International Journal of Cyber Threat Intelligence and Secure Networking (IJCTISN)
eISSN: 3087-4297
Volume03 Issue07 | Page No: - 06-19

Received: 25 May 2026, Revised: 15 June 2026, Accepted: 26 June 2026, Published: 21 July 2026

DOI: https://doi.org/10.55640/ijctisn-v03i07-02

# A Review of Explainable Machine Learning Methods for Malware Detection and Classification

**Mr. Deepak Mehta** [1]

[1] Assistant Professor, Department of Computer Sciences and Applications, Mandsaur University, Mandsaur, India

**ABSTRACT:** Traditional cybersecurity solutions have been greatly challenged by the fast growth of malware, making accurate and interpretable malware detection crucial. In recent years, deep learning (DL) and machine learning (ML) have gained traction as potent methods for identifying malware, both known and undiscovered, polymorphic, and zero-day. These methods learn intricate patterns from both static and dynamic data analysis. Many ML and DL models, however, are opaque and untrustworthy because to their black-box design, which is particularly problematic for applications that rely on security. Malware detection and categorisation using explainable machine learning approaches is thoroughly reviewed in this study. Starting with a general introduction to malware detection and the most frequent kinds of malware, it moves on to cover the three main classical detection approaches: signature-based, behavioral-based, and heuristic-based. Advanced malware detection approaches based on ML and DL are further examined in the paper, which highlights frequently used algorithms, their working principles, and benefits. Along with that, it delves into XAI approaches like LIME, KernelSHAP, and Shapley values, which are model-agnostic, to enhance the interpretability of malware detection models. These techniques use transparent machine learning models and both global and local explanations. Accumulated Local Effects (ALE), Individual Conditional Expectation (ICE), and Partial Dependence Plot (PDP) are among the visual methods of explanation that are covered. The study concludes with a review of the literature, an analysis of the current state of affairs, and a plan for the future of research into the topic of malware detection systems as it pertains to building confidence among users and facilitating educated cybersecurity decisions.

**KEYWORDS:** Malware Detection, Cybersecurity, Classification, Machine Learning, Deep Learning, Explainable Artificial Intelligence (XAI).

## 1. Introduction

Malicious software has grown in sophistication and reach in recent years, becoming a major cybersecurity risk to desktop computers, smartphones, the Internet of things, and cloud computing platforms alike [1]. Code obfuscation, encryption, polymorphism, and metamorphism are some of the ways that malware is getting more sophisticated and diverse. Everyone from people to governments has taken a financial, operational, and reputational hit due to malware's growing complexity and its ability to evade traditional security measures. Cybercriminals may now hack numerous systems at once due to the increased integration of technology across platforms. As a result, if wants to make sure that cybersecurity is resilient, and need to look at multi-platform malware detection [2].

Malicious software, or malware, is software with the explicit goal of causing harm to computer systems or gaining unauthorised access to them. The different types of malware—ransomware, spyware, adware, rootkits, worms, horses, botnets, trojans, and viruses—are classified according to their purposes and information-sharing systems. Multi-platform malware, on the other hand, can infect and spread across multiple platforms, frequently all at once. Computer methods that enable systems to learn from data and gradually enhance their performance without being explicitly programmed are referred to in this study as ML and DL [3]. Particularly in highly complex and variable situations, ML's application to malware detection has demonstrated great promise for automating threat identification and decreasing detection delay [4][5].

The security of data and the dependability of the system depend on the efficient identification of malware. Data loss and leakage events may be successfully reduced, data information can be kept secret and intact, the system can run normally, and business continuity can

be assured by quickly detecting and isolating harmful software [6]. ML algorithms that are easy to understand and use for malware detection are thoroughly examined in this paper. It covers the standard methods for malware detection, advanced ML and DL approaches, and how explainable AI (XAI) may make models more trustworthy and transparent. For the purpose of creating malware detection systems that are accurate, dependable, and interpretable, the article also summarises current research, lists current obstacles, and emphasises future paths.

This review study primarily aims to summarise the following points:

- Comprehensive review of malware detection approaches, These encompass more conventional approaches, such as strategies based on signatures, behaviour, and heuristics.

- Systematic analysis of machine learning and deep learning techniques applied to the task of malware detection, drawing attention to their underlying concepts, capabilities, and practicality.

- Detailed review of Explainable Artificial Intelligence (XAI) methods, including transparent models, global and local explainability, LIME, KernelSHAP, Shapley values, ICE, PDP, and ALE.

- Comparative analysis of existing research studies, identifying current trends, challenges, and limitations in explainable malware detection and classification.

- Discussion of future research directions ln order to create trustworthy malware detection systems that are accurate, interpretable, and built using explainable AI and advanced learning approaches.

A. Organization of the paper

This paper is structured as follows for the rest of it. Malware detection and the many varieties of malware are covered in Section II. Conventional methods for detecting malware are covered in Section III. Methods for detecting malware using ML and DL are discussed in Section IV. Methods for explainable ML are presented in Section V. Section VI provides a literature assessment of current works, while Section VII provides suggestions for further research.

## 2. Malware Detection and Types: An Overview

Malicious software, commonly spread by user-interfaced links or files, has the ability to corrupt data or processes or get unauthorised access to a network. Malicious software, also known as malware, is created with the intention of compromising systems or gaining unauthorised access. Data theft, file corruption, or the inaccessibility of services due to system interruption is still a major concern, even with cybersecurity developments. There are many different types of malware, each with its own unique set of characteristics and goals [7][8]. Common examples include viruses, worms, trojans, ransomware, spyware, adware, botnets, and rootkits. Viruses, worms, rootkits, and trojans are all examples of malicious software that may alter or destroy data, reproduce itself over networks, or enable remote control. Unwanted advertisements are shown by adware, user actions are tracked by spyware, resources are exploited by botnets, and unauthorised access is gained by backdoors. Malware has many different functional aims, and its danger landscape is always changing, thus this categorisation is important.

