Malware Classification on EMBER 2024 with 5-Fold and Temporal Evaluation: A Hybrid SAE–Top-K–LightGBM Model

[1]Sultan Tazefidan, [2]Heya Meylem, [3]Gülay Çiçek

[1,2]Department of Software Engineering, Faculty of Engineering and Architecture
Istanbul Beykent University, Sariyer, Istanbul, Türkiye
[1]sultantazefidan.1@gmail.com, [2]heyameylem96@gmail.com, [3]gulaycicek@beykent.edu.tr

I. Literature Review

In this section, recent studies conducted on the classification of malicious software (malware) are comprehensively examined. The studies are categorized into three main groups based on their methodological approaches: machine learning, deep learning, and hybrid models. For each group, relevant research is presented in summary tables, and the employed methods, datasets, and obtained performance results are analyzed in detail. At the end of the section, current trends and gaps in the literature are evaluated, and the original contributions and distinctive aspects of this study are discussed.

A. *Machine Learning–Based Malware Classification Studies*

This subsection presents malware analysis studies conducted using machine learning techniques, and Table I provides summary information on these works.

Kincl and colleagues (2025) [1] proposed a knowledge graph (KG) embedding + machine learning (ML)–based analysis pipeline using Android application–based malware data. In the study, the widely used CICMalDroid 2020 dataset was employed. The dataset initially contained more than 17,000 Android applications, and after feature extraction, this number was reduced to 13,077 samples. Subsequently, experiments were conducted on a balanced set consisting of 850 benign and 850 malware samples, totaling 1,700 instances.

In the research, 12 different machine learning classifiers—SVM, RF, KNN, MLP, LR, GNB, DT, AB, GB, GP, RG, and PA—were evaluated; the data were split into 80% training (1500 samples) and 20% testing (200 samples). According to the results, accuracy ranged from 85.5% to 96.0% for KG-based representations, from 77.0% to 94.9% for PCA representations, and from 89.7% to 98.4% for BOW representations. KG-based approaches achieved higher accuracy compared to PCA and, in some cases, produced results comparable to or better than the BOW method.

The study can be characterized within the scope of the Android platform, binary classification, and ML-based KG-embedding approaches. However, only static analysis was employed; therefore, there are limitations in detecting previously unseen (out-of-sample) malware. Real-time or dynamic analysis has not yet been applied. In future studies, integrating dynamic analysis and multi-edge structures into the knowledge graph, automatically learning new malware families, and making the model inductive (updatable after training) are planned.

Vasan and colleagues (2025) [2] proposed a machine learning–based model called the Adaptive Embedded Framework (AEF) for detecting malware across different platforms (Windows, Android, and IoT). The study utilized several datasets provided by the Canadian Institute for Cybersecurity (CIC), including CICMalDroid-2020 (Android), MalMem-Obfuscated (Windows, memory-based), Dumpware10 (Windows, memory images), and IoMT-2024 (IoT devices). The sample sizes of these datasets were as follows: CICMalDroid-2020: 17,341 samples, CIC-MalMem-2022: 58,596 samples, and IoMT-2024 and Dumpware10: 4,292 samples each.

The proposed AEF model incorporates embedded feature selection and PCA-based transformations, and it was compared against both classical machine learning classifiers (LR, RF, KNN, XGB) and deep learning models (1D-CNN, Bi-LSTM). According to the experimental results, accuracy values ranged from 54.99% to 96.46% on the CICMalDroid-2020 dataset, from 92.73% to 97.44% on the Dumpware10 dataset, from 98.90% to 99.98% on the MalMem-Obfuscated dataset, and from 89.80% to 98.39% on the IoMT-2024 dataset. Notably, the AEF model achieved the highest accuracy on the Windows-based datasets (Dumpware10 and MalMem), while maintaining strong performance on the Android dataset (CICMalDroid) despite its multiclass structure.

As a limitation of the study, the complexity and limited interpretability of the employed models were highlighted. Additionally, it was noted that feature importance rankings and explainability analyses were not fully explored for some datasets, and that the deep architectures used were not yet fully optimized. Future work aims to improve the interpretability of the AEF model and integrate deeper neural network architectures into the AEF structure to enhance performance in detecting complex and emerging types of malware.

Güngör and colleagues (2023) [3] performed classification on 25 malware families using the Malimg image-based PE dataset. In the study, Android APK files were converted into grayscale images, and features extracted from these images were analyzed and classified using machine learning methods (LR, LDA, KNN, CART, RF, NB, SVM). The results were evaluated using the K-fold cross-validation method. According to the findings, the lowest accuracy was reported as 68.94% (SVM), while the highest accuracy was reported as 97.44% (RF).

