International Journal for Multidisciplinary Research (IJFMR)

E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

Review On Ensemble Based Learning for Malware Detection: A Review

Paramjeet Kaur[1], Dr. Vijay Laxmi[2]

[1]Research Scholar in Computer Applications, Guru Kashi University Talwandi Sabo Bathinda
[2]Dean UCCA Guru Kashi University Talwandi Sabo Bathinda

Abstract
Malware analysis becomes one of the main issues in today's world since attackers are producing a variety of malwares, and even their characteristics are updating at an incredibly fast rate. Malware has to be discovered before it impacts a lot of systems to safeguard computer systems and the internet from them. Several types of research on malware detection techniques have recently been conducted. But it is still difficult to identify malware. For identifying malware, there are essentially two methods: One is an identification method that is based on signatures, and the other is dependent on behaviour. The behaviour-based strategy may detect new and complicated malware to some degree utilizing machine learning and other techniques, but it is a difficult one. Signature-based is rapid and effective just for detecting known malware. In this paper, various techniques of malware detection are reviewed and analysed in terms of certain parameters.

Keywords: Malware detection, Machine Learning, Feature extraction, Classification

1. Introduction

Attacks by malware have considerably turned into a menace to contemporary internet security. These programs can easily obtain sensitive information, such as banking account details, login credentials, messages, and contacts. Furthermore, if anything goes wrong, they can damage the data's integrity and accessibility. Recent security reports have shown that malware is the most common method of cyber-attack used by hackers, and ransomware families are the most dangerous. The detection and removal of these attacks from the damaged systems are very difficult [1].Due to the rapid malware attacks on the Internet, it has become crucial to invent advanced software for malware detection to be able to protect the cyberspace as well as to save money from such attacks. Problems related to inadequateness in analytical processes, performance precision, and techniques which are incapable of detecting unknown malware types have made classification and the differentiation of malware one of the major issues throughout the years.Digital learning platforms have opted for ML algorithms in the defence of internet-connected devices against malware. Different ML frameworks have been used, studied, and developed to accomplish the task of malware classification into specific families by utilizing the features obtained from both static and dynamic analyses of the malicious software.

1.1 Malware Detection Process

Figure 1 elucidates the overall malware detection process. The analysis of the malware and the detection of malware are the two stages that are included in the process of malware detection [2].

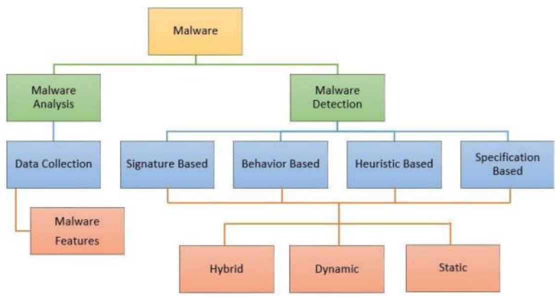

Figure 1: The overall process of malware detection [2]

1.1.1 Malware Detection

Researchers classify malware detection techniques in four categories: behaviour-based, heuristic-based, specification-based, and signature-based. These techniques are used to identify, detect, and counteract malware such that the computer systems do not end up simulating the loss of data and resources.

i.Signature-based: Most of the antivirus software uses signature-based techniques to find infected files. The antivirus software deconstructs the compromised file's code and looks for a pattern that is analogous to a family of malware. A signature consisting of a series of bits is injected into the code when malware is created, which can be used to identify its family. To compare malware signatures during detection and to keep them in a database is also the case. This method of detection is also referred to as pattern matching, string, or pattern scanning. The method can be dynamic or static or a hybrid combining both [3].

ii. Behaviour-based: Behaviour of malware is the reason for behaviour-based detection. A behaviour-based approach solves the signature-based technique's weakness. The main advantage of this method is that it is able to discover zero-day malware. Nonetheless, if not all the possible malware scenarios are examined thoroughly, this method can cause a lot of false positives.

iii. Heuristic-based: Heuristic-based detection makes it possible to identify the known and unknown malware attacks through the detection of a system's typical or atypical activity. The heuristic detection method is divided into two sections. The first step is to monitor the system's functioning without being impacted by an attack and simultaneously keeping a record of the data that is important and can be verified by subsequent checking. This difference is used to the second stage to track down malware belonging to a certain family.

iv. Specificationbased: Specification-based detection approaches further include monitoring applications and analysing them to detect non-normal and abnormal behaviour which is based on system specifications that are already predefined. A major difference between specification and heuristic-based detection is the way they work. Heuristic-based detection uses machine learning and artificial intelligence to learn the normal or suspicious activities that are present in legitimate programs, on the other hand [4], specification-based detection is the evaluation of the system behaviour that is described in its specifications. This method includes manual comparisons of the system's signature activities to the expected behaviours as specified in the specifications.