### Types of Malware Detection and analysis

Malware categorisation is far from an easy task. Malicious software is easily identifiable as something which permits unauthorised control of a system. There are many different kinds of malware [9][10]. Common ways to categorise them include the harmful program's ability to spread and the specific tasks it performs on compromised machines. The most typical forms of malicious software are detailed here:

#### 1) Virus

A computer virus is a malicious piece of software that may replicate and spread once it is executed on a computer. Viruses often come alongside other programs, like documents, and can infect computers. An example of this would be opening a malicious attachment in an email and inadvertently causing a virus infection on computer. Viruses have the ability to corrupt data, slowly drain system resources, and secretly record keystrokes.

#### 2) Trojan

This malicious software typically infiltrates a user's system through deceptive means, such as free downloads or email attachments [11]. Whether it's stealing sensitive data, tracking users' online activity, or obtaining unauthorised access to company networks, once downloaded, malicious malware carry out its intended goal. When a Trojan is present, it causes the device to perform strangely, such making changes to the computer's settings without warning.

#### 3) Worm

Computer worms are a kind of Trojan horse virus that, once installed, may infect other computers and duplicate themselves automatically. The Internet or local area network (LAN) is a common vector for worms to propagate.

#### 4) Spyware

A common misconception is that spyware is malicious software with the intent to penetrate computer, gather

personal information about, and then unknowingly transmit it to an outside entity. Even legal programs that monitor online activity for advertising or other commercial ends are considered spyware. But the point of making malicious software is to make money off of stolen information.

**5) Ransomware**

Ransomware is a kind of malicious software that can encrypt data and prevent users or organisations from accessing their files. Cybercriminals encrypt these data and then demand payment to unlock them; this forces businesses to choose between paying the ransom and losing access to their files. Data theft is only one of many additional features that some ransomware variants have implemented to make victims pay the requested ransom.

**6) Bot**

A malicious actor can get control of the user's machine or execute a predetermined action invisibly thanks to this virus. It is common practice for large-scale assaults to utilise bot malware in order to exploit the system's processing power.

**7) Crypto-Malware**

A threat actor can perform cryptojacking attacks using this software. While both criminals and legitimate cryptocurrency miners utilise the same basic technique, crypto-malware takes advantage of users' devices and processing power to steal money.

**8) Fileless Malware**

A malicious program that resides in memory. Malware that operates in memory, as the name suggests, does not save its files to the hard drive but rather uses RAM. Since no files exist to check, it is more elusive than traditional malware. Also, as the sufferer dies, the infection goes away.

A. Malware Detection Techiques in Cyberseucity

The ever-changing nature of malware has made malware detection a cornerstone of cybersecurity research. As shown in figure 1, there are two main groups into which the current literature divides malware detection techniques: online based approaches and file classification. The latter group is further split into static, dynamic, and hybrid analysis. With respect to processing overhead, detection accuracy, and resistance to escape strategies used by contemporary malware, each of these methods has its own set of benefits and drawbacks.

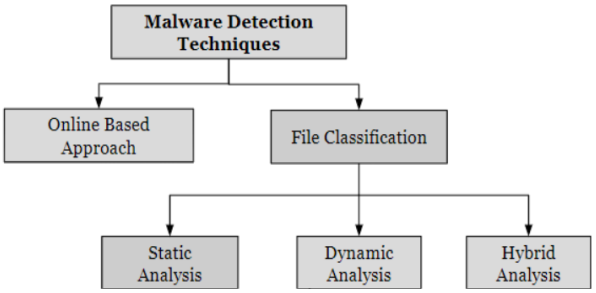


**Fig.1. Malware Detection Techniques**

There are two types of malware detection techniques are:

A. Online Malware Detection

One unique method in the field of cybersecurity is online malware detection. Online detection keeps tabs on the whole system in real-time, allowing the capture of malware at any given instant, irrespective of its activity level, in contrast to static, dynamic, or hybrid approaches that examine individual malware samples. Instead than concentrating on the actions of specific viruses, the method examines the computer as a whole [12]. Rapid identification of previously reported malware samples is achieved through the use of crowd-sourced expertise, which aggregates detection results from numerous antivirus engines and worldwide threat feeds.

B. File Classification Approach

The primary goal of file categorisation is to identify malicious software by analysing its source code. Finding a file that could be malicious is the first step [13]. File categorisation uses a variety of approaches, mostly grouped into three types, to evaluate its nature: Three types of analysis: static, dynamic, and hybrid.

- **Static Analysis**: The process entails inspecting the information, code, and structure of a file without actually running it. It is the first line of defense in file classification.

- **Dynamic Analysis**: This method entails running the file in a sandbox setting and keeping an eye on its actions to see what it does in real time.

- **Hybrid Analysis**: It is the modern gold standard for file classification. It combines the speed and structural insights of static analysis with the behavioral depth of dynamic analysis.

Table 1 below compares and contrasts static and dynamic assessments according to their respective benefits and drawbacks [14].

**Table 1: Comparison of static and dynamic analysis**

| Sr. No. | Static Analysis | Dynamic Analysis |
|---|---|---|

| | | |
|---|---|---|
| **1** | Fast and safe. | Time-consuming and more vulnerable. |
| **2** | Good at analyzing multipath malware (provides a global view). | Difficult to analyze multipath malware. |
| **3** | Cannot analyze obfuscated and polymorphic malware. | Can analyze obfuscated and polymorphic malware. |
| **4** | Cannot detect new or unknown malware. | Can detect both known and unknown malware. |
| **5** | Low level of false positives (high accuracy). | High level of false po |

Previous research has combined data extracted through static and dynamic analysis to mitigate the limitations of both methodologies and increase the detection rate. A variety of tools are utilised to gather both static and dynamic data, including OlleyDbg, IDA pro disassembler, and Cuckoo sandbox. Subsequently, hybrid feature sets are constructed using a variety of data kinds, including text, opcode, API calls, and others [15]. While hybrid analysis does have certain advantages over static and dynamic approaches, it also has some drawbacks.

## 3. Traditional Malware Detection Approaches

A malware detection method is a necessary automated procedure for finding and identifying files that are harmful. Thus, several malware detection methods have evolved over the years, but no one method has shown to be 100% effective against every malware family and in every scenario. Signatures and behaviours are the two primary features of malicious software. Three malware detection approaches—heuristic-based, behavioral-based, and signature-based—have been developed to address this issue. Consequently, the methods for detecting malware are covered in the sections that follow [16].