In the study, the exclusive use of the Malimg dataset, the low complexity of the model, and the absence of deep learning methods

TABLE I: Summary of Machine Learning–Based Literature Review Results

Authors (Year)	Sample Size	Language (if specified)	Platform	Method	Results	Limitations	Future Work
Kincl et al. (2025) [1]	CICMalDroid 2020, 1700 Samples (Benign: 850, Malware: 850)	Python (implicitly indicated)	Android (Docker)	KG embedding (TransE, DistMult, ComplEx, HoLE, PCA, BOW) + 12 ML classifiers	HoLE: 97.4%, PCA: 94.9%, TransE: 94.0%	Lack of generalizability; static analysis only; undirected edges	Dynamic analysis and directed graph structures will be added; the model will be made generalizable through integration with external knowledge bases.
Vasan et al. (2025) [2]	CICMalDroid-2020: 17,341 samples; CIC-MalMem-2022: 58,596 samples; IoMT-2024 and Dumpware10: 4,292 samples	-	Windows and Android (multi-platform)	LR, AEF — embedded feature selection, stacking, RF, KNN, XGBoost	LR: 54.99%, AEF: 96.46% (for CICMalDroid dataset)	Limited model interpretability; deep architectures not fully optimized; feature contribution analysis not performed for some datasets.	Improving interpretability, integrating feature contribution analysis and visualization tools; planned expansion with deeper neural networks.
Güngör et al. (2023) [3]	Malimg, 50,000 samples (25 classes, 2000 samples each)	Python (sklearn library)	Android (APK)	SVM, RF, LR, LDA, KNN, CART, NB	SVM: 68.94%, RF: 97.44%, LR: 84.22%	Limited data diversity; deep learning methods not applied; low model complexity	Expansion of the dataset and inclusion of deep learning methods.
Islam et al. (2023) [4]	CCCS-CIC-AndMal-2020, 400,000 samples (benign: 200K; malware: 200K)	Python	Android (dynamic)	Ensemble ML, MV	Ensemble ML: 95%, MV: 93.4%	Model exhibits confusion in certain classes (e.g., Riskware–Adware) and shows misclassification rates.	The proposed model is planned to be evaluated on different and more recent datasets to assess its effectiveness.
Rawat et al. (2022) [5]	Emotet dynamic network traffic dataset, 483,782 flows (Emotet: 2,143; non-Emotet: 481,639)	-	Windows	RF, MLP, NB, SMO, LR	RF: 99.9726% precision and 92.3% TPR	Model is limited to current attack variants and may be insensitive to new vectors.	Retraining and optimization of the model are planned.
Zada et al. (2024) [6]	MMCC (approximately 10,000 samples)	Python	Windows	KNN, NB, SGDC, DT	KNN: 93.3%, NB: 99.54%	Issues with data quality, imbalance, feature selection errors, and lack of model generalizability	Hybrid approaches, advanced feature engineering, real-time analysis, low-resource models, and collaboration with cybersecurity experts.
Bakshi et al. (2025) [7]	PE_Header, PE_Section, DLLs_Imported (approximately 29,000 samples)	Python	Windows	KNN, RF, DT	KNN: 90.01%, RF: 98.13%	Deep learning methods not used; limited generalization	Integration of deep learning methods and development of additional hybrid models to improve accuracy.

Note: The table summarizes the key characteristics, methods, performance comparisons, and limitations of machine learning–based malware detection studies.

were identified as the main limitations. It was stated that future work aims to increase the model's accuracy and generalization capability by expanding the dataset and incorporating deep learning–based methods.

Islam and colleagues (2023) [4] proposed a Weighted Voting–based ensemble machine learning model leveraging dynamic features of Android malware using the CCCS-CIC-AndMal-2020 dataset. The study combined six different machine learning algorithms: Random Forest (RF), K-Nearest Neighbors (KNN), Multi-Layer Perceptron (MLP), Decision Tree (DT), Support Vector Machine (SVM), and Logistic Regression (LR). In the architecture, RF and KNN were placed in the base layer, while the second layer produced the final prediction via a weighted voting mechanism based on the outputs of these two models.

In the dimensionality reduction stage, PCA was employed to reduce 141 features to 45 components, thereby lowering the computational complexity of the model. Among the individual models, the lowest accuracy was reported as 57.7% (LR), while the highest accuracy exceeded 90% (RF and KNN). The proposed Weighted Voting Ensemble model achieved 95% accuracy, outperforming both the individual classifiers and the traditional Majority Voting (MV) method (93.4%).

The study noted that despite achieving high overall accuracy, certain classes exhibited false positive and false negative instances, which presents a limitation regarding the model's generalization ability and class discrimination power. Future studies aim to evaluate the proposed method on different and more recent datasets to assess its generalizability and effectiveness.

Rawat and colleagues (2022) [5] conducted a study in which the network traffic generated by the Emotet trojan targeting Windows operating systems was collected through dynamic analysis, and binary classification was performed using this data. The researchers dynamically executed Emotet samples from version 2 to version 187 over a ten-month period, creating a dedicated Emotet dynamic network traffic dataset. The dataset consists of a total of 483,782 network flows (2,143 Emotet flows and 481,639 non-Emotet flows). In the classification stage, machine learning methods such as RF, MLP, NB, SMO, and LR were evaluated. According to the analysis results, the RF classifier achieved 99.9726% precision and a 92.3% true positive rate (TPR). The study highlighted that malware behaviors and infection vectors may change over time, meaning that the developed models may lose their effectiveness in the future. Future work aims to track new attack methods, analyze different trojan families, and monitor changes in features/vectors associated with malware classes.