1.1.2 Malware Detection Analysis

Malware analysis is the very first step in understanding if the software is malicious or not. Devs should incorporate proper methods to protective malware that has been reflected for its functionality and purpose. Depending on the time of use and the technique applied, malware analysis can be of three types: static, dynamic and hybrid.

i.Static analysis:Static analysis, which is the inspection of the structure of an executable file without act-

ually running it, relies on some attributes such as distinct sections and memory layout. Such analysis is done in two phases [5]: basic and advanced. Basic static analysis is the technique that is used to explain the properties of the malware, for example, by providing information such as file size, type, and header information, to give you a quick understanding of the malware. Hundreds of different tools are used in this stage, for example, to collect and analyse and measure these attributes. After basic analysis, advanced static analysis goes to the next level by investigating the malware's behaviour and using more effective tools that first analyse commands to find out how the program works in detail. What is more, static analysis has some limitations when it comes to the detection of the obfuscated malware, because it hardly assesses the packed or encrypted samples.
ii. Dynamic analysis: Dynamic analysis is the process of running malware to see how it behaves and thereby, to let the analysts understand how it works. This is to ensure the machine does not fall victim to the malware, hence the analysis is done in a controlled environment, for instance, a virtual machine or sandbox [6]. Dynamic analysis is broken down into two parts: the basic and the advanced. In basic dynamic analysis, the monitoring tools are used to track the activities of the malware, while the advanced dynamic analysis uses the debuggers to execute commands line by line, giving the analysts the possibility to change the parameters and variables.Throughout the time, the malware is granted absolute authority to use all system resources at will; when the execution is done, the environment goes back to its state as depicted at the beginning of the process. An agent located in the isolated environment is responsible for recording the malware's behaviour. As opposed to static analysis, dynamic analysis can recognize obfuscated malware as well as newly developed threats.
iii. Hybrid Analysis: Hybrid analysis connects two types of two techniques, static and dynamic, in order to find and analyse malware. In the beginning, it uses static analysis to check the malware's code and structure and see if it works or not. Then, the dynamic analysis technique is applied to get more information about the malware's movements. Through this hybrid process, the limitations of only using static or dynamic analysis are coped with, thus providing a more ample insight into the functionality of the malware. By having the two methods side by side, the hybrid analysis improves the whole process of detecting and analysing, therefore, making it easier and more comprehensive for the identification of malware.

1.2 Machine learning For Malware detection and classification

It can be said that ideology of malware developers mainly is to invade computer networks and infrastructure with the sole aim of stealing confidential data, extorting money or demonstrating their capabilities [7]. Traditional malware detection methods have been very effective in the identification of already known threats. However, they found it more difficult to block malware drafted from new and emerging techniques. Advanced techniques in machine learning have contributed significantly to the detection accuracy of models used for malware classification. Malware detection by means of machine learning consists in the following two main steps: first extracting features from the input data and selecting the most relevant ones that effectively represent the data, then performing classification or clustering. Figure 1 shows the machine learning process for malware detection

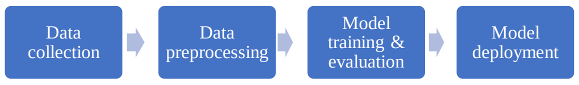

Figure 2: Machine learning pipeline for malware detection [7]

i.Data collection:The collection of data that contributes to a machine learning project is indeed the first as well as the most important step in the machine learning pipeline as it has a straight influence on the model's behaviour. A common principle in machine learning, "garbage in, garbage out," highlights the fact that the quality of the input data accurately describes the quality of the model's output. When the training data is poor, wrong, incomplete, or biased, the model will probably perform even worse than before, no matter the model's complexity, the intelligence of the data scientists, or the resources spent [8]. The model's training data contains only the patterns provided;thus, it is of utmost importance to strictly collect and validate the data before it is deployed in machine learning models.In the scenario of malware detection, data collection refers to the process of getting hold of and labelling both benign and malicious executables. Commercial Windows software is under copyright protection, therefore, the distribution of it without proper authorization might result in legal issues. To this end, researchers often collect benign data from uninfected copies of different versions of Windows operating systems (e.g., Windows XP, Windows 7, and Windows 10) and from various software package managers such as Chocolatey and Windget.

