarking Trustworthiness Under Distribution Shift and Non-IID Data

Trustworthy evaluation must distinguish random holdout performance from resistance to temporal, family, packing, and platform shifts. Chronological splits expose whether a detector recognizes future samples rather than memorizing collection-specific artifacts, as shown by studies designed to remove dataset bias through disjoint timescales [25]. BenchMFC addresses this problem by attaching family, packer, and timestamp information to samples, enabling separate drift tests and comparisons of retraining strategies [8]. Federated settings require additional reporting because client distributions are non-IID and local updates may represent different stages of malware evolution [56]. Accuracy alone cannot establish operational trustworthiness when false negatives, calibration, latency, and analysis cost are omitted. Dynamic analysis may improve behavioral coverage, but sandbox execution remains imperfect and no single tool captures all malware behavior [1]. Benchmarks should therefore report sample provenance, duplicate handling, temporal separation, packer overlap, benign software composition, execution environment, and attack access. Such reporting would make robustness claims comparable and connect evaluation directly to deployment conditions.

Research Gaps: Toward Trustworthy, Cost-Aware, and Reproducible Malware Detection

Reconciling Detection Accuracy With Execution Cost, Interpretability, and Operational Latency

Reported accuracy does not capture the cost of extracting features, executing samples, retraining models, or investigating alerts. Dynamic runtime opcode studies explicitly compare classification accuracy with CPU time and retraining costs as datasets expand, identifying parallelized random forest as a favorable accuracy-cost configuration in that setting [15]. Static models can reduce latency and memory requirements, with low-parameter analysis reporting 93.44% accuracy under extreme memory constraints [6]. Yet static representations are sensitive to packing and may learn packer signals rather than malicious semantics [8]. Hybrid systems offer a route to selective escalation, but their cost must be measured per sample rather than described only as improved accuracy. Interpretability is similarly uneven: MalDAE links correlated static and dynamic API sequences to behavior types and reported detection and classification accuracies of 97.89% and 94.39% [2]. Future

studies should evaluate explanation fidelity, analyst time, false-positive triage, energy use, and end-to-end latency alongside predictive metrics.[57]

Need for Holistic Benchmarks Across Platforms, Families, Time Periods, and Attack Conditions

The field lacks a common benchmark that simultaneously tests platforms, malware families, chronological drift, evasion, and operational resources. EMBER2024 moves toward broader evaluation with more than 3.2 million files across six formats, seven classification tasks, metadata, feature vectors, and a challenge set containing files initially missed by antivirus products [29]. BenchMFC complements this scope with family, packer, and timestamp tags for concept-drift analysis [8]. These resources address different dimensions, but neither alone resolves the need for aligned static, dynamic, and hybrid traces under identical sample partitions. Evaluation should also include benign packed software because packing is used by legitimate applications as well as malware [7]. Adversarial datasets such as ERMDS-X demonstrate the value of unseen combinations of binary, source-code, and packing transformations [48]. Reproducible benchmarks should publish hashes, labels, extraction configurations, temporal protocols, attack generators, resource measurements, and pretrained baselines.

Open Problems in Explainable Hybrid Models and Adaptive Defense Evaluation

Hybrid models still require clearer evidence about when each view contributes reliable information. Integrated static and dynamic classification improved accuracy relative to separate analyses and reduced testing time in one study, while retaining greater robustness across older and newer malware collections [35]. Multiple-kernel learning similarly combines distinct views without assuming that one representation is universally sufficient [3]. These findings support fusion, but they do not establish causal explanations for individual alerts or identify when a view has been compromised. Explainable systems should expose behavior-level evidence, uncertainty, missing observations, and conflicts between static and dynamic signals. Adaptive defenses also need evaluation against repeated attack and update cycles rather than one-time adversarial tests. Existing surveys identify a wider range of attacks than defenses, while practical feasibility remains underexamined [58]. Progress therefore depends on protocols that measure explanation stability, adaptation benefit, forgetting, attack transfer, annotation burden, and degradation over time.@

Conclusion: Integrating Multiview Learning With Adaptive Malware Defense

Synthesis of the Static–Dynamic Trade-Off and the Case for Feature Fusion

Static analysis supports rapid, scalable screening without execution, but packing and obfuscation can conceal or distort its evidence. Dynamic analysis observes runtime behavior and can resist several code-level transformations, although sandbox dependence, incomplete execution, and resource cost constrain deployment. Comparative experiments frequently favor dynamic or integrated approaches, while hybrid representations connect syntax with behavioral semantics and support explanations . Feature fusion should therefore be treated as a conditional design: inexpensive static evidence can filter routine samples, while dynamic analysis can investigate uncertainty or suspicious behavior. The objective is not to maximize one benchmark score, but to combine complementary evidence under explicit cost and latency constraints.