Zada and colleagues (2024) [6] conducted a study using the Microsoft Malware Classification Challenge (MMCC) dataset to detect malware attacks targeting Windows systems. The dataset contains approximately 10,000 malware samples, and the data were split into 70% training and 30% testing. For binary classification, KNN, NB, SGDC, and DT models were used.

In the preprocessing stage, memory optimization was applied to reduce dataset size and improve memory usage; afterward, Principal Component Analysis (PCA) was employed to reduce model complexity and mitigate overfitting risk.

A two-stage experimental analysis was conducted in the study. In the first stage, the models were tested with default parameters, and accuracy values ranged between 53% and 59%. These results reflect the baseline performance of the models without hyperparameter tuning. In the second stage, parameter optimization was performed using Grid Search and cross-validation, leading to a significant improvement in performance. According to the obtained results, the highest accuracy was 99.54% with the NB model, while the lowest accuracy was 99.3% with the KNN model.

The study's limitations include the fact that the quality and diversity of the training data directly affect model performance, the potential limitation in the model's generalizability, and the likelihood that results obtained under controlled test conditions may degrade in real-world systems. It was also emphasized that incorrectly selected feature extraction or preprocessing techniques may introduce noise or bias into the model, negatively affecting accuracy. Furthermore, the applicability of these models may be restricted in systems with limited hardware resources.

Future work aims to develop new detection approaches, improve feature engineering, integrate hybrid learning methods, and implement real-time analysis systems.

Bakshi and colleagues (2025) [7] conducted a study in which three different datasets available on the Figshare platform — PE_Header, PE_Section, and DLLs_Imported, containing approximately 29,000 Windows-based malware samples — were used for the classification of malicious software targeting Windows systems. Each dataset includes SHA256 values, label information (e.g., "Benign," "RAT," "Downloader," "Spyware," etc.), and DLL/section headers. To address data imbalance, the dataset was balanced to a 50% benign – 50% malware ratio.

In the classification process, three machine learning models — KNN, RF, and DT — were compared within the framework of the Machine Learning-Based Malware Detection and Analysis Mechanism (MLBM-DAM). According to the experimental results, the highest accuracy (98.13%) and highest precision (98.21%) were achieved by the RF model. The lowest performance was observed in the KNN model, with 90.01% accuracy and 88.29% precision.

The limitations of the study include the exclusive use of machine learning–based models and the lack of integration of deep learning methods. It was also noted that the generalization capability of the model is limited.

Future work aims to integrate deep learning–based approaches into the MLBM-DAM system, develop neural network–based hybrid models to improve accuracy, and enhance the overall malware detection performance.

B. *Deep Learning–Based Malware Classification Studies*

In this subsection, literature studies conducted on deep learning–based malware classification are summarized. Comparative results of the relevant studies are presented in Table II.

Gupta and colleagues (2025) [8] focused on the binary classification problem. In the study, API call sequences of Windows PE–based malicious (malware) and benign files were used. The dataset consisted of 2,570 samples obtained from GitHub and an additional validation dataset of 1,793 samples created by the authors.

The study compared an LSTM-based RNN model with three different Multilayer Perceptron (MLP) architectures — ReLU, Tanh, and the proposed MLP. Windows API calls were converted into embedding vectors, and the dataset was split into 80% training and 20

According to the experimental results, the lowest accuracy and recall values were obtained from the RNN model, calculated as 55.408% and 40.77%, respectively. In contrast, the highest accuracy and recall values were achieved by the proposed MLP model, with 89.309% and 84.549%, respectively. Additionally, the proposed model reached an accuracy of 93.33% in the final evaluation stage.

The limited number of samples in the dataset restricted the model's generalizability. Moreover, k-fold cross-validation was not applied, and it was observed that due to high computational demands, the model exhibited default behaviors as the number of epochs increased.

Future work aims to incorporate Transformer- and LSTM-based hybrid models, retrain the models on larger and more balanced datasets, and integrate them into real-time sandbox environments.

Taheri and colleagues (2024) [10] proposed a deep learning–based defense mechanism called DP-NC (Differential Privacy–Noise Cancellation), grounded in differential privacy (DP) principles, for the detection of Android-based malicious software (malware). The study was conducted on the Drebin (115,080 samples), Genome (1,200 samples), and Contagio (13,670 samples) datasets, focusing on binary benign–malicious classification.

Two types of attack scenarios were addressed in the research: DP-NI (Data Poisoning with Noise Injection): an attack type that injects noise into the data in accordance with differential privacy principles, causing the model to learn incorrect samples; and GDP (Gradient Differential Privacy): an attack type that manipulates the model's gradient information, resulting in privacy breaches and classification errors. The DP-NC defense mechanism developed against these attacks aims to protect DNN-based classification models from both DP-NI and GDP threats.

