A Unified Multi-Model Learning Framework for Reliable Static Malware Detection

Muhammad Adeel Abid
Institute of Computer Science (Khwaja Fareed University of Engineering and Information Technology, Rahim Yar Khan).
*Correspondence: adeel.abid@kfueit.edu.pk

Citation | Abid. M. A, "A Unified Multi-Model Learning Framework for Reliable Static Malware Detection", IJIST, Special Issue pp 87-99, April 2026
Received | March 11, 2026 **Revised |** April 22, 2026 **Accepted |** April 26, 2026 **Published |** April 28, 2026.

Malware has emerged as a key threat to computer systems and networks, and it is essential to achieve accurate and reliable malware detection. In this article, a unified multi-model learning framework is introduced that integrates deep learning and classical machine learning techniques to provide a comprehensive approach to detecting static malware. Experiments were conducted on a high-dimensional dataset (799912 rows, 2382 columns, 50000 rows) of static malware features using multiple models that include deep neural network models such as MLP, MalConv-X, CNN Hybrid, and classical models such as Logistic Regression, Random Forest, and LightGBM. Each model is trained and evaluated using evaluation metrics such as accuracy, precision, recall, f1-score, and AUC to ensure fair comparison and assessment. The results show that Light GBM achieved the highest performance with an accuracy of 95.48% and an AUC of 0.9915. Thus, LightGBM achieved the highest discriminative performance between malware and benign files. Deep learning models such as MLP and MalConv-X also performed well, showing 0.92 f1-score after training over 10 epochs. The CNN-hybrid model showed the highest precision value of 0.9459 but a comparatively lower recall value of 0.8721. Correlation metrics, radar charts, and epoch-wise results indicate that ensemble learning models achieve strong performance in multiple evaluation parameters, and on the other hand, deep learning models exhibit stable convergence behavior during training. The proposed unified multi-model framework shows a reliable performance for static malware detection and provides a practical approach for model selection in real-world cybersecurity applications.

Keywords: Static Malware Detection, Multi-Model Learning, Cybersecurity, Machine Learning Classification, Ensemble and Deep Learning Models

Introduction:

Malware is becoming increasingly complex, making accurate detection essential for protecting modern computing systems [1][2][3]. Traditional signature-based systems often face difficulties in detecting new and evolving threats, which motivates researchers to explore automated techniques that learn from file features. Recent work has explored both handcrafted feature engineering and end-to-end learning approaches to improve detection performance and reduce false alarms. Static analysis is still widely used because it avoids sandboxing and supports large-scale data efficiency. However, it faces challenges such as feature representation and changes in malware behavior over time [4]. Ensemble methods and gradient boosting have shown strong results on high-dimensional feature vectors and remain a practical baseline for many applied systems. At the same time, deep models show that patterns can be learned automatically without heavy manual feature engineering [5].

Recent studies on static malware comprise a mix of deep learning, classical learning, and interpretability techniques that are used to better understand the detection of malicious behavior. [6] Used LIME to interpret Conv-LSTM auto encoder predictions in industrial IoT traffic and illustrated that local explanations can highlight influential features. But their approach is limited to IoT data and lacks evaluation on high dimensions. [7] Applied LIME to Android opcode sequences converted into images for CNN classifiers, but did not generalize to search for malware. [8] Used it alongside BERT-based models to interpret Android app features. [9] Developed TabLSTMNet by combining TabNet with LSTM and integrating LIME to provide clearer explanations of deep learning outputs, but the evaluation focused only on Android apps, and classical classifiers were ignored. [10] Introduced hierarchical LIME to generate more structured and readable explanations at different code levels, but scalability to large datasets was not achieved. [11][9] integrated SHAP with hybrid CNN-BiGRU architectures to explain model predictions, but they did not provide a standard comparison of different models. Higher-level interpretability methods have been introduced in recent research. [12] Proposed the CFG Explainer, which utilizes control-flow graphs to provide details about malware behavior. [13] Extended this idea through the GAGE framework by applying genetic algorithms to identify malicious functions. However, these approaches were computationally complex and domain-specific. Recent research has explored transformer-based models. [14] Demonstrated the effectiveness of transformers in capturing sequential patterns in API call data. Similarly, [15] and other encoder-decoder-based approaches focused on improving detection performance rather than explanation. The limitation in this research is that the authors only focused on deep learning models and ignored classical models. Most of the studies focus on single models or on re-implementations under different settings, and do not focus on various approaches under a single standardized pipeline. There is a need for unified multi-model evaluation that applies the same preprocessing, training, and metrics to classical and deep learning models so that their relative strengths and weaknesses for real-world malware detection can be fairly assessed.