ii. Data Preprocessing: The second stage of data preprocessing is to carry the raw binaries out to the suitable format for analysis [9]. This stage incorporates the extraction of information from different sources such as the hexadecimal representation, the assembly language code, the file metadata and the sequences of API calls that are visible in dynamic analysis. The purpose of data preprocessing is to make raw data more intelligible to machine learning models.Datapreprocessing is the main process in this step and it is done in machine learning based malware detection. Data preprocessing is the activity of preparing and cleaning data so that it may be used for analysis and can generally be classified into two categories: static and dynamic approaches.

- Static Analysis: This involves analyzing programs without the use of execution. It consists of a study of the Portable Executable (PE) structure, headers, dynamic library references, and sections. The same can be done for disassembling programs so as to give the assembly language source code which, in turn, gives an insight into low-level instructions and memory interactions.
- Dynamic Analysis: This is the procedure of executing the software in a controlled environment, which enables the supervisor to keep track of its behaviour. However, if static analysis is no longer enough because of obfuscation, packing, or other limitations, dynamic analysis comes into play. The latter makes it clearer to see the malware's actual functionality since the program is tracked while it is being executed [10].

Machine learning-based models for malware detection have been classified into static and dynamic detectors according to the type of analysis they perform.

iii. Model Training and Model Evaluation: The process of model training and evaluation involves choosing the most suitable machine learning algorithm, training the model, and assessing its performance.

- Model selection: The job of choosing the most appropriate machine learning algorithm for the learning task is the function of this pipeline stage. The particular type of algorithm that is most effective would vary based on the characteristics of the data being passed through. A spatial relationship, for example, is better suited to convolutional neural networks than Support Vector Machines when the input to the neural network contains a sequence of bytes. The primary advantage of using decision trees over neural networks is their ability to be non-interpretable. This means that PE file input data consisting of a collection of features is a better candidate for decision trees [11].

- Training the model:In this part, the chosen model or models are trained with the training data set and their hyperparameters are tuned.
- Model evaluation: Many things first of all must be done right - tuning the model's parameters is one of them, and also, ensuring its transferability is another part of the task. This consists of evaluating the model using the validation set to determine whether it is able to generalize well to unseen data. In regard to malware detection, the system predicts whether the input file is malicious or benign; hence, there are four possible scenarios. A false negative occurs when the model says the file is malware despite it being innocent. A false positive would be when the file is benign and the model claims it is malware. A true negative would occur when the file is benign and the model says it is benign. Hence, these are the four possible outcomes of a given scenario.In these kinds of situations, striving only for precision can be misleading as the classifier can get a high score simply by predicting the majority class all the time. The F1 score is a critical indicator in such cases. The F1 score is a model's performance score that includes both recall and precision to give a more objective assessment of a model's performance.

iv. Model Deployment, Monitoring and Maintenance: Model deployment entails deploying a model for its intended purpose [12]. This although means the incorporation of the trained model into a system that performs real-time malware detection. Model monitoring and maintenance consist of conducting regular performance tests of the model in the deployed environment to identify whether there are any changes in performance and to periodically refresh the model with new benign and malicious examples to adapt to new patterns and maintain its effectiveness.

1.3 Machine learning models for Malware detection

A method that makes the program to self-evolve based on recent input data is compulsory for each machine learning algorithm to produce optimum results. This mechanism implements the system to learn more and more. Subsequently are the incompetently popular machine learning algorithms of suitable users for malware detection [13].

i.Support Vector Machine (SVM): SVM algorithms establish one or multiple surrogate hyperplanes for the high-dimensional space or even the infinite-dimensional space to perform data segmentation. In the end, the hyperplane is responsible for the optimal partition which has the greatest distance from the nearest points belonging to different classes and is known as the margin. A high-value margin infers the low probability of a misclassification.

ii. K-Nearest Neighbour (KNN): KNN was mainly used in classification tasks but it is also important in malware detection for its interpretability, computational speed, and prediction capabilities as noted in our study. KNN can help with planning and regression issues but it is used in our context to classify the malware by identifying the class that the input is most similar to. The model finally classifies the data into the two classes: whether the malware is present or not.