According to the experimental results: on the Drebin dataset, accuracy reached 94.63% in the absence of attacks, and 27.27% under the GDP attack. On the Contagio dataset, the highest accuracy was 94.21% without attacks, while accuracy dropped to 34.89% under attack. On the Genome dataset, accuracy reached 94.63% without attacks and 44.8% under attack. The main limitations of the study include high computational cost and long training time. Furthermore, the model demonstrated limited generalizability in complex attack scenarios. Future work aims to perform hyperparameter tuning and architectural optimization of the DP-NC defense mechanism, enhance its applicability for real-time Android malware detection, and integrate differential privacy with Transformer- and attention-based models.

Ghahramani and colleagues (2024) [11] conducted a study using the UCI Malware dataset to classify Windows-based malicious software (malware). The dataset contains more than 100,000 malware samples and includes 486 static features (API, File System, Directory, Misc.).

The study presented a comparative analysis among Clustering, Similarity-based, Probabilistic, Lattice-based, and Deep Learning (Deep Image) approaches. According to the obtained results, the deep learning method outperformed the other approaches (which achieved 3.63–7.31% accuracy) by reaching 45.1% accuracy and an error rate of 11.1%. The probabilistic approach ranked second in terms of the accuracy–performance balance.

In contrast, the clustering-based method showed a high error rate (41.59%) and high computational cost. Moreover, the low accuracy of the model was adversely affected by the limited diversity of the dataset. Future work aims to test the model on larger datasets, optimize the Deep Image architecture, and develop lightweight deep learning models for real-time detection.

Çatak and colleagues (2020) [13] conducted a study in which both binary and multi-class classification of Windows-based malware was performed using an LSTM-based deep learning model built on API call sequences. The dataset used in the study is the Windows PE API Sequence dataset, which contains a total of 7,107 malware samples. This dataset covers eight different malware classes: Adware, Backdoor, Downloader, Dropper, Spyware, Trojan, Virus, and Worm.

The dataset was split into 80% training and 20% testing. Experimental analyses were carried out in a Python 3.6.5 environment, with LSTM used as the main model. Additionally, KNN, DT, RF, RBF-SVM, and Sigmoid-SVM models were applied for comparison.

In binary classification, accuracy values ranged between 83.5% and 98.5%, while in multi-class classification the average F1-score was obtained in the range of 39%–47%. The LSTM model demonstrated one of the most stable results in multi-class classification with a precision of 50%. Among the other models, KNN achieved 35%, RF 46%, SVM 78%, and DT 40% precision.

This study stands out in the literature as a deep learning–based approach performing both binary and multi-class classification on Windows PE–based datasets. However, limitations in the labeling accuracy of the VirusTotal service led to misclassifications in some malware types, thereby restricting the model's generalizability. Future work suggests applying LSTM models to more complex malware types such as metamorphic malware, developing new methods against malware that detect sandbox environments, and evaluating additional sequence-based classifiers (e.g., RNN, GRU) beyond TF-IDF.

Aryal and colleagues (2025) [9] conducted a study in which binary malware classification was performed using a dataset consisting of 6,000 malware and 1,000 benign Windows PE samples. In the study, CNN-based deep learning models MalConv and MalConv2 were used for malware detection. Different data-splitting ratios were tested during training, and with the 70:20:10 split configuration, the MalConv model achieved an accuracy of 92.77%, while MalConv2 reached 94.58%.

TABLE II: Deep Learning–Based Literature Review Results

Authors (Year)	Sample Size	Language (if specified)	Platform	Method	Results	Limitations	Future Work
Gupta et al. (2025) [8]	Primary dataset: 2,570 samples; Secondary dataset: 1,793 samples	Python	Windows	RNN, three different MLP architectures (ReLU, Tanh, MLP)	RNN: 55.408%, MLP: 89.309%	Limited generalization; the model could not fully capture the sequential effects of function calls	Use of larger datasets, integration in sandbox environments, and application of Transformer- and LSTM-based hybrid models are recommended.
Aryal et al. (2025) [9]	Windows PE files, 7,000 samples (malware: 6,000, benign: 1,000)	Python	Windows	MalConv, MalConv2	MalConv: 92.77%, MalConv2: 94.58%	No temporal analysis performed; limited model comparisons	Testing against real-world black-box detectors to assess generalizability is recommended.
Taheri et al. (2024) [10]	Drebin: 115,080 samples; Genome: 1,200 samples; Contagio: 13,670 samples	Python 3.10.12 (Google Colab)	Android	Four-layer DNN model	DNN Accuracy: 27.27%, Recall: 8.66% (Drebin dataset)	High computational cost and long training time; limited generalizability	Architectural optimizations and real-time Android malware detection are recommended.
Ghahramani et al. (2024) [11]	UCI Malware (>100,000 samples)	-	Windows	Deep Image–DNN	Deep Image–DNN: 45.1%	High computational cost; limited model accuracy	Optimization of the Deep Image architecture and testing on larger and more diverse datasets are planned.
Divakarla et al. (2022) [12]	EMBER dataset (75% training, 25% testing)	-	Windows	DNN	DNN: 96.76%	Does not include temporal or real-time attack scenarios	Investigating the applicability of the GAN-based approach in real-world malware detection systems is recommended.
Çatak et al. (2020) [13]	Windows PE API call sequences (7,107 samples)	Python 3.6.5	Windows	LSTM and additionally KNN, DT, RF, SVM	LSTM Accuracy: 98.5%, F1: 47%	VirusTotal labeling is not fully accurate; generalizability is limited	Using image-based classification techniques and applying LSTM models to more complex malware types is planned.