The pipeline begins with data preprocessing, followed by feature standardization is applied to the features. Six different models, three machine learning and three deep learning models, are applied for training purposes. The performance of these models is evaluated using evaluation metrics such as accuracy, precision, "recall", "f1-score", "AUC", and visual graphs & charts are used to understand the comparison and practical approach [16].

This research paper is organized as section 3 presents the Materials and Methods that describe the dataset, preprocessing, and model development. Section 4 provides the Results and Discussion that includes experimental evaluation and comparative analysis. In the end, Section 5 concludes the paper with future work.

The objective and novelty of this study are given below.

Objective of the Study:

The research objectives are as follows

To design a unified pipeline for the evaluation of multiple models.

To compare deep learning and machine learning methods.

To analyze performance trade-offs using evaluation metrics.

To recognize the most reliable model for the real-world.

Novelty Statement:

Most of the existing research was based on the evaluation of single models or different combinations of heterogeneous models under different experimental settings. This research proposes a unified multi-model framework that integrates both machine learning and deep learning algorithms with a standard evaluation pipeline for static malware detection. This framework ensures consistent preprocessing, feature extraction, and evaluation, enabling fair comparison across models.

Materials and Methods:

Investigation site:

The experiments are executed in an artificial digital environment that is practically possible for the evaluation of malware detection algorithms. The experiments are carried out using the high-dimensional static malware dataset, which includes samples of both benign and malicious Windows Portable Executables [17]. The dataset consists of a large variety of different file formats and threat types that make it appropriate for training and testing purposes. It covers the realistic behavior of malicious software and offers a variety that allows us to carefully test the classifier [18]. All the experiments are executed using the same workstation with the same hardware configuration to eliminate performance differences. Different important factors, such as processor loading, memory usage, and temperature, are constantly measured. In this way, the controlled environment provides a feasible platform for model comparison and allows us to validate the hypotheses about a single learning pipeline that can point out the pros and cons of classifiers [19].

Materials and methods:

Data used for this study is taken from publicly accessible and widely recognized Kaggle malware repositories to ensure reliability and reproducibility [20]. The dataset contains 799912 rows with 2382 columns. Memory-efficient sampling is done, and 50000 rows are selected for experimental purposes. The neural network models (MLP, MalConv-X, and CNN-Hybrid) are trained utilizing the AdamW optimizer (learning rate = 0.0001). A batch size of 32 was used for training, and each model was trained for 10 epochs. Machine learning models such as Logistic Regression are trained with a maximum of 500 iterations, Random Forest is configured with 200 trees and a maximum depth of 20, and LightGBM is trained with 300 estimators with a learning rate of 0.05 and 64 leaves. Cross-entropy loss is implemented as well. Classical machine learning models, such as Logistic Regression [21] are trained with a maximum of 500 iterations. Random Forest [22] is configured with 200 trees and a maximum depth of 20, whereas LightGBM is trained with 300 estimators and a learning rate of 0.05 and 64 leaves. All models are trained and evaluated under the same data splits to ensure consistency. The unified multimodal framework algorithm is as follows.

Input = Dataset D

Output: Performance metrics for all models

Preprocess D

Extract features F

Split the dataset into train/test

For each model M:

Train M on the training set

Evaluate M on the test set

Store metrics
Compare all models

The Unified framework ensures that the same dataset split, the same feature representation, the same evaluation metrics, and a controlled experimental setting. In developing this methodology, an effort has been made to ensure ease of reproducing the work for other researchers with basic machine learning experience, while also embedding model improvements that give value to the research area. Figure 1 shows the adopted methodology to accomplish this research. The EMBER 2018 dataset is taken and then preprocessed by extracting static features like header information, metadata, imported functions, and byte-level characteristics. Deep learning models [23] named MLP, MalConv-X, and CNN-Hybrid, and classical algorithms [24] named Logistic Regression, Random Forest, and LightGBM are applied. Each model is trained and tested with the same train-test split and identical feature inputs. Models are compared based on Accuracy, Precision, Recall, F1-score, and AUC.