1) Signature-Based

There have been a number of attempts to enhance malware detection and classification models by comparing the collected file signature with a previously derived unique signature, either statically or dynamically stored. One or more API calls, opcodes or byte-code series, and the entropy amount are some of the signs that can be used. Generated static signatures based on n-grams and binary vectors, as well as static string-based signatures that describe the retrieved strings using frequency vectors, have been used to identify malicious software written in VBasic. Also used opcode statistics to create harmful static signatures. However, behavioural fingerprints have been built using data that is acquired dynamically [17]. The behavioural malicious signatures are built using certain sets of API calls that are known to indicate harmful activity [18]. Classifying unknown signatures that may be associated with new malware or various versions of known malware is a challenge for static and behavioural signature-based malware detection algorithms, leading to low detection rates.

2) Behavioral-Based

The developed model can classify known malicious behaviours and any behaviour that seems similar with respect to false positive behaviours [19][20]. Features extraction techniques were developed to extract the sensitive features after monitoring the executable files in an isolated environment and collecting the exhibited behaviours. This method outperforms the signature-based approach because it can detect new malware behaviours alongside the existing ones by collecting behaviours during runtime. Therefore, to improve malware detection rates, much of the research in the literature review concentrated on behavioral-based techniques that exploited common, sequential, and continuous behaviours.

3) Heuristic-Based

A number of studies have relied on heuristics to bolster their models for identifying malevolent intent by creating general rules to examine the retrieved data provided by either dynamic or static analysis [21]. ML, the YARA tool, and other technologies can automatically construct the rules, or skilled analysts can use their expertise and experience to build the rules manually.

There are three primary methods for detecting malware, and Table 2 contrasts them: heuristic, behavioural, and signature-based. It describes how they detect malware, what characteristics they have, and what limits they have when it comes to recognising current and future threats.

**Table 2: Comparison of Malware Detection Approaches**

| Criteria | Signature-Based | Behavioral-Based | Heuristic-Based |
|---|---|---|---|
| **Detection Principle** | Matches file signatures with a database of known malware signatures. | Monitors and analyzes malware behavior during execution. | Applies predefined or machine-generated rules to identify suspicious activities. |
| **Analysis Type** | Static and/or Dynamic | Dynamic | Static and Dynamic |
| **Detection Basis** | Known signatures (API calls, opcodes, strings, byte-code, entropy). | Runtime behaviors (API calls, system calls, registry, network, CPU, memory). | Heuristic rules, YARA rules, machine learning rules, DNS patterns, JavaScript/HTML elements. |
| **Known Malware Detection** | Excellent | Excellent | Good |
| **Unknown Malware Detection** | Poor | Excellent | Good |
| **Detection of Polymorphic/Obfuscated Malware** | Poor | Excellent | Good |
| **Detection Speed** | Fast | Slow | Moderate |
| **Resource Consumption** | Low | High | Moderate |
| **False Positive Rate** | Low | Moderate to High | Moderate |
| **Computational Overhead** | Low | High | Moderate |
| **Strengths** | High accuracy for known malware, low false positives, efficient scanning. | Detects zero-day, polymorphic, and unknown malware by analyzing runtime behavior. | Can identify previously unseen malware using generalized rules and learning techniques. |
| **Limitations** | Cannot detect new or modified malware variants; requires frequent signature updates. | Time-consuming, resource-intensive, and may produce higher false positives. | Effectiveness depends on rule quality; may generate false positives if rules are too general. |
| **Typical Applications** | Antivirus software, malware scanners. | Sandboxing, endpoint detection and response (EDR), malware analysis systems. | Intrusion detection systems (IDS), YARA-based scanners, machine learning malware detection frameworks. |

## 4. Advanced Malware Detection Techniques Machine and Deep Learning

Modern malware detection methods discover patterns from massive datasets using ML and DL, which can detect both well-known and previously unseen malware. These approaches identify malicious activity by analysing data such as API calls, opcode sequences, network traffic, and system behaviour, as opposed to traditional signature-based methods [22][23]. ML techniques offer precise categorisation at a reduced computational cost, but they need human feature extraction. The detection of complicated, polymorphic, and zero-day malware is greatly improved by DL models, which automatically extract complex characteristics from raw data. With the help of these smart methods, malware detection has become much more accurate, flexible, and resistant to new cyber threats.

### A. Machine Learning-Based Malware Detection

Malware detection using ML is becoming increasingly popular. ML learns patterns from both malicious and benign files that have been analysed before. When building ML models, it is necessary to first extract features from static or dynamic analysis. These

information might include API calls, opcode sequences, permissions, network traffic, and file attributes. DT, RF, SVM, NB, KNN, and LR are algorithms that are commonly used in ML, as shown in table 3. With the use of these algorithms, the requirement for manually developed signatures may be significantly reduced while efficiently classifying both known and novel malware. Their efficacy, however, is conditional on the characteristics that were retrieved and the data that was labelled for training [24].

**Table 3: Machine Learning Techniques for Malware Detection**

| Technique | Description | Advantages |
|---|---|---|
| **Decision Tree (DT)** | Classifies malware using a tree-like structure based on feature values. | Easy to understand, fast training, and interpretable decision-making. |
| **Random Forest (RF)** | Combines multiple decision trees to improve classification accuracy. | High accuracy, robust to overfitting, and handles large datasets effectively. |
| **Support Vector Machine (SVM)** | Separates malware and benign samples using an optimal hyperplane. | Effective for high-dimensional data and provides accurate classification. |
| **Naïve Bayes (NB)** | Uses Bayes' theorem to classify samples based on feature probabilities. | Simple, fast, and performs well with limited training data. |
| **K-Nearest Neighbors (KNN)** | Classifies malware based on the labels of the nearest neighboring samples. | Easy to implement and effective for pattern recognition. |
| **Logistic Regression (LR)** | Predicts the probability of a file being malicious using a logistic function. | Fast, efficient, and suitable for binary malware classification. |
| **Artificial Neural Network (ANN)** | Learns malware patterns through interconnected artificial neurons. | Capable of modeling complex relationships and imp |

B. Deep Learning-Based Malware Detection

Automated feature learning from raw malware data is possible using DL, a cutting-edge subfield of ML, thanks to the elimination of the need for human feature engineers [25]. The ability of DL models to analyse massive datasets and uncover malware's hidden patterns of behaviour makes them excellent at identifying zero-day, polymorphic, and obfuscated threats [26]. NNs, AEs, CNNs, and DNNs are well-known DL models utilised for malware detection, as shown in table 4. Although DL provides higher detection accuracy and better generalization than traditional ML methods, it requires large datasets, high computational resources, and longer training times.