Note: The table summarizes the datasets, model architectures, performance metrics, and limitations of deep learning–based malware detection studies.

With the alternative 80:10:10 split configuration, these accuracy values were reported as 94% for MalConv and 98% for MalConv2. Additionally, the study introduced a new adversarial attack method based on intra-section code cave injection and analyzed its impact on deep learning models. As a result of these attacks, the models' evasion rates—indicating their success in avoiding detection—were measured as 92.31% for MalConv and 97.93% for MalConv2.

These findings demonstrate that the models possess significant vulnerabilities against adversarial examples in binary classification tasks. From a scientific standpoint, the study presents an important achievement, as the proposed intra-section adversarial evasion method provides a substantially higher evasion success than existing techniques in the literature. However, from a cybersecurity perspective, the results reveal a concerning scenario: such high evasion rates indicate that deep learning–based malware detectors may not be sufficiently reliable in practical use and may remain vulnerable to adversarial attacks. The study did not include temporal (time-based) analysis, and model comparisons were limited; the effects of different architectures and temporal data splits were not examined. Future work aims to test the proposed method against real-world black-box detectors to improve generalizability; conduct temporal experiments; compare different model architectures; and evaluate runtime/dynamic scenarios.

Divakarla and colleagues (2022) [12] conducted a study in which binary malware classification was performed on the JSON-

formatted EMBER dataset using static analysis. The dataset consists of 200,000 samples, with 100,000 labeled as benign and 100,000 as malicious. The data were split into 75% training and 25% testing. In the study, a DNN-based hybrid model enhanced with a GAN (Generative Adversarial Network) approach was employed. The data were vectorized and one-hot encoded before being fed into the model. According to the experimental results, the baseline DNN model achieved an accuracy of 96.76%. In contrast, the GAN-enhanced hybrid model achieved a higher accuracy of 97.42%. The results also revealed that the model accuracy increased as the amount of data increased.

The study relies solely on static features (vectorized data) and does not include dynamic analysis, temporal evaluation, or real-time attack scenarios. Future work aims to test the model on larger datasets, integrate it with dynamic analysis, and investigate the applicability of the GAN-based approach in real-world malware detection systems.

C. *Malware Classification Studies Using Hybrid Methods*

In this section, studies conducted on malware classification based on hybrid model integration are presented, and summary information regarding these studies is provided in Table III.

$$1em]$$

Awwal and colleagues (2025) [14] conducted binary classification on Windows PE files using three different subsets (2017–2018) derived from the EMBER dataset. In the study, a hybrid deep learning model combining Autoencoder + 1D-CNN + BiLSTM architectures and enhanced with BVR-SFO-AEDL optimization was proposed. According to the experimental results, the accuracy rates were obtained in the range of 89.73%–96.47% for the first dataset, 90.00%–96.53% for the second dataset, and 90.00%–96.73% for the third dataset. Similarly, F1-score values ranged from 93.29%–97.75% for the first dataset, 93.08%–97.65% for the second dataset, and 91.97%–97.41% for the third dataset. These findings demonstrate that the proposed hybrid BVR-SFO-AE-DL model achieves high performance across all datasets.

Although the model achieves high accuracy, it carries a risk of overfitting. Furthermore, the study does not include real-time application or cross-platform testing (e.g., Android). Future work aims to integrate the model into real-time distributed systems and combine it with AI-based adaptive protection mechanisms.

Rodrigo and colleagues (2021) [17] conducted a study in which binary malware classification was performed for Android applications using the Omnidroid dataset, which includes 22,636 static features and 2,210 dynamic features. The data were split into 70% training, 15% validation, and 15% testing. According to the performance results of the proposed hybrid model, the dynamic model achieved 81.1% accuracy and 83.4% precision, the static model reached 92.9% accuracy and 91.1% precision, and the hybrid model achieved 91.1% accuracy and 91.0% precision. The hybrid model demonstrated higher overall performance compared to using only static or dynamic approaches.

The limitations of the study include the lack of testing on iOS or other mobile platforms and the restricted capability for real-time analysis. Future work aims to update the dataset, develop new feature extraction tools, generalize the model to the iOS platform, and design continuously learning systems.

par Feng and colleagues (2025) [15] conducted a study in which both binary and multi-class malware classification were performed on Android applications using the CICAndMal2017 (2,100 samples; 1,700 benign, 426 malware) and CICMalDroid2020 (1,884 samples; 981 benign, 903 malware) datasets in a Windows 10 environment. The proposed hybrid model, HGDetector, performs hybrid feature extraction by combining the Network Behavior Function Call Graph and the Network Traffic Node Interaction Graph components. Additionally, various classifiers such as LR, SVM, RF, and MLP were used. In the hybrid feature extraction stage, a static + traffic-based approach (graph embedding + network flow analysis) was applied. According to the experimental results, the accuracy rate was measured as 93.2% on the CICMalDroid2020 dataset and 96.7% on the CICAndMal2017 dataset. The use of hybrid features provided higher performance compared to individual (static-only or traffic-only) approaches.