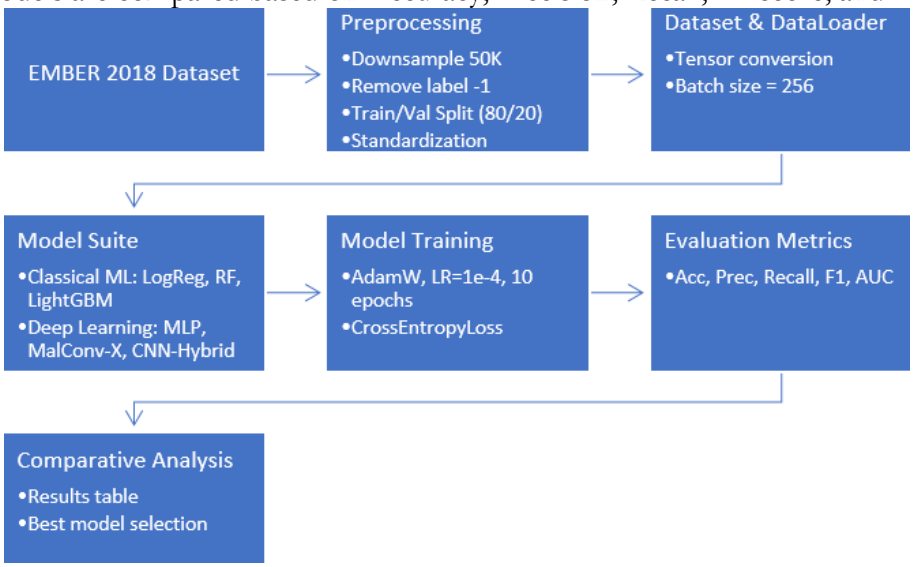

Figure 1. Adopted Methodology.

Results and Discussion:

The performance of six classification models, namely MLP, MalConv-X, CNN-Hybrid, Logistic Regression, Random Forest, and LightGBM, is evaluated across evaluation metrics such as Accuracy, Precision, Recall, F1-score, and AUC [23]. Tables 4.1 to 4.3 present epoch-wise performance of the deep learning models, showing metric evolution during training. Figures 2-8 present different graphs that compare the models in terms of overall performance, metric distribution, and tradeoffs among key measures [25]. These results show a comprehensive understanding of each model's strengths, weaknesses, and reliability in classifying the dataset. The results from the final epoch are used for all graphical analyses to ensure stable performance comparison.

Table 1 summarizes the epoch-wise performance of the MLP model [24]. The results show consistent and gradual improvement among all metrics as training progresses. The model begins with an accuracy of 0.8807 in the first epoch and gradually moves to 0.9306 in the 10th epoch, which indicates stable learning without sudden jumps or drops. Precision and recall follow the same pattern of accuracy and reach 0.9318 and 0.9291 in the 10th epoch. This suggests that the model becomes increasingly balanced in identifying both benign and malicious traffic samples. The F1-score improved from 0.8807 (epoch 1) to 0.9306 (epoch 10), which shows that the model is growing to refine its decision boundaries. The AUC also shows improvement from 0.9527 to 0.9755, which is a clear signal that the MLP has a stronger capability to separate the two classes even in borderline cases.

Table 1. Epoch-wise evaluation metrics of the MLP model

Epoch	Accuracy	Precision	Recall	F1-score	AUC
1	0.8807	0.8814	0.8798	0.8807	0.9527
2	0.9058	0.8968	0.9171	0.9058	0.9672
3	0.9128	0.9089	0.9176	0.9128	0.9715
4	0.9182	0.9162	0.9206	0.9182	0.9731
5	0.9218	0.9285	0.9139	0.9218	0.9741
6	0.9242	0.9174	0.9323	0.9242	0.9747
7	0.9248	0.9283	0.9208	0.9248	0.9737
8	0.9271	0.9252	0.9294	0.9271	0.9751
9	0.9266	0.9355	0.9163	0.9266	0.9747
10	0.9306	0.9318	0.9291	0.9306	0.9755

Table 2 presents the epoch-wise evaluation metrics of the MalConv-X model, which begins with strong initial performance and maintains stable results throughout training [26]. The model shows consistent progress, as in Epoch 1st, accuracy starts at 0.8992 and reaches 0.925 in the final epoch. Precision is high in all epochs and touches a maximum of 0.9365. Recall varies due to the MalConv-X model's sensitivity to fine-grained byte-level structures. The F1-score improves steadily and closely follows the accuracy curve. The AUC remains consistently high, ranging from 0.9658 to 0.9751, which confirms that the model is reliable in discriminating between malicious and benign files.