iii. Naïve Bayes (NB): Naive Bayes (NB) is a classification algorithm that overrides normal distribution to analyze the relationship between two or more variables using probability. It computes the chance of data existing in a particular category by employing the principles of probability. The first step in the training phase consists of providing the system with a set of data and appropriate class labels for each data[14]. As for new test data, the system uses the previous learned probabilities to get the highest probable category for the test data by performing probability operations on the trained data.

iv. Decision Tree (DT): A decision tree (DT) is represented as a rooted node, branches, and leaves as the last nodes. One node stands for a test attribute, each branch denotes the possible outcome of a test, and each leaf is a class label. DTs are not designed for heavy training but are based on the concept of data entropy which is easy to grasp. A learning phase and a decision tree classification phase are the two simple and efficient phases of a decision tree.
v. Random Forest:The Random Forest borrowed its name from the collection of tree predictors, which are used to tackle both classification and regression tasks. The step of the Random Forest algorithm consists of the following: At first, the feature data vector is put into the random forest classifier. But, to begin, every tree in the forest classifies the data. The final output is decided by the label or the class that gets the most votes from the trees. In regression tasks, the algorithm is first averaging the values that are predicted by all the trees. The trees are trained in the same way, but they are on the different training datasets [15].

1.4 Ensemble Learning

Ensemble methods rely on the principle of generating many sets of classifiers through the readjustment of whole training datasets with the help of some techniques like resampling or the reweighting. Each of the base classifiers is uniquely framed from a different version of the dataset and a new ensemble classifier is constructed using the stacked ensemble method. The strategy enables the amalgamation of various base classifier predictions, with a new model that gradually learns how to better merge these predictions. Thus, this method attempts to deal with the problem concerning the high false positive and false negative rates in the machine learning system by bringing about a general increase in accuracy through the different techniques such as Bagging, Boosting, and Stacking.
i.Bagging: Bagging, or Bootstrap Aggregating, is a process of creating several different models of a single learning algorithm by taking random samples from the subsets of the training dataset with replacement. To combine the predictions from these models, two methods are commonly used: majority voting and averaging [16]. In the case of voting, the model that has the most votes from the classifiers is the one that is selected as the final prediction. In averaging, the outcome is obtained by summing up all the predictions of the classifiers and dividing them by the number of classifiers.
ii. Boosting: This method is used for improving the predictions of the model. The boosting technique chooses the samples that are wrongly predicted and reinforces their weights. Boosting, in a nutshell, is a slight twist on bagging. The first step in boosting is to distribute the weights equally among all the instances. Teach the classifiers to make predictions on wrongly classified instances, then adjust the weights of the incorrectly predicted instances. Lastly, take the weighted mean of all weak learners to form a strong learner which is your final model. Many boosting algorithms like AdaBoost, Gradient Tree Boosting, and XGBoost are available.
iii. Stacking: Stacking, or stacked generalization, is a method that combines several classifiers which are produced by different machine learning algorithms. The first phase includes the training of each algorithm with the training data. In the second phase, the models' predictions are used as input for a second-level model, which is trained to make final predictions on the test data.

2. Literature review

S. A. Roseline, (2019) developed a Malware author’s new variants by applying polymorphic and evasion techniques to existing malware, which made them harder to detect. To solve this problem, malware