The limitations of the study include the fact that the model was tested only in the Android environment, real-time traffic analysis remained limited, and no validation was performed on large-scale network data. Future work aims to adapt the model to other mobile platforms (e.g., iOS) and integrate it with deeper learning-based architectures.

Taher and colleagues (2023) [16] conducted a study in which both binary and multi-class malware classification were performed using the Drebin, CICAndMal2017, APKMirror, and VirusShare datasets. The dataset consists of 4,890 samples, including 1,910 malware instances, 2,980 benign applications, and 13 different malware classes. Both static analysis (Apktool) and dynamic analysis (CuckooDroid sandbox) were applied to Android applications. The proposed hybrid architecture combines static and dynamic features to create an ANN-based model. Hyperparameter optimization of the ANN was performed using the Enhanced Harris Hawks Optimization (EHHO) algorithm, and its performance was compared with classical machine learning models. According to the results, multi-class classification achieved 82.7% accuracy, while binary classification reached 96.7% (static) and 89% (dynamic).

The limitations of the study include lower detection performance for malware types employing code obfuscation and encryption, as well as testing exclusively on the Android platform. Future work aims to integrate the model with real-time detection systems and support it with temporal analysis methods.

D. *Comparative Evaluation of Methods*

Performance Comparison of Methods: When examining the results of machine learning, deep learning, and hybrid learning approaches, it is observed that machine learning and hybrid-based methods generally demonstrate higher and more consistent performance compared to deep learning methods.

In machine learning–based models, accuracy rates reported in the literature range from 54% to 98%, while hybrid models achieve 81%–96%, and deep learning–based models vary between 27% and 98%. Notably, in hybrid integrations based on CNN + BiLSTM and Autoencoder architectures, accuracy rates are significantly improved. This indicates that combining static and dynamic features provides higher and more balanced performance compared to using single feature types.

Furthermore, classical machine learning architectures such as Random Forest (RF) and K-Nearest Neighbors (KNN) consistently deliver high accuracy in many studies; however,

TABLE III: Hybrid Method Literature Review Results

Authors (Year)	Sample Size	Language (if specified)	Platform	Method	Results	Limitations	Future Work
Awwal et al. (2025) [14]	Three subsets derived from the EMBER dataset (2017–2018)	-	Windows	Hybrid model (Autoencoder + 1D-CNN + BiLSTM)	Accuracy: 96.47%	Carries a risk of overfitting and no real-time application was performed	Integration of the model into real-time distributed systems is planned.
Feng et al. (2025) [15]	CICAndMal2017: 2,100 samples; CICMal-Droid2020: 1,884 samples	Python 3.9	Android	Hybrid model: HGDetector	Accuracy for CICMal-Droid2020: 93.2%	No validation was performed on large-scale real network data	Adapting the model to other mobile platforms (e.g., iOS) and integrating with deeper learning architectures is planned.
Taher et al. (2023) [16]	Drebin, CICAnd-Mal2017, etc. (4,890 samples)	-	Android	Hybrid model: DroidDetectMW	Static binary classification accuracy: 96.9%	Single-type dynamic features (e.g., network traffic or system calls) are insufficient	Inclusion of temporal analyses is recommended.
Rodrigo et al. (2021) [17]	Omnidroid dataset (31,931 Android apps; 22,636 static + 2,210 dynamic features)	Python 3.7.6	Android	Hybrid model (Static + Dynamic + Fully Connected Hybrid Neural Network)	Hybrid model accuracy: 91.1%, Dynamic model: 81.1%	iOS or other mobile platforms were not tested; real-time analysis is limited	Generalization to iOS and design of continuously learning systems with automated labeling are planned.

Note: The table summarizes the datasets, methods, achieved performance metrics, and limitations of hybrid model–based malware detection studies.

deep learning–based models outperform them in complex and multi-layered attack scenarios.

Impact of Datasets: Examination of the literature reveals that the size, balance, and type of datasets play a critical role in the performance and accuracy of the models. For instance, models trained on large and balanced datasets such as CICMalDroid2020, CICAndMal2017, and EMBER have produced more stable results compared to models trained on smaller or imbalanced datasets (e.g., Emotet or Drebin). Additionally, static analysis methods generally achieved higher accuracy on Windows-based datasets, whereas dynamic or hybrid approaches were found to be more effective on Android-based datasets.

Since nearly all datasets exhibited some degree of class imbalance, data balancing techniques were applied in these studies, positively contributing to model performance. In particular, results obtained using feature selection and balancing techniques such as SMOTE and AEF reported significant improvements in accuracy rates.