Table 2. Epoch-wise evaluation metrics of the MalConv-X model

Epoch	Accuracy	Precision	Recall	F1-score	AUC
1	0.8992	0.9177	0.8771	0.8992	0.9658
2	0.9038	0.9287	0.8747	0.9037	0.9704
3	0.9135	0.9273	0.8974	0.9135	0.9726
4	0.9182	0.9306	0.9038	0.9181	0.974
5	0.9123	0.9319	0.8897	0.9123	0.9721
6	0.9187	0.9064	0.9339	0.9187	0.9744
7	0.9215	0.9227	0.92	0.9215	0.9745
8	0.9192	0.924	0.9136	0.9192	0.9733
9	0.9091	0.8843	0.9414	0.909	0.973
10	0.925	0.9365	0.9118	0.925	0.9751

Table 3 shows the epoch-wise performance of the CNN-Hybrid model, which starts lower than the other two models but improves consistently over the training epochs [27]. Accuracy increases from 0.8625 to 0.9127 by epoch 8, which reflects a gradual improvement. Precision and recall show an inverse trend, where an increase in one is often accompanied by a decrease in the other, but the model maintains a stable F1-score near 0.91 in the final epoch. The AUC shows a consistent rise from 0.9452 to 0.9726, which shows the model's ability to distinguish between classes improves over successive training epochs. Although CNN-Hybrid exhibits slightly higher variability than MLP and MalConv-X, it achieves strong final-epoch performance across all evaluation metrics.

A bootstrap confidence interval analysis is conducted on epoch-wise accuracy values to assess the statistical reliability of the results. The results show that LightGBM consistently performed best among all evaluated modelsfv, with confidence intervals of pairwise differences not including zero, which indicates a statistically significant improvement. MLP and MalConv-X show similar performance, and there is no clear statistical difference between them. Both models perform better than CNN-Hybrid, and this difference is statistically reliable, meaning it is not caused by random variation in results.

Table 3. Epoch-wise evaluation metrics of the CNN-Hybrid model

Epoch	Accuracy	Precision	Recall	F1-score	AUC
1	0.8625	0.8383	0.8982	0.8623	0.9452
2	0.8782	0.8361	0.9408	0.8777	0.959
3	0.8968	0.9046	0.8873	0.8968	0.9644
4	0.897	0.8697	0.9339	0.8968	0.9664
5	0.9052	0.9194	0.8883	0.9052	0.9686
6	0.906	0.9238	0.8851	0.906	0.9703
7	0.9071	0.9186	0.8934	0.9071	0.9716
8	0.9127	0.9074	0.9192	0.9127	0.9716
9	0.9126	0.9074	0.919	0.9126	0.9726
10	0.9111	0.9459	0.8721	0.911	0.9722

Figure 2 presents a heatmap visualization of the performance metrics obtained from the six evaluated classification models, named as MLP, MalConv-X, CNN-Hybrid, Logistic Regression, Random Forest, and LightGBM. The heatmap provides a clear comparative overview of five key evaluation parameters named Accuracy, Precision, Recall, F1-score, and AUC. The darker shades in the heatmap represent higher metric values. LightGBM shows the best performance across all metrics and achieved the highest accuracy (0.9548), precision (0.9535), recall (0.9563), F1-score (0.9548), and AUC (0.9915). This consistent performance of LightGBM indicates that LightGBM is the most reliable model for the given classification task. Among the deep learning models, MLP and MalConv-X perform well with an accuracy value of 0.9305 and 0.9249, respectively. Both models also show balanced precision and recall values. The CNN-Hybrid model shows moderate performance by showing a lower recall value (0.8720), indicating a higher misclassification rate for the positive class. Logistic Regression has the lowest accuracy (0.8928) and F1-score (0.8928), which shows the limitations of traditional linear classifiers for complex malware detection patterns. Random Forest performs comparatively well and achieves a high AUC score of 0.9833.