**Table 4. Deep Learning Techniques for Malware Detection**

| Technique | Description | Advantages |
|---|---|---|
| **Deep Neural Network (DNN)** | Uses multiple hidden layers to learn complex patterns from malware data. | Learns complex feature representations and provides high |

| | | |
|---|---|---|
| | | detection accuracy. |
| **Convolutional Neural Network (CNN)** | Extracts spatial features automatically from malware images or binary files. | Automatic feature extraction with excellent performance on malware image classification. |
| **Recurrent Neural Network (RNN)** | Processes sequential data by retaining information from previous inputs. | Effectively captures sequential malware behaviors such as API and system call sequences. |
| **Long Short-Term Memory (LSTM)** | A specialized RNN that captures long-term dependencies in sequential data. | Accurately models long-term behavioral patterns and detects advanced malware. |
| **Gated Recurrent Unit (GRU)** | Simplified version of LSTM that efficiently models sequential dependencies. | Faster training with performance comparable to LSTM while using fewer parameters. |
| **Autoencoder (AE)** | Learns compact feature representations by encoding and reconstructing input data. | Excellent for unsupervised feature learning and anomaly detection. |
| **Generative Adversarial Network (GAN)** | Consists of a generator and discriminator to learn malware data distributions. | Enhances model robustness by generating realistic malware samples for training. |

## 5. Explainability In Machine Learning

Models based on ML allow computers to build intricate hierarchical data structures, which are crucial for jobs like detection and categorisation. These opaque models can improve systems' prediction abilities by combining and stacking several layers of data representation. Their decision logic may be called into doubt, nevertheless, because this heightened complexity frequently masks their internal decision-making process [27][28][29]. A more open and honest approach is provided by white-box models. They are made to be simply understood by anyone, so people can see how their data is turned into forecasts or judgements. This openness is especially helpful in domains where comprehending the logic underlying a choice is just as crucial as comprehending the decision itself [30]. Blanco-Justicia and Domingo-Ferrer were the authors of the in-depth research. In order to improve the openness and efficiency of AI systems, outlined the seven features that constitute XAI.

- **Accuracy**: This metric measures the accuracy with which a XAI model can foretell the results of using previously unknown data. These models' predictions need to be quite precise.

- **Consistency**: This property defines the fairness of applying explanations to two models trained on the same dataset. Continued reliability. Specifically, it checks if the stability is reflected in the explanation model, which would imply that comparable cases should provide comparable explanations.

- **Degree of Importance**: Understanding the relative importance of the model's characteristics in making decisions relies on this property, which shows how accurately the explanation represents those features.

- **Representativeness**: This aspect greatly impacts explainability, highlighting the need of explanations being relevant and applicable in many decision-making

scenarios to guarantee their value in different contexts.

A. Transparent Machine-Learning Models

The capacity to explain themselves is a defining feature of transparent models. Because of their straightforward interpretability, users are able to understand how they formulate decisions. The next sections go into further into about various approaches, describing them and pointing out their advantages, disadvantages, and real-world uses.

1) Global Explanation

A model's decision-making process can be better understood by looking at its global explainability rather than just at its predictions. Knowing which qualities are systemically important and how they interact requires this information. A common method for attaining global explainability is to employ surrogate models. These models, which are more interpretable (e.g., decision trees or linear models) and trained on the same dataset as the sophisticated black-box model, mimic the decision-making process of the latter [31]. Transparent analysis of feature contributions is made possible by these models, which aids in the understanding of the models.

2) Local explainability

The goal of local explainability is to deduce the reasoning behind a model's predictions for individual instances, as opposed to providing an explanation for the model's overall behaviour. Questions like "What characteristics led to this anomaly?" or "Why was this specific file labelled as malware?" may be addressed with the use of these techniques. Local explanations shed light on decision-making at the individual level, in contrast to global explanations that give a picture of feature value over a whole dataset.

B. Model-Agnostic Techniques

There are a number of popular model-agnostic methods for local explainability, such as LIME, KernelSHAP, and Shapley's values. What follows is a more in-depth discussion of these strategies, with an emphasis on their merits, shortcomings, and real-world uses.

**1) LIME (Local Interpretable Model-Agnostic Explanations)**

A new local explainability method called LIME is initially introduced by Ribeiro et al. [32]. This approach generates interpretable predictions by approximating the model's behaviour around a specified prediction, acting as a local surrogate model. Model-agnostic means that it can be used with a variety of ML models. Using a local fidelity metric, LIME determines how well the surrogate model performed. This measure determines how well LIME's approximations capture the essence and precision of the black-box model. Keep in mind that LIME can't helps, understand how a model works on a global scale. Another issue that might undermine LIME's interpretability dependability is if the local fidelity metric shows low accuracy.

**2) KernelSHAP (Shapley Additive Explanations)**

A major obstacle among the several local interpretability strategies is picking the right one for each given situation. A concept inspired from game theory, Shapley Additive explanations (SHAP) is proposed by Lundberg and Lee [33] to solve this. It examines the relevance of each characteristic in contributing to a given prediction. By defining a novel solution with desired qualities, the SHAP framework creates a new class of additive feature significance measures. The fact that KernelSHAP is modelagnostic means that it may be used with a wide variety of ML models. Exact SHAP values computed using KernelSHAP can take an exponential amount of time, demonstrating the computing needs of the method.