patterns were identified using a vision-based method that detected and visualizes these patterns in images and characterizing their features [17]. This paper presented a hybrid stacked multi-layer integration method that proved to be more powerful and efficient than deep learning models. The proposed model achieved an impressive accuracy of 98.91%, surpassing both traditional machine learning and deep learning methods. It was well-suited for both small-scale and large-scale datasets due to its adaptive nature, which automatically adjusted the number of sequential layers. Additionally, this method was computationally efficient and required fewer hyperparameters than deep neural networks.
M. J. Hussain, et.al (2022) projected a behavioural malware detection method using machine learning classifiers for PE (Portable Executable) data [18]. The system was trained and tested using the BODMAS (Blue Hexagon Open Dataset for Malware Analysis), which contained samples from August 2019 to September 2020. This approach was divided into two stages: The first stage used a binary classifier (specifically random forest) to determine if the PE profile was bad. The second stage involved identifying malware families according to various probing methods. This classifier included KNN (K-nearest neighbour), SVM (support vector machine), Random Forest, Decision Tree, and Gradient Optimization with equal weight for each classifier. This proposed method achieved good results with 99.48% accuracy at binary classification level and 92.49% accuracy at malware family classification level of BODMAS dataset
J. Wang, T. Yang, et.al (2022) presented a novel MCES (multi-classifier ensemble system) using behaviour analysis for malware detection [19]. The MCES model adopted a hierarchical structure, and its main components include a base classifier and a meta-classifier. To test its performance, the model was evaluated on 14,800 malware samples and 14,800 good samples. The results showed that MCES outperforms existing machine learning methods in detecting malware based on an API (application program interface) called session. More importantly, the MCES mode achieved the highest accuracy of 97.54% and the highest recovery of 97.85% compared with the state-of-the-art deep learning-based malware detectors.
M. N. Al-Andoli, (2023) proposed an ensemble-based parallel DL (deep learning) classifier for malware detection. More specifically, they developed a joint learning method that combined five deep learning models with neural networks based on meta models [20]. These deep learning models were trained and optimized using an optimal method combining BP and particle swarm optimization PSO (Particle Swarm Optimization) algorithms. They leverage a parallel computing framework to improve scalability and efficiency. This combination was evaluated on five malware datasets (Drebin, NTAM, TOP-PE, DikeDataset, and ML_Android) and achieved 99.2%, 99.3%, 98.7%, 100%, and 100% accuracy, respectively. Moreover, using the same processing time effectively increases the computational speed, resulting in a performance increase of up to 6.75x. These results confirmed that the proposed integration was both efficient and effective, outperforming many existing malware detection methods.
H. Zhu, Y. et.al (2023) investigated the limitations of existing methods to improve malware detection, such as considering each feature separately and relying on duplicate data [21]. To solve these problems, they proposed a multi-system integration model called MEFDroid, which used deep learning-based inference techniques together with predictors to learn the details of the raw materials. They also adopt a novel fusion strategy to integrate multiple representation methods and relationships to utilize the original data. In order to evaluate the effectiveness of the MEFDroid framework, they conducted a series of benchmark tests to determine how our proposed systems (ESAES, EDAES, and EDAFS) progressively improved malware detection. They also compared these algorithms with classical machine learning

methods and conventional sampling. They tested the reliability of this method using other demographic data. This extensive testing showed that the EDAFS model in MEFDroid outperforms other models in most benchmarks, proving it to be an effective solution for Android malware detection.

H. Naeem, et.al (2023) presented a malware detection method had significant limitations, leading to the development of other dynamic platform-friendly methods to debug hardware issues [22]. This approach involved converting data collected from process memory dumps into images and performing composite modeling and texture analysis. The combination of local and global descriptors improved the data training ability of deep hybrid models by reducing input dimensions. The model combined the predicted results of weak learners such as CNNs and feeds them into a MLP (meta-learner) for further learning. Descriptive cognitive techniques were also used to interpret and validate the results. This concept was evaluated on three databases: Dumpware10 (3686 samples from 10 malware families), CIC-MalMem-2022 (2916 samples from 15 obfuscated malware families), and Real Knowledge of malware samples and 2375 samples of benign Android apps. The test results showed that the system achieved 99.1% accuracy of Windows malware memory dumps, 94.3% accuracy of Android malware memory dumps, and 99.8% accuracy of Windows obfuscated malware memory dumps, indicating its benefits in preventing malicious schemes.

P. Bhat, S. Behal, et.al (2023) presented an accurate dynamic analysis method to identify various malicious attacks. This method focused on analyzing the behavior of malware, especially on reproducing the behavior of Android malware [23]. The identified behavioral features include call, binding, and complex Android devices, which together represent hybrid behavior. A task selection process was applied to remove irrelevant features to improve malware detection and classification. Homogeneous and heterogeneous ensemble machine learning algorithms were used for classification. Among these, the stacking method has the highest distribution rate of 98.08%. Rigorous tests demonstrated the effectiveness and superiority of the model.