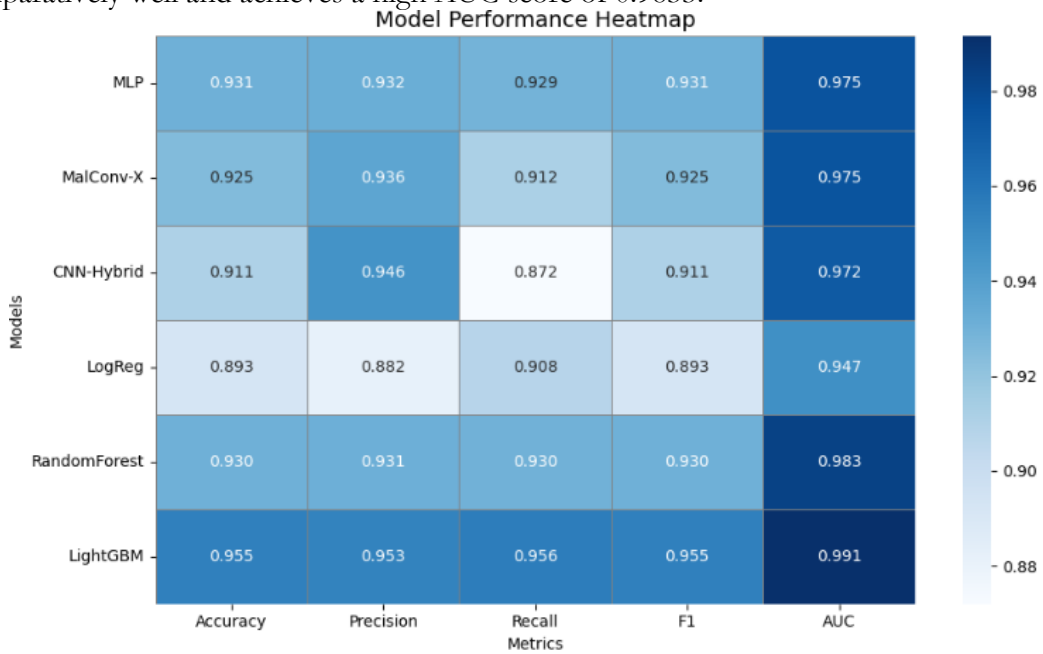

Figure 2. Model performance evaluation heatmap.

Figure 3 shows the accuracy of six classification models: MLP, MalConv-X, CNN-Hybrid, Logistic Regression, Random Forest, and LightGBM. The bar plot provides a clear visual comparison and highlights differences in predictive performance across models.

LightGBM achieves the highest accuracy of 0.9548 and shows a strong ability to capture complex patterns in the dataset. MLP and Random Forest also perform well, with accuracies of 0.9306 and 0.9304, respectively, and serve as strong alternative classifiers. MalConv-X and CNN-Hybrid have lower accuracy values of 0.9249 and 0.9110, while Logistic Regression has the lowest accuracy of 0.8928, indicating that linear models struggle with the nonlinear characteristics of the data.

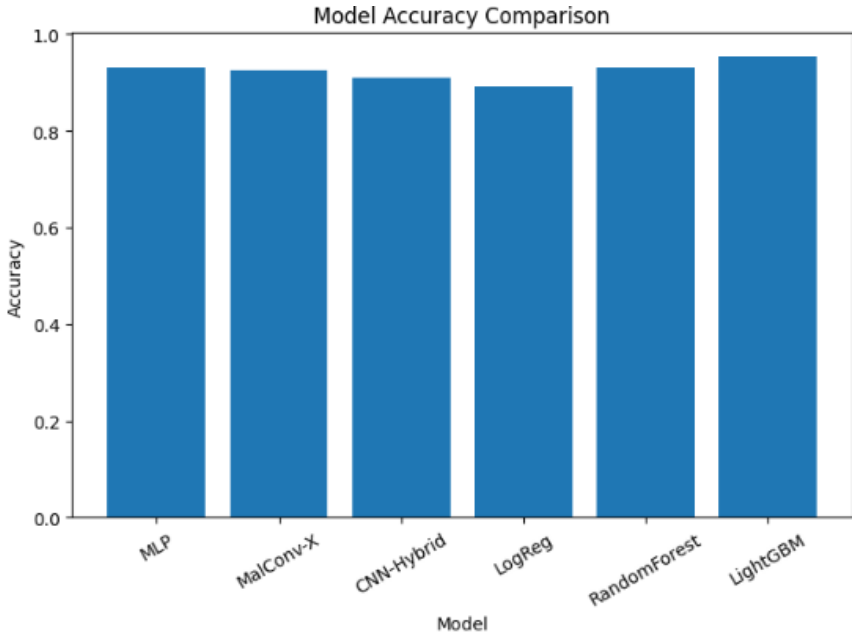

Figure 3. Comparison of model accuracy.