**3) Shapley Values**

The idea behind them is based on coalitional game theory, which positions each instance's feature value as a "player" and the forecast outcome as the "payout." This method may show how each information affects the final forecast by giving it a measurable contribution [34]. Standards like local precision and consistency are what set Shapley values apart. Following these guidelines guarantee that features are fairly and understandably prioritised, with each feature's impact on the final result being fairly reflected.

C. Visual Explanation Techniques

This method includes ways to make models more accessible and easier to understand by turning them into images. Accumulated Local Effects (ALE), Individual Conditional Expectation (ICE), and Partial Dependence Plot (PDP) are crucial resources for this visualisation. Methods like this make it easier to comprehend how models work by providing visual representations of the connection between features and the predictability of the model [35].

**1) Partial Dependence Plot (PDP)**

According to Molnar [58], it provides information on how a small number of characteristics might affect the anticipated result of an ML model. A linear or complicated relationship between the target and characteristics can be better understood using this tool. A linear regression model is one setting where PDP may show a linear relationship and the correlation between feature alterations and forecast changes. While other approaches look at how characteristics affect specific

predictions, PDP takes a broader view, focusing on how features affect the model as a whole. But, in most cases, and only use it to examine two characteristics at once, and that's on the premise that the ones choose are completely separate from the others that aren't shown in the graph.

**2) Individual Conditional Expectation (ICE)**

Visual explanations, especially those that work with model-agnostic methodologies, are quite uncommon in the post-hoc explainability. The ICE plot, first shown by Goldstein et al. [36], is a method for visualising the results that models controlled by supervised learning algorithms are expected to produce. The ICE plot deviates from the PDP by showing, on separate lines for each instance, how predictions are dependent on a certain attribute. Users may learn the case-by-case effect of feature modifications on prediction using this method.

**3) Accumulated Local Effects (ALE)**

The ALE plot is a graphical representation of the impact that characteristics have on ML model predictions. It provides a clear picture of the interplay between the dependent variable (the dependent variable itself) and the regressors (the independent variables). The effectiveness of ALE plots is well-known, and they are quicker to produce than PDP plots.

## 6. Literature Review

AI based on intelligent ML, DL, and XAI has recently replaced traditional signature-based methods in malware detection research. These techniques not only improve the detection of sophisticated malware but also enhance the transparency of prediction models.

Firstly, P. Sen et al. (2026), paper established a unified and explainable phishing-attachment detection framework using the CIC-Trap4Phish dataset. This work construct data analysis on 79,293 samples and consolidate four structures into a single aligned feature space for an 8-class setting. Seven supervised models are empirically compared under identical conditions; Random Forest performs best, achieving 98.45 % test accuracy with approximately 1.5% generalization gap, supported by cross-validation, confusion matrix analysis, learning curves, ROC evaluation and OOB estimation. By demonstrating that security-relevant cues (e.g., file size, entropy, macro/script signals, and obfuscation patterns) determine predictions, LIME as well as SHAP are used to increase transparency[37].

Similarly, M. Mia et al. (2026), utilized these to train several DL and ML frameworks. MalXAI amalgamates traditional models including LR, RF, LightGBM, and XGBoost with sequential DL framework including BiLSTM, LSTM, BiGRU, and GRU, facilitating both superior predictive accuracy and interpretability. Experimental analyses of CIC-dgg-2025 and DikeDataset benchmarks showed that RF provided the best overall performance of 97.47 % and AUC of 99.25, whereas BiGRU demonstrated the best performance among the DL models with an accuracy of 91.70. The SHAP explainability method was employed to ensure transparency, and evidence supports the fact that file size and graph connection are the most important attributes used in the classification of malware[38].

Furthermore, A. Alomar et al. (2025), used ML to create a model for detecting malware on Android that is permissions-based. The main objective is to find the permissions list that applications with malicious intent have so that can tell them apart. In order to find the most effective prediction model, gathered a dataset that included permissions for both kinds of apps and compared feature selection strategies (IG, RFE, CHI, GA) with ML algorithms (SVM, DS, RF, GB, XG). Among the models tested, the SVM model with 13 permissions and RFE had the best accuracy, at 98.88%. In twelve milliseconds, the model finished the detecting procedure [39].

In another study, A. Mahato et al. (2025), recommend a classifier called MDCML that can accurately identify suspicious behaviour in Portable Executable (PE) files and group them into different types of malware. Big-2015 and Mal-API-2019, two publicly available datasets that include a wide variety of malware families, are used to assess it. With an accuracy of 98.97% and 95.42% on the relevant datasets, the suggested framework clearly outperforms more conventional ML models, demonstrating notable progress [40].

Likewise, S. Poornima et al. (2024), This study presents a new method for detecting malware assaults on Android devices using a deep belief network, called MAD-NET. This method improves device security by properly detecting and mitigating malware attacks. The feature extraction technique is initially fed the CICAndMal2017 datasets' benign and malicious data. Using the CICAndMal2017 datasets for evaluation and Network Simulator for simulation, and provide the MAD-NET technique. To identify and mitigate malware attacks on Android devices, the MAD-NET technique achieves an overall accuracy of 99.83% for Deep Belief Networks (DBN), which is higher than the 93.11%, 96.75%, and 94.42% achieved by ANN, GAN, and LSTM techniques, respectively [41].

Moreover,R. Hilabi et al. (2024), The objective of this study is to improve an existing model by including several ML methods, specifically random forest (RF) and extreme gradient boosting (XGB), such that it can identify Windows operating system (OS) infection. Along with that, these learning algorithms are trained and tested on

the SOMLAP dataset, which is part of the PE. The results obtained with the combination of XGB and RF were 0.966 for accuracy, 0.990 for precision, and 0.918 for recall [42].

Therefore A. Patel et al. (2024), An Android malware detection (AMD) framework based on ML and an XAI system for smart computing environments are presented in this research. The CICAndMal2019 android malware dataset is used to assess the suggested model with nine distinct ML models. The suggested AMD-XAI-ML model, which includes the XGB, Extra Tree, and RF classifiers, achieved the greatest classification accuracy of 98.54%, 98.52%, and 98.42%, respectively [43].