Saddam Hussain Khan et al., (2023) developed a novel malware detection framework called DSBEL, which used a novel SB-BR-STM (Squeezed-Boosted Boundary-Region Split-Transform-Merge) CNN and integrated the work. The STM block of this framework utilized various methods of extended convolutions, boundary and region operations to capture both homogeneous and heterogeneous global patterns [24]. In addition, the enhancement of initial and final level by transition learning and combination of various methods were utilized to identify the changing pattern. Deep SB-BR-STM CNN extracted discriminative features, which were then embedded into the classifier (SVM, MLP and AdabooSTM1) to improve the learning hybrid generalization ability. They evaluated the performance of the DSBEL framework and SB-BR-STM CNN using the IOT_Malware dataset and performance benchmarks. The results showed good performance with 98.50% accuracy, 97.12% F1-Score, 91.91% MCC, 95.97% recall and 98.42% accuracy. The framework proved to be powerful and useful for exploring terrorism opportunities and providing insights for future strategies.

D. K. Ghaghre, G. P. Gupta et.al (2024) presented a comprehensive study on malware detection utilized various machine learning techniques [25]. It investigated the use of Adaboost ensemble learning, stacked ensemble learning, hard voting ensemble learning, and soft voting ensemble learning to solve the challenge of malware classification. Additionally, this paper explored feature engineering methods, specifically the Variance Threshold and wrapper-based forward selection, to extract relevant features from malware samples for effective detection. The study applied these malware detection techniques to the CCCS-CIC-AndMal-2020 dataset and achieved an accuracy of 99.48%.

Pascal Manirihoet.al (2024) introduced an improved dataset with temporal attributes, which allowed the evaluation of the accuracy of memory-based malware detection strategies based on the concept of drift (temporal data partitioning) [26]. MeMalDet uses deep autoencoders to extract the best features from the memory dump in an unsupervised manner, eliminate the need for manual processing. A group of quarantine inspectors then performed the actual malware detection. Extensive testing on our largest public database showed that MeMalDet maintains high performance in detecting obfuscated malware in time partitioning. They achieved up to 98.82% accuracy and 98.72% F1 scores in previous analyses without detecting advanced obfuscated malware, indicating a significant improvement in malware detection based on state-of-the-art identification. The improved data enabled periodic quality analysis and represented new sources. MeMalDet effectively combined the advantages of representation learning and supervised machine learning to integrate and detect malware over time through memory analysis. This research provided new capabilities to detect modern malware and address evolving global threats.

2.1 Comparative analysis of exiting malware detection and classification techniques

Author & Year	**Technique used**	**Dataset**	**Evaluation metrics**	**Accuracy**	**Key features**	**Limitations**
S. A. Roseline, 2019	Hybrid stacked multi-layer integration	Real-time dataset	Accuracy	98.91%	Adaptive model with fewer hyperparameters than deep learning models. Suitable for both small-scale and large-scale datasets. Computationally efficient.	High dependence on high-quality training data
M. J. Hussain et al., 2022	Behavioral malware detection with ensemble learning	BODMAS dataset	Accuracy	99.48% (binary), 92.49% (family)	Two-stage method: binary classification and malware family identification. Equal weight assigned to multiple classifiers.	Limited to the BODMAS dataset
J. Wang et al., 2022	Multi-classifier ensemble system (MCES)	Real time dataset containing 14,800 malware and 14,800 good	Accuracy, Recovery	97.54%	Hierarchical structure with a base and meta-classifier. Outperforms state-of-the-art	Challenges in generalizing to new malware types.

		samples			malware detectors.	
M. N. Al-Andoli, 2023	Ensemble-based parallel deep learning classifier	Drebin, NTAM, TOP-PE, DikeDataset, ML_Android	Accuracy, Recovery	Up to 100%	Combines 5 deep learning models with PSO for optimization. Improved computational speed and performance by 6.75x.	Complexity in combining multiple deep learning models
H. Zhu et al., 2023	Multi-system integration model (MEFDroid)	Real-time dataset	Accuracy	98.4%	Novel fusion strategy integrating multiple representations and relationships for raw data. Outperforms classical machine learning methods.	Considered limitations of existing methods but specifics not detailed
H. Naeem et al., 2023	Dynamic analysis using image conversion and hybrid models	Dumpware10, CIC-MalMem-2022, Real Knowledge of malware samples	Accuracy	99.1% (Windows), 94.3% (Android), 99.8% (obfuscated)	Combines local and global descriptors with CNNs and MLPs for accurate malware detection.	May not address all dynamic platform issues
P. Bhat et al., 2023	Homogeneous & heterogeneous ensemble machine learning	Real-time dataset	Distribution rate, Accuracy	98.08%	Focuses on analyzing Android malware behavior with feature selection to improve detection accuracy.	Focused on Android malware behaviour, might not generalize to all types of malware