Figure 4 presents the AUC scores of six classification models using a horizontal bar chart. The AUC provides a measure of the model's ability to distinguish between positive and negative classes. Among the evaluated models, LightGBM achieves the highest AUC of 0.9915, indicating excellent discrimination capability. Random Forest also shows strong performance with an AUC of 0.9834, followed by MLP and MalConv-X with scores of 0.9755 and 0.9751, respectively. CNN-Hybrid shows a slightly lower AUC of 0.9722, while Logistic Regression has the lowest AUC of 0.9473, indicating weaker class discrimination.

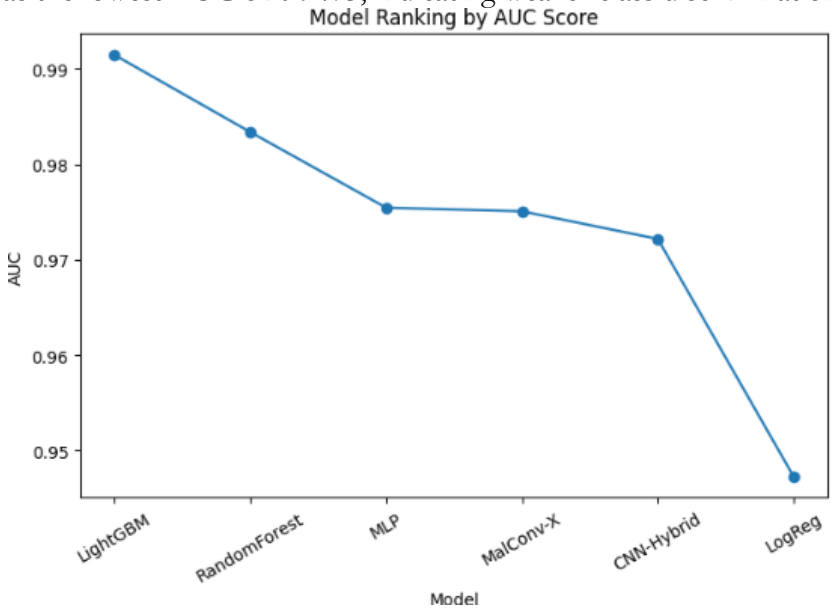

Figure 4. Comparison of model AUC scores.

Figure 5 presents a radar chart illustrating the performance of six classification models evaluated with respect to five metrics: Accuracy, Precision [28], Recall, F1-score, and AUC. Each axis in the chart represents one metric, and each model corresponds to a polygon connecting its metric values, providing a comprehensive visual overview of strengths and weaknesses. LightGBM [29] shows strong performance in all metrics and has the most balanced results. Indicating strong classification performance across precision and recall. MLP [30] and Random Forest also perform well, but with only minor variations across metrics. Logistic regression has lower performance, especially in accuracy and F1-score, which shows it is less effective than more complex models.