Implications for Robust, Drift-Aware, and Continually Evaluated Detection Systems

A dependable detector must be evaluated as a changing service rather than a fixed classifier. Its lifecycle should include chronological testing, unseen-family assessment, adversarial transformation, poisoning detection, calibrated updating, and measurement of false negatives after deployment. Incremental and federated studies show that adaptation can improve performance, but they also expose annotation, storage, privacy, and non-IID challenges. Reproducible reporting must retain the relationship between data period, platform, feature view, execution environment, and operational cost. Multiview learning offers the technical foundation, while continual evaluation determines whether that foundation remains reliable as malware changes. Future systems should consequently report predictive, security, interpretability, and resource outcomes together, with claims limited to the attack conditions and distributions actually tested.

References

1. [1] O. Or-Meir, N. Nissim, Y. Elovici, and L. Rokach, "Dynamic malware analysis in the modern era—A state of the art survey," *ACM Computing Surveys*, vol. 52, no. 5, pp. 1–48, Sep. 2019, doi: 10.1145/3329786.
2. [10] H. S. Galal, Y. B. Mahdy, and M. A. Atiea, "Behavior-based features model for malware detection," *Journal of Computer Virology and Hacking Techniques*, vol. 12, no. 2, pp. 59–67, Jun. 2015, doi: 10.1007/s11416-015-0244-0.
3. [11] H. Darabian *et al.*, "Detecting cryptomining malware: a deep learning approach for static and dynamic analysis," *Journal of Grid Computing*, vol. 18, no. 2, pp. 293–303, Jan. 2020, doi: 10.1007/s10723-020-09510-6.
4. [12] R. Haider, Md. R. Amin, Md. S. Arafat, I. Hossain, and R. Ahmad, "Smart Automation: Revolutionizing Business Operations, Advertising Costs and Customer Satisfaction Through Technology," *European Economic Letters*, vol. 16, no. 1, p. null, 2026.
5. [13] "Generative AI and quantum-inspired optimization: Redefining portfolio risk management and real-time capital allocation in volatile markets," *Journal of Informatics Education and Research*, Mar. 2026, doi: 10.52783/jier.v6i1.4540.
6. [14] Z. Zhang, P. Qi, and W. Wang, "Dynamic malware analysis with feature engineering and feature learning," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 01, pp. 1210–1217, Apr. 2020, doi: 10.1609/aaai.v34i01.5474.
7. [15] D. Carlin, P. O'Kane, and S. Sezer, "A cost analysis of machine learning using dynamic runtime opcodes for malware detection," *Computers & Security*, vol. 85, pp. 138–155, Aug. 2019, doi: 10.1016/j.cose.2019.04.018.
8. [16] H. Raiyan, D. Tosendra, G. A. Gauri, V. Nidhi, K. Babhuti, and D. V. Shankar, "Neuromarketing Approaches to Shaping Healthy Consumer Choices: An Integrative Analysis of Methods, Efficacy, and Ethical Considerations," 2026.
9. [17] H. Raiyan, Md. F. I. Shaif, R. Ahmed, N. H. Nafi, M. R. Sumon, and M. Rahman, "Assessing the impact of influencer marketing on brand value and business revenue: An empirical and thematic analysis," *International Journal of Science and Research Archive*, vol. 16, no. 02, pp. 471–482, 2025, doi: 10.30574/ijsra.2025.16.2.2355.
10. [18] D. Gibert, C. Mateu, and J. Planes, "The rise of machine learning for detection and classification of malware: Research developments, trends and challenges," *Journal of Network and Computer Applications*, vol. 153, pp. 102526–102526, 2020, doi: 10.1016/j.jnca.2019.102526.
11. [19] D. Ucci, L. Aniello, and R. Baldoni, "Survey of machine learning techniques for malware analysis," *Computers & Security*, vol. 81, pp. 123–147, Mar. 2019, doi: 10.1016/j.cose.2018.11.001.
12. [2] W. Han, J. Xue, Y. Wang, L. Huang, Z. Kong, and L. Mao, "MalDAE: Detecting and explaining malware based on correlation and fusion of static and dynamic characteristics," *Computers & Security*, vol. 83, pp. 208–233, Jun. 2019, doi: 10.1016/j.cose.2019.02.007.
13. [20] R. Hasan *et al.*, "Enhancing malware detection with feature selection and scaling techniques using machine learning models," *Scientific Reports*, vol. 15, no. 1, Mar. 2025, doi: 10.1038/s41598-025-93447-x.
14. [21] G. Kale, G. E. Bostancı, and F. V. Çelebi, "Evolutionary feature selection for machine learning based malware classification," *Engineering Science and Technology, an International Journal*, vol. 56, p. 101762, Aug. 2024, doi: 10.1016/j.jestch.2024.101762.
15. [22] Y. Ye, L. Chen, S. Hou, W. Hardy, and X. Li, "DeepAM: a heterogeneous deep learning framework for intelligent malware detection," *Knowledge and Information Systems*, vol. 54, no. 2, pp. 265–285, May 2017, doi: 10.1007/s10115-017-1058-9.
16. [23] K. Shaukat, S. Luo, and V. Varadharajan, "A novel deep learning-based approach for malware detection," *Engineering Applications of Artificial Intelligence*, vol. 122, p. 106030, Jun. 2023, doi: 10.1016/j.engappai.2023.106030.
17. [24] D. Gibert, J. Planes, C. Mateu, and Q. Le, "Fusing feature engineering and deep learning: A case study for malware classification," *Expert Systems with Applications*, vol. 207, p. 117957, Nov. 2022, doi: 10.1016/j.eswa.2022.117957.
18. [25] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, and S. Venkatraman, "Robust intelligent malware detection using deep learning," *IEEE Access*, vol. 7, pp. 46717–46738, 2019, doi: 10.1109/access.2019.2906934.