Distribution by Application Domains: Although the majority of studies examined in the literature focus on cybersecurity and malware detection, differences can be observed among the platforms and analysis approaches used. In the Android environment, models that primarily utilize dynamic analysis or network traffic features (e.g., HGDetector) are predominant. In contrast, in the Windows environment, static analysis–based methods, relying on file contents and PE structure for feature extraction, are widely employed.

Furthermore, some studies conducted on multiple platforms (e.g., Taher et al., 2023) have evaluated both Android and Windows data within the same model, developing platform-independent hybrid detection systems. Overall, models developed for Android environments are more effective at capturing dynamic threats, whereas Windows-based models generally achieve higher accuracy rates.

Common Limitations and Gaps: Similar limitations have been identified across most studies in the literature.

First, the lack of temporal analyses is prominent. The models' ability to classify and predict future attack types based on past threats is often unexamined or not included in the studies. Second, comparisons are typically conducted on a limited number of models, resulting in evaluations that may not provide sufficient statistical diversity. Third, many studies do not employ k-fold cross-validation, limiting the generalizability of the obtained results. Fourth, the absence of real-time analysis represents a significant gap; most models have been tested solely on offline data, with no validation conducted in live systems.

Moreover, the majority of studies are conducted on a single platform (predominantly Android), which reduces cross-platform generalizability. Finally, the complexity of deep learning–based models is among the key factors that limit their practical applicability.

Support with Figures: In Figure 1, the results obtained using LR (Güngör et al.), KNN (Zada et al.), Autoencoder + 1D-CNN + BiLSTM hybrid model (Awwal et al.), DroidDetectMW hybrid

multi-class model (Taher et al.), HGDetector (CICMalDroid2020 dataset; Feng et al.), MalConv (Aryal et al.), and RNN (Gupta et al.) are presented. The figure provides a comparative view of the accuracy performance across different approaches, highlighting that hybrid models generally demonstrate the highest performance.

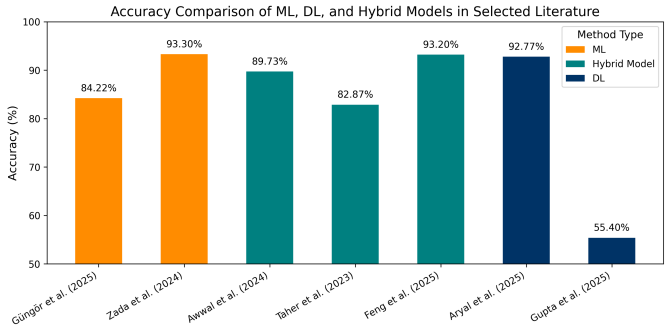

Fig. 1: Comparison of Accuracy Rates for ML, DL, and Hybrid Model Approaches in Selected Literature Studies

The graph shown in Figure 2 presents the average accuracy values of ML, DL, and hybrid models based on selected studies from the literature. As observed, hybrid models (88.6%) and machine learning–based methods (88.8%) demonstrate similarly high performance, whereas the average accuracy of deep learning–based approaches (74.1%) remains lower. These results indicate that hybrid and machine learning–based approaches generally provide more stable and higher accuracy in the literature, while deep learning methods, despite their model complexity, may exhibit lower performance on certain datasets.

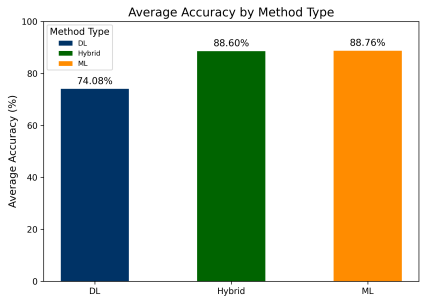

Fig. 2: Comparison of Average Accuracy Rates by Method Type

E. Overall Evaluation of the Literature and the Original Contribution of This Study

Comprehensive literature reviews and analyses indicate that significant progress has been made in recent years in malware detection. However, several common limitations are evident across existing studies.

For instance, many studies are typically conducted on a single platform (predominantly Android), neglect dataset imbalance, predominantly favor dynamic analysis approaches, and insufficiently employ cross-validation (K-Fold) methods to demonstrate model generalizability. Moreover, although deep learning–based studies achieve high accuracy, computational cost and overfitting issues have limited model stability.

In the present study, these limitations are systematically addressed through a comprehensive integrated analysis using the EMBER 2024 dataset, which has not been utilized in any previous study. EMBER 2024 is a large-scale, up-to-date dataset containing more than 3.2 million samples across six different platforms (Win32, Win64, .NET, APK, ELF, PDF). In this regard, the study relies on a dataset that is significantly broader, more current, and multi-platform compared to existing datasets and previous EMBER versions (2017/2018).

The study applies six machine learning, four deep learning, and one hybrid model for binary malware classification (benign–malware). Model performance will be evaluated in the main experiments using Stratified K-Fold cross-validation, while an additional experiment will employ a temporal training–test split to assess the models' ability to learn from up-to-date attack samples. Furthermore, in the context of time-series drift analysis, weekly accuracy rates will be calculated to evaluate the trend of performance changes over time. During analysis, modern dimensionality reduction and data balancing techniques will be applied to mitigate the effects of imbalanced data and to enhance model stability and generalizability.