However, S.Smmarwar et al. (2023), There are a lot of DL models that are used for large-scale android malware detection in existing approaches, however these models don't truly explain how each feature contributes to the detection system, which is quite problematic. Hence, XAI-AMD-DL, a hybrid Android malware detection system employing DL models, is proposed in this study. It combines XAI with CNNs and Bi-Gated Recurrent Units (4RUs). Using the CICAndMal2019 android malware dataset, the suggested model is assessed. The proposed XAI-AMD-DL model beats the current DL models with various metrics such as accuracy (97.98%), precision (97.75%), recall (97.76%), and f1score (97.75%) [44].

Table 5 summarizes recent studies, highlighting the datasets, detection models, explainability techniques, and key performance outcomes.

**Table 5. Summary of Related Work on Explainable Malware Detection and Classification**

| Ref. | Authors (Year) | Dataset | Method / Model | Explainability | Key Results |
|---|---|---|---|---|---|
| **[10]** | P. Sen et al. (2026) | CIC-Trap4Phish | Random Forest, SVM, XGBoost, and other supervised ML models | LIME, SHAP | Random Forest achieved **98.45%** accuracy with transparent feature explanations. |
| **[11]** | M. Mia et al. (2026) | CIC-dgg-2025, DikeDataset | LR, RF, LightGBM, XGBoost, LSTM, BiLSTM, GRU, BiGRU | SHAP | RF achieved **97.47%** accuracy (AUC **99.25%**); BiGRU achieved **91.70%** accuracy. |
| **[12]** | A. Alomar et al. (2025) | Android permission dataset | SVM, RF, GB, XG with RFE, IG, CHI, and GA feature selection | No | SVM with RFE achieved **98.88%** accuracy using only **13 permissions**. |
| **[13]** | A. Mahato et al. (2025) | Big-2015, Mal-API-2019 | Cascade Machine Learning (MDCML) | No | Achieved **98.97%** accuracy on Big-2015 and **95.42%** on Mal-API-2019. |
| **[14]** | S. Poornima et al. (2024) | CICAndMal2017 | Deep Belief Network (MAD-NET), ANN, GAN, LSTM | No | MAD-NET achieved **99.83%** accuracy, outperforming ANN, GAN, and LSTM. |

| | | | | | |
|---|---|---|---|---|---|
| **[15]** | R. Hilabi et al. (2024) | SOMLAP (PE dataset) | Hybrid XGBoost + Random Forest | No | Achieved **96.6%** accuracy, **99.0%** precision, and **91.8%** recall. |
| **[16]** | A. Patel et al. (2024) | CICAndMal2019 | XGB, Extra Trees, Random Forest, and other ML models | XAI | XGB achieved **98.54%** accuracy, followed by Extra Trees (**98.52%**) and RF (**98.42%**). |
| **[17]** | S. Smmarwar et al. (2023) | CICAndMal2019 | Hybrid CNN–BiGRU (XAI-AMD-DL) | XAI | Achieved **97.98%** accuracy with **97.75%** precision, recall, and F1-score. |

**Research Gaps/Challenges**: ML and DL models, including XGBoost, CNN, Deep Belief Networks, BiGRU, and RF, typically exhibit excellent accuracy in malware detection, frequently over 97%, according to the evaluated research. More recently, studies have started to use Explainable AI methods like SHAP and LIME to make models more transparent and to aid analysts in making decisions. Several research gaps persist, even with these advancements. Instead of striking a balance between accuracy, interpretability, computational efficiency, and real-time performance, the majority of present research concentrate on improving detection accuracy. In addition, models that are intrinsically interpretable have not been well investigated, and explainability is frequently restricted to post-hoc procedures. The generalisability of many techniques across varied malware families is questionable because they are examined on restricted datasets. Furthermore, additional research is needed to address issues with adversarial robustness, resource-constrained situations (such as the IoT and edge devices), standardised assessment criteria, and the detection of explainable malware in real-time. These deficiencies call attention to the fact that future cybersecurity applications require malware detection frameworks that are strong, scalable, and intrinsically explicable.

## 7. Conclusion and Future Work

Malware detection has progressed from methods based on classical signatures to more sophisticated methods utilising ML and DL, as discussed in this study. Conventional approaches work well for identifying previously discovered malware, but they fail miserably when faced with complex dangers like zero-day, polymorphic, and metamorphic threats. The detection skills have been greatly improved by ML and DL, which learn complicated patterns from both static and dynamic information. However, there are worries about interpretability and trust due to their black-box nature. In response to these difficulties, XAI methods like as LIME, KernelSHAP, ICE, PDP, and ALE offer explanations for model predictions that are both clear and interpretable. The integration of XAI with intelligent malware detection systems improves decision-making, increases analyst confidence, and enhances system reliability. Future research should focus on creating small, real-time models for detecting explainable malware in IoT, edge, and mobile environments; creating hybrid frameworks that combine static and dynamic analysis with ML, DL, and XAI; making these models more resistant to adversarial, polymorphic, metamorphic, and zero-day malware while lowering the number of FP; and creating standard datasets and evaluation metrics to allow fair comparison, reproducibility, and the real-world use of explainable malware detection solutions.

## REFERENCES


1. S. Chatterjee, "Advanced Malware Detection in Operational Technology: Signature-Based Vs. Behaviour-Based Approaches," ESP J. Eng. Technol. Adv., vol. 1, no. 2, pp. 272–279, 2021, doi: 10.56472/25832646/JETA-V1I2P128.
2. J. Ferdous, R. Islam, A. Mahboubi, and M. Z. Islam, "A Survey on ML Techniques for Multi-Platform Malware Detection: Securing PC, Mobile Devices, IoT, and Cloud Environments," 2025. doi: 10.3390/s25041153.

3. P. H. Desai, P. Mahalle, and P. Chandre, "Intelligent Access Control Schemes for the Internet of Everything: A Survey of Techniques, Challenges, and Future Directions," 2026, pp. 95–105. doi: 10.1007/978-3-032-13196-6_9.

4. N. Adik, "The Rise of Open and Free Networks: A Community-Driven Paradigm for Decentralized Connectivity," in 2025 IEEE First International Conference on Innovations in Engineering and Next-Generation Technologies for Sustainability (ICINVENTS), IEEE, Nov. 2025, pp. 1–9. doi: 10.1109/ICINVENTS64613.2025.11402224.