Saddam Hussain Khan et al., 2023	Novel DSBEL framework with SB-BR-STM CNN	IOT_Malware dataset	Accuracy, F1-Score, MCC, Recall	98.50%	Uses Squeezed-Boosted Boundary-Region Split-Transform-Merge CNN. Extracts homogeneous and heterogeneous global patterns.	Complexity in the integration of multiple methods
D. K. Ghaghre et al., 2024	Various ensemble learning techniques	CCCS-CIC-AndMal-2020	Accuracy	99.48%	Explores various ensemble learning techniques and feature extraction methods for malware classification.	May not address all feature engineering challenges
Pascal Maniriho et al., 2024	Deep autoencoders and stacked ensemble	Largest public database	Accuracy, F1-Score	98.82%	Detects obfuscated malware through memory dump analysis. Combines representation learning and supervised machine learning.	Focused on memory-based detection and may not address all advanced obfuscation techniques

Conclusion

In conclusion, malware analysis has become a critical issue as attackers continuously evolve their methods, creating new forms of malware with increasingly sophisticated characteristics. Detecting malware before it can cause widespread harm is crucial to protecting computer systems and the internet. While substantial research has been conducted on malware detection techniques, accurately identifying malware remains a significant challenge.There are two primary methods for malware detection: signature-based and behavior-based. Signature-based detection is fast and efficient but only effective against known malware. In contrast, behavior-based detection, which leverages machine learning and other advanced techniques, has the potential to identify new and complex malware. However, it remains a difficult approach to implement.

References

1. K. Xu, Y. Li and R. H. Deng, "ICCDetector: ICC-Based Malware Detection on Android," in IEEE Transactions on Information Forensics and Security, vol. 11, no. 6, pp. 1252-1264, June 2016
2. M. Sun, X. Li, J. C. S. Lui, R. T. B. Ma and Z. Liang, "Monet: A User-Oriented Behavior-Based Malware Variants Detection System for Android," in IEEE Transactions on Information Forensics and Security, vol. 12, no. 5, pp. 1103-1112, May 2017
3. Z. Yuan, Y. Lu and Y. Xue, "Droiddetector: android malware characterization and detection using deep learning," in Tsinghua Science and Technology, vol. 21, no. 1, pp. 114-123, Feb. 2016
4. A. Guerra-Manzanares, M. Luckner and H. Bahsi, “Android malware concept drift using system calls: Detection, characterization and challenges”, Expert Systems with Applications, vol. 12, no. 5, pp. 859-867, 21 April 2022
5. P. Faruki et al., "Android Security: A Survey of Issues, Malware Penetration, and Defenses," in IEEE Communications Surveys & Tutorials, vol. 17, no. 2, pp. 998-1022, Secondquarter 2015
6. Q. Han, V. S. Subrahmanian and Y. Xiong, "Android Malware Detection via (Somewhat) Robust Irreversible Feature Transformations," in IEEE Transactions on Information Forensics and Security, vol. 15, pp. 3511-3525, 2020
7. S. H. Moghaddam and M. Abbaspour, "Sensitivity analysis of static features for Android malware detection," 2014 22nd Iranian Conference on Electrical Engineering (ICEE), 2014, pp. 920-924
8. G. Canbek, S. Sagiroglu and T. TaskayaTemizel, "New Techniques in Profiling Big Datasets for Machine Learning with a Concise Review of Android Mobile Malware Datasets," 2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT), 2018, pp. 117-121
9. J. H. Abawajy and A. Kelarev, "Iterative Classifier Fusion System for the Detection of Android Malware," in IEEE Transactions on Big Data, vol. 5, no. 3, pp. 282-292, 1 Sept. 2019
10. M. Samara and E. -S. M. El-Alfy, "Benchmarking Open-Source Android Malware Detection Tools," 2019 2nd IEEE Middle East and North Africa COMMunications Conference (MENACOMM), 2019, pp. 1-6,
11. Y. Xue et al., "Auditing Anti-Malware Tools by Evolving Android Malware and Dynamic Loading Technique," in IEEE Transactions on Information Forensics and Security, vol. 12, no. 7, pp. 1529-1544, July 2017
12. Y. Zhang et al., "Familial Clustering for Weakly-Labeled Android Malware Using Hybrid Representation Learning," in IEEE Transactions on Information Forensics and Security, vol. 15, pp. 3401-3414, 2020
13. R. B. Hadiprakoso, I. K. S. Buana and Y. R. Pramadi, "Android Malware Detection Using Hybrid-Based Analysis & Deep Neural Network," 2020 3rd International Conference on Information and Communications Technology (ICOIACT), 2020, pp. 252-256
14. S. Liang and X. Du, "Permission-combination-based scheme for Android mobile malware detection," 2014 IEEE International Conference on Communications (ICC), 2014, pp. 2301-2306
15. E. C. Bayazit, O. KoraySahingoz and B. Dogan, "Malware Detection in Android Systems with Traditional Machine Learning Models: A Survey," 2020 International Congress on Human-Computer