19. [26] F. Deldar and M. Abadi, "Deep learning for zero-day malware detection and classification: A survey," *ACM Computing Surveys*, vol. 56, no. 2, pp. 1–37, Sep. 2023, doi: 10.1145/3605775.
20. [27] A. A. Darem, F. A. Ghaleb, A. A. Al-Hashmi, J. H. Abawajy, S. M. Alanazi, and A. Y. Al-Rezami, "An adaptive behavioral-based incremental batch learning malware variants detection model using concept drift detection and sequential deep learning," *IEEE Access*, vol. 9, pp. 97180–97196, 2021, doi: 10.1109/access.2021.3093366.
21. [28] K. Wang *et al.*, "FALCON: Federated active learning-based concept drift adaptation for malware detection," *IEEE Transactions on Mobile Computing*, pp. 1–18, 2026, doi: 10.1109/tmc.2026.3684501.
22. [29] 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*, ACM, Aug. 2025, pp. 5516–5526. doi: 10.1145/3711896.3737431.
23. [3] B. Anderson, C. Storlie, and T. Lane, "Improving malware classification," in *Proceedings of the 5th ACM workshop on Security and artificial intelligence*, ACM, Oct. 2012, pp. 3–14. doi: 10.1145/2381896.2381900.
24. [30] Raiyan Haider, Md. Farhan Ishrak Shaif, Raiyan Ahmed, Nahid Hasan Nafi, Mahmudul Reza Sumon, and Mushfiqur Rahman, "The influence of social media branding on consumer purchase behavior: A comprehensive empirical and thematic analysis," *International Journal of Science and Research Archive*, vol. 16, no. 2, pp. 460–470, Aug. 2025, doi: 10.30574/ijsra.2025.16.2.2354.
25. [31] Raiyan Haider, Jafia Tasnim Juba, and Satu Chowdhury, "Exploring the link between suicidal ideation and digital environments: The hidden impact of marketing content," *International Journal of Science and Research Archive*, vol. 16, no. 2, pp. 607–614, Aug. 2025, doi: 10.30574/ijsra.2025.16.2.2353.
26. [32] K. Shaukat, S. Luo, and V. Varadharajan, "A novel machine learning approach for detecting first-time-appeared malware," *Engineering Applications of Artificial Intelligence*, vol. 131, p. 107801, May 2024, doi: 10.1016/j.engappai.2023.107801.
27. [33] A. Damodaran, F. D. Troia, C. A. Visaggio, T. H. Austin, and M. Stamp, "A comparison of static, dynamic, and hybrid analysis for malware detection," *Journal of Computer Virology and Hacking Techniques*, vol. 13, no. 1, pp. 1–12, Dec. 2015, doi: 10.1007/s11416-015-0261-z.
28. [34] K. Rieck, P. Trinius, C. Willems, and T. Holz, "Automatic analysis of malware behavior using machine learning," *Journal of Computer Security*, vol. 19, no. 4, pp. 639–668, Jun. 2011, doi: 10.3233/jcs-2010-0410.
29. [35] R. Islam, R. Tian, L. M. Batten, and S. Versteeg, "Classification of malware based on integrated static and dynamic features," *Journal of Network and Computer Applications*, vol. 36, no. 2, pp. 646–656, Mar. 2013, doi: 10.1016/j.jnca.2012.10.004.
30. [36] P. V. Shijo and A. Salim, "Integrated static and dynamic analysis for malware detection," *Procedia Computer Science*, vol. 46, pp. 804–811, 2015, doi: 10.1016/j.procs.2015.02.149.
31. [37] Raiyan Haider, Wahida Ahmed Megha, Jafia Tasnim Juba, Aroa Alamgir, and Labib Ahmad, "The conversational revolution in health promotion: Investigating chatbot impact on healthcare marketing, patient engagement, and service reach," *International Journal of Science and Research Archive*, vol. 15, no. 3, pp. 1585–1592, Jun. 2025, doi: 10.30574/ijsra.2025.15.3.1937.
32. [38] Raiyan Haider, Farhan Abrar Ibne Bari, Osru, Nishat Afia, and Mohammad Abiduzzaman khan Mugdho, "Leveraging internet of things data for real-time marketing: Opportunities, challenges, and strategic implications," *International Journal of Science and Research Archive*, vol. 15, no. 3, pp. 1657–1663, Jun. 2025, doi: 10.30574/ijsra.2025.15.3.1936.
33. [39] V. Syrris and D. Geneiatakis, "On machine learning effectiveness for malware detection in android OS using static analysis data," *Journal of Information Security and Applications*, vol. 59, p. 102794, Jun. 2021, doi: 10.1016/j.jisa.2021.102794.
34. [4] J. Singh and J. Singh, "A survey on machine learning-based malware detection in executable files," *Journal of Systems Architecture*, vol. 112, pp. 101861–101861, 2020, doi: 10.1016/j.sysarc.2020.101861.
35. [40] M. S. Akhtar and T. Feng, "Evaluation of machine learning algorithms for malware detection," *Sensors*, vol. 23, no. 2, p. 946, Jan. 2023, doi: 10.3390/s23020946.
36. [41] M. Ahmadi, D. Ulyanov, S. Semenov, M. Trofimov, and G. Giacinto, "Novel feature extraction, selection and fusion for effective malware family classification," in *Proceedings of the Sixth ACM*