5. S. Kakkar, S. Janarthanan, B. Makkena, and A. Kakkar, "Deep Learning Approaches for Predicting Attack Vulnerabilities in Quantum-Secure Financial Networks," in 2026 International Conference on Emerging Systems and Intelligent Computing (ESIC), IEEE, Feb. 2026, pp. 780–785. doi: 10.1109/ESIC68176.2026.11496277.

6. Y. Wu, H. Zhuang, Y. Jia, and Y. Zhang, "A Survey of Machine Learning Approaches for Malware Detec-tion," in Proceedings of 2025 5th International Conference on Computer Network Security and Software Engineering, CNSSE 2025, 2025. doi: 10.1145/3732365.3732410.

7. A. Joon, B. K. R. Janumpally, A. Gogineni, and P. Chatterjee, "Efficient Large-Scale Intrusion Identification and Prevention in Distributed Cloud Networks Using Artificial Intelligence," in 2025 5th International Conference on Intelligent Technologies (CONIT), IEEE, Jun. 2025, pp. 1–8. doi: 10.1109/CONIT65521.2025.11167760.

8. M. Mittal, "The Emergence of Blockchain: Security and Scalability Challenges in Decentralized Ledgers," Int. J. Multidiscip. Sci. Emerg. Res., vol. 04, no. 01, Jan. 2016, doi: 10.15662/IJMSERH.2016.0401002.

9. L. A. Yeruva, D. Singh, S. Suddala, N. Bhatt, and R. Uddin, "Augmented Data Management for Cache Performance, Cybersecurity, and Mobile Integration," J. Comput. Mech. Manag., vol. 5, no. 3, Jun. 2026, doi: 10.57159/jcmm.5.3.26691.

10. R. N. Rajendran, D. K. Rai, S. K. Anumula, and S. Agrawal, "Zero Trust Security Model Implementation in Microservices Architectures Using Identity Federation," in 2025 2nd International Conference on Recent Trends in Electrical, Electronics and Computing Technologies (ICRTEECT), IEEE, Oct. 2025, pp. 1–6. doi: 10.1109/ICRTEECT67512.2025.11448625.

11. S. Jain and D. Jain, "Artifact Comparison Analyzer: Evaluating Microservice Build Metrics for Performance and Efficiency Improvements," in 2026 IEEE International Conference on AI Engineering and Innovations (AIEI), IEEE, Mar. 2026, pp. 1–6. doi: 10.1109/AIEI69164.2026.11497468.

12. Y. Muni, "Understanding the Evolution of Internet Protocols: An In-Depth Review of IPV4 and IPV6: A Comparative Review of Transition Challenges and Solutions," Int. J. Adv. Res. Comput. Sci., vol. 16, no. 4, pp. 109–117, Aug. 2025, doi: 10.26483/ijarcs.v16i4.7307.

13. S. Pawar, G. Patil, K. Patel, P. Pawar, S. Khedkar, and B. More, "Falsified News Detection Using Deep Learning Approach," 2021 Asian Conf. Innov. Technol., pp. 1–5, 2021.

14. J. Landage, "Malware and Malware Detection Techniques: A Survey," Int. J. Eng. Res. Technol., vol. 2, no. 12, pp. 1–8, 2013, [Online]. Available: https://www.ijert.org/research/malware-and-malware-detection-techniques-a-survey-IJERTV2IS120163.pdf

15. A. Warrier, "Securing and Scaling API Gateways in Hybrid Environments," J. Artif. Intell. Mach. Learn. Data Sci., vol. 3, no. 3, pp. 2914–2920, Sep. 2025, doi: 10.51219/JAIMLD/Arjun-warrier/607.

16. F. A. Aboaoja, A. Zainal, F. A. Ghaleb, B. A. S. Al-rimy, T. A. E. Eisa, and A. A. H. Elnour, "Malware Detection Issues, Challenges, and Future Directions: A Survey," 2022. doi: 10.3390/app12178482.

17. Vandana Chaturvedi, "A Review on AI-Driven Project Management for Transforming Software Engineering Perspectives," Int. J. Adv. Res. Sci. Commun. Technol., vol. 4, no. 2, p. 895, Dec. 2024, doi: 10.48175/IJARSCT-22800E.

18. K. R. Kiran, "API Integration Barriers in Case-Based Process Management: A Review of Interoperability and Real-Time Response Constraints Outline," Qual. Res., vol. 25, no. 3, pp. 21–48, 2025, doi: https://doi.org/10.5281/zenodo.16782616.

19. S. K. Malaraju and S. K. Madishetty, "AI-Augmented Compiler Optimization for Energyefficient Software Execution on Embedded Systems," in 2025 14th International Conference on System Modeling & Advancement in Research Trends (SMART), IEEE, Nov. 2025, pp. 1–6. doi: 10.1109/SMART66937.2025.11389313.

20. M. R. C. Mukkolakkal, "Deploy, Calibrate, Monitor, Heal -- No Human Required: An Autonomous AI SRE Agent for Elasticsearch," arvix.org, Apr. 2026, [Online]. Available: http://arxiv.org/abs/2604.03933

21. R. Palwe, "SAFIRE: Secure AI Framework for Institutional Readiness & Enablement," Int. J.

Comput. Artif. Intell., vol. 7, no. 4, pp. 136–139, 2026.

22. S. P.Sivashanmugam, S. Kumara, A. Mohile, and D. R. Suram, "Improving Detection Accuracy of Phishing Attacks with Feature Selection and Machine learning Techniques in Cybersecurity," in 2026 1st International Conference on Emerging Trends in Advancements and Applications of Computational Intelligence Techniques (ETAACT), Bhubaneswar, India: IEEE, Apr. 2026, pp. 1–6. doi: 10.1109/ETAACT69135.2026.11542138.

23. R. Dandigam and C. Agrawal, "Comparative Study of Machine Learning Algorithms for Predictive Analytics in Web Applications," in 2026 International Conference on Artificial Intelligence, Systems, and Emerging Technologies (ICAISET), IEEE, Apr. 2026, pp. 1–6. doi: 10.1109/ICAISET66439.2026.11542167.