Interaction, Optimization and Robotic Applications (HORA), 2020, pp. 1-8

16. J. Wu and A. Kanai, "Utilizing obfuscation information in deep learning-based Android malware detection," 2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC), 2021, pp. 1321-1326
17. S. A. Roseline, A. D. Sasisri, S. Geetha and C. Balasubramanian, "Towards Efficient Malware Detection and Classification using Multilayered Random Forest Ensemble Technique," 2019 International Carnahan Conference on Security Technology (ICCST), Chennai, India, 2019, pp. 1-6, doi: 10.1109/CCST.2019.8888406.
18. M. J. Hussain, A. Shaoor, S. Baig, A. Hussain and S. A. Muqurrab, "A hierarchical based ensemble classifier for behavioral malware detection using machine learning," 2022 19th International Bhurban Conference on Applied Sciences and Technology (IBCAST), Islamabad, Pakistan, 2022, pp. 702-706, doi: 10.1109/IBCAST54850.2022.9990203.
19. J. Wang, T. Yang, B. Yan, P. Yao, W. Wang and Q. Yang, "MCES: Multi-classifier Ensemble System for Malware Detection and Identification," 2022 IEEE 6th Conference on Energy Internet and Energy System Integration (EI2), Chengdu, China, 2022, pp. 1565-1570, doi: 10.1109/EI256261.2022.10117041.
20. M. N. Al-Andoli, K. S. Sim, S. C. Tan, P. Y. Goh and C. P. Lim, "An Ensemble-Based Parallel Deep Learning Classifier With PSO-BP Optimization for Malware Detection," in IEEE Access, vol. 11, pp. 76330-76346, 2023, doi: 10.1109/ACCESS.2023.3296789.
21. H. Zhu, Y. Li, L. Wang, and V. S. Sheng, “A multi-model ensemble learning framework for imbalanced android malware detection,” Expert Systems with Applications, vol. 234, pp. 120952–120952, Dec. 2023, doi: https://doi.org/10.1016/j.eswa.2023.120952.
22. H. Naeem, S. Dong, Olorunjube James Falana, and F. Ullah, “Development of a deep stacked ensemble with process based volatile memory forensics for platform independent malware detection and classification,” Expert Systems with Applications, vol. 223, pp. 119952–119952, Aug. 2023, doi: https://doi.org/10.1016/j.eswa.2023.119952.
23. P. Bhat, S. Behal, and K. Dutta, “A system call-based android malware detection approach with homogeneous & heterogeneous ensemble machine learning,” Computers & Security, vol. 130, pp. 103277–103277, Jul. 2023, doi: https://doi.org/10.1016/j.cose.2023.103277.
24. Saddam Hussain Khan et al., “A new deep boosted CNN and ensemble learning based IoT malware detection,” Computers & Security, vol. 133, pp. 103385–103385, Oct. 2023, doi: https://doi.org/10.1016/j.cose.2023.103385.
25. D. K. Ghaghre, G. P. Gupta and S. P. Sahu, "Ensemble Machine Learning and Feature Selection for Effective Malware Detection," 2024 IEEE 13th International Conference on Communication Systems and Network Technologies (CSNT), Jabalpur, India, 2024, pp. 293-299, doi: 10.1109/CSNT60213.2024.10546128.
26. Pascal Maniriho, AbdunNaser Mahmood, and M. Jabed, “MeMalDet: A Memory analysis-based Malware Detection Framework using deep autoencoders and stacked ensemble under temporal evaluations,” Computers & Security, pp. 103864–103864, Apr. 2024, doi: https://doi.org/10.1016/j.cose.2024.103864.