Conference on Data and Application Security and Privacy, ACM, Mar. 2016, pp. 183–194. doi: 10.1145/2857705.2857713.

37. [42] G. M. and S. C. Sethuraman, "A comprehensive survey on deep learning based malware detection techniques," *Computer Science Review*, vol. 47, p. 100529, Feb. 2023, doi: 10.1016/j.cosrev.2022.100529.
38. [43] J.-Y. Kim, S.-J. Bu, and S.-B. Cho, "Zero-day malware detection using transferred generative adversarial networks based on deep autoencoders," *Information Sciences*, vol. 460–461, pp. 83–102, Sep. 2018, doi: 10.1016/j.ins.2018.04.092.
39. [44] J. Wang, X. Chang, Y. Wang, R. J. Rodríguez, and J. Zhang, "LSGAN-AT: enhancing malware detector robustness against adversarial examples," *Cybersecurity*, vol. 4, no. 1, Dec. 2021, doi: 10.1186/s42400-021-00102-9.
40. [45] A. Brown, M. Gupta, and M. Abdelsalam, "Automated machine learning for deep learning based malware detection," *Computers & Security*, vol. 137, p. 103582, Feb. 2024, doi: 10.1016/j.cose.2023.103582.
41. [46] Raiyan Haider, Md Farhan Abrar Ibne Bari, Md. Farhan Israk Shaif, Mushfiqur Rahman, Md. Nahid Hossain Ohi, and Kazi Md Mashrur Rahman, "Quantifying the impact: Leveraging AI-Powered sentiment analysis for strategic digital marketing and enhanced brand reputation management," *International Journal of Science and Research Archive*, vol. 15, no. 2, pp. 1103–1121, May 2025, doi: 10.30574/ijsra.2025.15.2.1524.
42. [47] Raiyan Haider, Md Farhan Abrar Ibne Bari, Md. Farhan Israk Shaif, and Mushfiqur Rahman, "Engineering hyper-personalization: Software challenges and brand performance in AI-driven digital marketing management: An empirical study," *International Journal of Science and Research Archive*, vol. 15, no. 2, pp. 1122–1141, May 2025, doi: 10.30574/ijsra.2025.15.2.1525.
43. [48] L. Jia, Y. Yang, B. Tang, and Z. Jiang, "ERMDS: A obfuscation dataset for evaluating robustness of learning-based malware detection system," *BenchCouncil Transactions on Benchmarks, Standards and Evaluations*, vol. 3, no. 1, p. 100106, Feb. 2023, doi: 10.1016/j.tbench.2023.100106.
44. [49] A. Al-Dujaili, A. Huang, E. Hemberg, and U.-M. OReilly, "Adversarial deep learning for robust detection of binary encoded malware," in *2018 IEEE Security and Privacy Workshops (SPW)*, IEEE, May 2018, pp. 76–82. doi: 10.1109/spw.2018.00020.
45. [5] L. Liu, B. Wang, B. Yu, and Q. Zhong, "Automatic malware classification and new malware detection using machine learning," *Frontiers of Information Technology & Electronic Engineering*, vol. 18, pp. 1336–1347, 2017, doi: 10.1631/fitee.1601325.
46. [50] K. Shaukat, S. Luo, and V. Varadharajan, "A novel method for improving the robustness of deep learning-based malware detectors against adversarial attacks," *Engineering Applications of Artificial Intelligence*, vol. 116, p. 105461, Nov. 2022, doi: 10.1016/j.engappai.2022.105461.