24. M. H. Hamza Afzal, "Securing AI Systems Against Adversarial Attacks: A Framework for Building Robust and Trustworthy Machine Learning Models," Comput. Fraud Secur., no. 5, pp. 65–74, May 2024, doi: 10.52710/cfs.867.

25. M. Kari, "Intelligent Deep Learning-Based System for Improved Phishing Identification Accuracy in Web Platforms," in 2026 IEEE International Conference for Convergence in Computing Technology (I3CTCON), IEEE, Mar. 2026, pp. 1–6. doi: 10.1109/I3CTCON68242.2026.11508030.

26. S. Irfan, "Enhancing Email Security Through Accurate Phishing Detection Using Deep Transformer Models," in 2026 World Conference on Computational Science and Technology (WcCST), IEEE, Mar. 2026, pp. 239–244. doi: 10.1109/WcCST67302.2026.11495864.

27. S. Chandrappa, S. Paheding, and Y. Cai, "Leveraging Large Language Models for Automated Detection of Cookie and Session Management Vulnerabilities," in 2025 Northeast Section Conference Proceedings, ASEE Conferences, 2025. doi: 10.18260/1-2--55022.

28. R. Lingam, "Integrating Trustworthiness Into the AI Lifecycle (Trustworthiness)," in AI Safety and Preventing Harm in AI Systems, IGI Global Scientific Publishing, 2026, pp. 213–248. doi: 10.4018/979-8-3373-6935-8.ch008.

29. S. K. Sarangi, R. Lenka, J. Mishra, R. Sahu, and A. Nanda, "Malicious detection and trust calculation using residual recurrent neural network for trust with quality of service-aware multicast routing in mobile ad-hoc network system," Eng. Appl. Artif. Intell., vol. 161, p. 112130, Dec. 2025, doi: 10.1016/j.engappai.2025.112130.

30. A. Blanco-Justicia and J. Domingo-Ferrer, "Machine Learning Explainability Through Comprehensible Decision Trees," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2019. doi: 10.1007/978-3-030-29726-8_2.

31. B. Singh, M. Augie, H. Singh, and T. Banerjee, "Strengthening Modern IAM Authentication with Quantum Cryptography and Anti-Phishing Techniques," Sarcouncil J. Eng. Comput. Sci., vol. 04, no. 10, pp. 1–8, 2025, doi: 10.5281/zenodo.17260292.

32. M. T. Ribeiro, S. Singh, and C. Guestrin, "'why should i trust you?' explaining the predictions of any classifier," in NAACL-HLT 2016 - 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Demonstrations Session, 2016. doi: 10.18653/v1/n16-3020.

33. S. M. Lundberg and S. I. Lee, "A unified approach to interpreting model predictions," in Advances in Neural Information Processing Systems, 2017.

34. S. Kukar, "Explainable AI Models for Transparent and Ethical Customer Relationship Management Decision," in 2026 IEEE 2nd International Conference on Secure IoT, Assured and Trusted Computing (SATC), IEEE, Mar. 2026, pp. 1–7. doi: 10.1109/SATC69565.2026.11542554.

35. 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, no. February, pp. 61611–61640, 2025, doi: 10.1109/ACCESS.2025.3555926.

36. A. Goldstein, A. Kapelner, J. Bleich, and E. Pitkin, "Peeking Inside the Black Box: Visualizing Statistical Learning With Plots of Individual Conditional Expectation," J. Comput. Graph. Stat., 2015, doi: 10.1080/10618600.2014.907095.

37. P. Sen and M. Chowdhury, "Generalizable Detection of Document-Based Phishing Malware: An Explainable Machine Learning Approach on CIC-Trap4Phish," in 2026 IEEE 2nd International Conference on Quantum Photonics, Artificial Intelligence & Networking (QPAIN), IEEE, Apr. 2026, pp. 1–6. doi: 10.1109/QPAIN69676.2026.11546321.

**38.** M. S. Mia, I. T. Moon, N. Hasan, and M. S. R. Jahin, "MalXAI: Explainable Malware Detection Using Dynamic Graph-Based Data with Machine and Deep Learning," in 2026 IEEE 2nd International Conference on Quantum Photonics, Artificial Intelligence & Networking (QPAIN), IEEE, Apr. 2026, pp. 1–6. doi: 10.1109/QPAIN69676.2026.11546230.

**39.** A. Alomar, A. AlJarullah, and S. Abu-Ghazalah, "Permissions-based Android malware detection using machine learning," Neural Comput. Appl., 2025, doi: 10.1007/s00521-024-10950-4.

**40.** A. Mahato, R. Majumdar, and S. K. Ghosh, "Feature-Driven Malware Detection using Cascade Machine Learning Models," SN Comput. Sci., 2025, doi: 10.1007/s42979-025-04342-1.

**41.** V. Sonawane, P. S. Patwal, and V. S. Wadne, "Android Malware Detection and Classification with Analysing Permission API's using Recurrent Neural Network," J. Inf. Syst. Eng. Manag., vol. 10, pp. 454–466, 2025, doi: 10.52783/jisem.v10i10s.1408.

**42.** R. Hilabi and A. Abu-Khadrah, "Windows operating system malware detection using machine learning," Bull. Electr. Eng. Informatics, 2024, doi: 10.11591/eei.v13i5.8018.

**43.** A. Patel and S. M. Ghosh, "AMD-XAI-ML: Android Malware Detection based on an Explainable AI using Machine Learning for Smart Computing Environment," in 2024 OPJU International Technology Conference on Smart Computing for Innovation and Advancement in Industry 4.0, OTCON 2024, 2024. doi: 10.1109/OTCON60325.2024.10687629.

**44.** S. K. Smmarwar, G. P. Gupta, and S. Kumar, "XAI-AMD-DL: An Explainable AI Approach for Android Malware Detection System Using Deep Learning," in Proceedings - 2023 IEEE World Conference on Applied Intelligence and Computing, AIC 2023, 2023. doi: 10.1109/AIC57670.2023.10263974.