As a result, this study not only achieves high accuracy but also provides a scalable, generalizable, and adaptable malware detection framework, offering a scientifically original contribution to the literature.

Based on the findings obtained from the literature review, the methods to be applied and the dataset to be used in this study are presented in the Methods section.

REFERENCES

[1] Jan Kincl, Tome Eftimov, Adam Viktorin, Roman Šenkeřík, and Tanja Pavleska. Comprehensive benchmarking of knowledge graph embeddings methods for android malware detection. *Expert Systems with Applications*, 288:127888, 2025.

[2] Danish Vasan, Junaid Akram, Mohammad Hammoudeh, and Adel F. Ahmed. An advanced ensemble framework for defending against obfuscated windows, android, and iot malware. *Applied Soft Computing*, 173:112908, 2025.

[3] Aslıhan Güngör, İbrahim Dogru, Necaattin Barışçı, and Sinan Toklu. Görüntü tabanlı özelliklerden ve makine öğrenmesi yöntemlerinden faydalanılarak kötücül yazılım tespiti. *Gazi Üniversitesi Mühendislik Mimarlık Fakültesi Dergisi*, 38(3):1781–1792, 2023.

[4] Rejwana Islam, Moinul Islam Sayed, Sajal Saha, Mohammad Jamal Hossain, and Md Abdul Masud. Android malware classification using optimum feature selection and ensemble machine learning. *Internet of Things and Cyber-Physical Systems*, 3:100–111, 2023.

[5] Romil Rawat, Sanjaya Kumar Sarangi, Yagya Nath Rimal, P. William, Snehil Dahima, Sonali Gupta, and K. Sakthidasan Sankaran. Malware threat affecting financial organization analysis using machine learning approach. *International Journal of Information Technology and Web Engineering*, 17(1), 2022.

[6] Islam Zada, Mohammed Naif Alatawi, Syed Muhammad Saqlain, Abdullah Alshahrani, Adel Alshamran, Kanwal Imran, and Hessa Alfraihi. Fine-tuning cyber security defenses: Evaluating supervised machine learning classifiers for windows malware detection. *Computers, Materials and Continua*, 80(2):2917–2939, 2024.

[7] Rabia Bakshi, Sakshi Lingwal, Kumud Chandra Bhatt, Siddhant Thapliyal, Mohammad Wazid, and Devesh Pratap

Singh. A robust machine learning-based mechanism for detection and analysis of malware attacks. *Procedia Computer Science*, 259:193–201, 2025. Sixth International Conference on Futuristic Trends in Networks and Computing Technologies (FTNCT06), held in Uttarakhand, India.

[8] Umesh Gupta, Shubham Kandpal, Hayam Alamro, Mashael M. Asiri, Meshari H. Alanazi, Ali M. Al-Sharafi, and Shaymaa Sorour. Efficient malware detection using nlp and deep learning model. *Alexandria Engineering Journal*, 124:550–564, 2025.

[9] Kshitiz Aryal, Maanak Gupta, Mahmoud Abdelsalam, and Moustafa Saleh. Intra-section code cave injection for adversarial evasion attacks on windows pe malware file. 159:104690.

[10] Rahim Taheri, Mohammad Shojafar, Farzad Arabikhan, and Alexander Gegov. Unveiling vulnerabilities in deep learning-based malware detection: Differential privacy driven adversarial attacks. *Computers Security*, 146:104035, 2024.

[11] Meysam Ghahramani, Rahim Taheri, Mohammad Shojafar, Reza Javidan, and Shaohua Wan. Deep image: A precious image based deep learning method for online malware detection in iot environment. *Internet of Things*, 27:101300, 2024.

[12] Usha Divakarla, K. Hemant Kumar Reddy, and K. Chandrasekaran. A novel approach towards windows malware detection system using deep neural networks. 215:148–157. 4th International Conference on Innovative Data Communication Technology and Application.

[13] Ferhat Ozgur Catak, Ahmet Faruk Yazı, Ogerta Elezaj, and Javed Ahmed. Deep learning-based sequential model for malware analysis using windows exe api calls. *PeerJ Computer Science*, 6:e285, 2020.

[14] Payal Awwal and Smita Naval. Development of heuristic adapted serial-based deep learning for efficient adversarial malware detection framework in windows. 326:114032.

[15] Jiayin Feng, Limin Shen, Zhen Chen, Yu Lei, and Hui Li. Hgdetector: A hybrid android malware detection method using network traffic and function call graph. 114:30–45.

[16] Fatma Taher, Omar AlFandi, Mousa Al-kfairy, Hussam Al Hamadi, and Saed Alrabaee. Droiddetectmw: A hybrid intelligent model for android malware detection. 13(13).

[17] Corentin Rodrigo, Samuel Pierre, Ronald Beaubrun, and Franjieh El Khoury. Brainshield: A hybrid machine learning-based malware detection model for android devices. 10(23).