47. [51] F. A. Narudin, A. Feizollah, N. B. Anuar, and A. Gani, "Evaluation of machine learning classifiers for mobile malware detection," *Soft Computing*, vol. 20, no. 1, pp. 343–357, Nov. 2014, doi: 10.1007/s00500-014-1511-6.
48. [52] S. Chen *et al.*, "Automated poisoning attacks and defenses in malware detection systems: An adversarial machine learning approach," *Computers & Security*, vol. 73, pp. 326–344, Mar. 2018, doi: 10.1016/j.cose.2017.11.007.
49. [53] C. Li *et al.*, "Backdoor attack on machine learning based android malware detectors," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 5, pp. 3357–3370, Sep. 2022, doi: 10.1109/tdsc.2021.3094824.
50. [54] Raiyan Haider, Md Farhan Abrar Ibne Bari, Osru, Nishat Afia, and Tanjim Karim, "Illuminating the black box: Explainable AI for enhanced customer behavior prediction and trust," *International Journal of Science and Research Archive*, vol. 15, no. 3, pp. 247–268, Jun. 2025, doi: 10.30574/ijsra.2025.15.3.1674.
51. [55] D. Li and Q. Li, "Adversarial deep ensemble: Evasion attacks and defenses for malware detection," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3886–3900, 2020, doi: 10.1109/tifs.2020.3003571.
52. [56] A. Patel, D. S. Tomar, R. K. Pateriya, and R. Haripriya, "FL-MalDrift: a federated learning framework for malware detection under local concept drift," *Scientific Reports*, vol. 16, no. 1, Dec. 2025, doi: 10.1038/s41598-025-31592-z.

53. [57] Raiyan Haider and Jasmima Sabatina, "Harnessing the power of micro-influencers: A comprehensive analysis of their effectiveness in promoting climate adaptation solutions," *International Journal of Science and Research Archive*, vol. 15, no. 2, pp. 595–610, May 2025, doi: 10.30574/ijsra.2025.15.2.1448.
54. [58] N. Martins, J. M. Cruz, T. Cruz, and P. Henriques Abreu, "Adversarial machine learning applied to intrusion and malware scenarios: A systematic review," *IEEE Access*, vol. 8, pp. 35403–35419, 2020, doi: 10.1109/access.2020.2974752.
55. [6] R. B. del Aguila, C. D. C. Pérez, A. G. S. Trujillo, J. C. Cuevas-Tello, and J. Nunez-Varela, "Static Malware Analysis Using Low-Parameter Machine Learning Models," *Computers*, vol. 13, pp. 59–59, 2024, doi: 10.3390/computers13030059.
56. [7] H. Aghakhani *et al.*, "When malware is packin' heat; limits of machine learning classifiers based on static analysis features," in *Proceedings 2020 Network and Distributed System Security Symposium*, Internet Society, 2020. doi: 10.14722/ndss.2020.24310.
57. [8] Y. Jiang, G. Li, S. Li, and Y. Guo, "BenchMFC: A benchmark dataset for trustworthy malware family classification under concept drift," *Computers & Security*, vol. 139, p. 103706, Apr. 2024, doi: 10.1016/j.cose.2024.103706.
58. [9] S. Naval, V. Laxmi, M. Rajarajan, M. S. Gaur, and M. Conti, "Employing program semantics for malware detection," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 12, pp. 2591–2604, Dec. 2015, doi: 10.1109/tifs.2015.2469253.