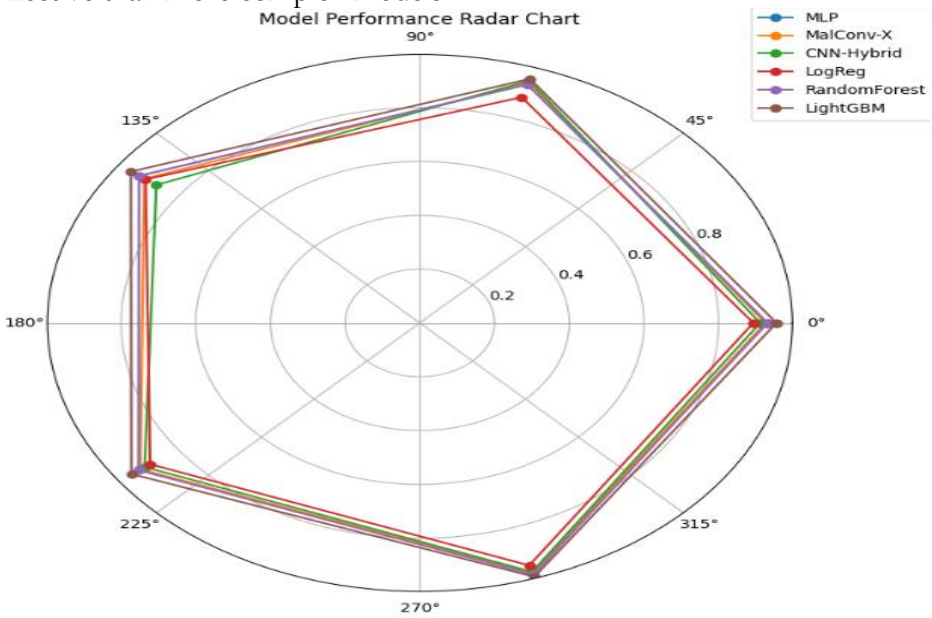

Figure 5. Model performance radar chart.

Figure 6 presents a scatter plot comparing the Precision and Recall scores of all six classification models. Each point represents a model and is labeled for clarity. The plot shows the tradeoff between precision and recall for different models. LightGBM appears in the top right corner, which means that it has both high precision and high recall. This shows that it can correctly identify positive cases with very few false positives. MLP and Random Forest also perform in a balanced manner, cluster in the top-right region. On the other hand, CNN-Hybrid and Logistic Regression show slightly lower recall values, showing a greater likelihood of misclassifying positive samples, even while their precision scores remain acceptable.

Figure 7 presents a scatter plot illustrating the relationship between Accuracy and AUC for the six classification models. All the points correspond to one model each and have been labelled for better readability. From the plot, a clear positive correlation is observed in which models with higher Accuracy generally achieve a higher AUC as well. LightGBM has the highest Accuracy, at 0.9548, and the highest AUC, at 0.9915. It shows great all-around classification performance with very good discriminative capability. MLP and Random Forest also show balanced performance with high values on both axes. In contrast, Logistic Regression has a much lower Accuracy of 0.8928 and an AUC of 0.9473, hence revealing its relatively weak capability in correctly classifying instances and distinguishing between positive and negative samples.

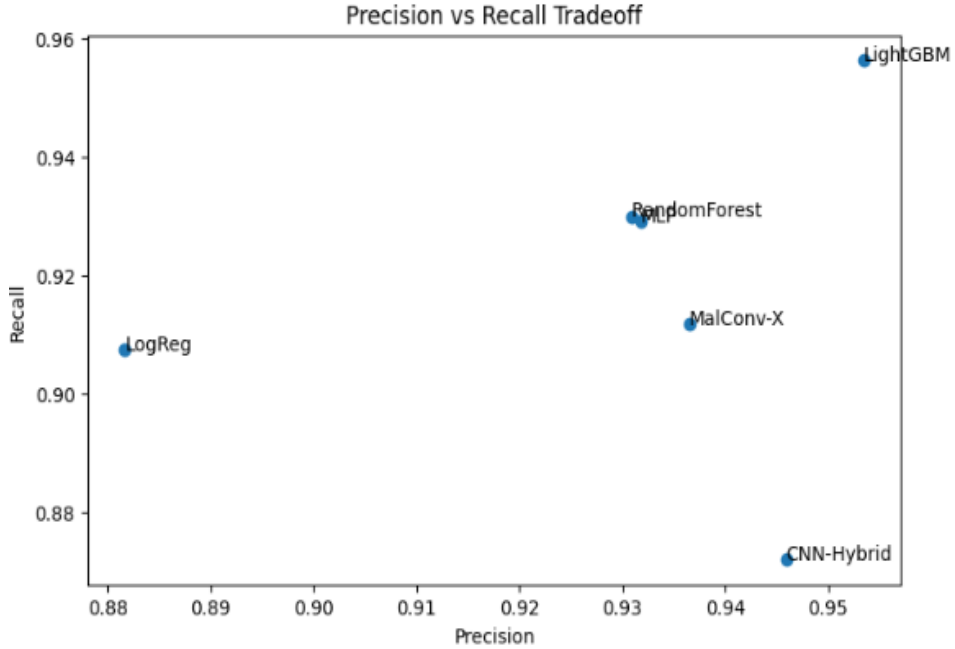

Figure 6. Precision and recall relationship.

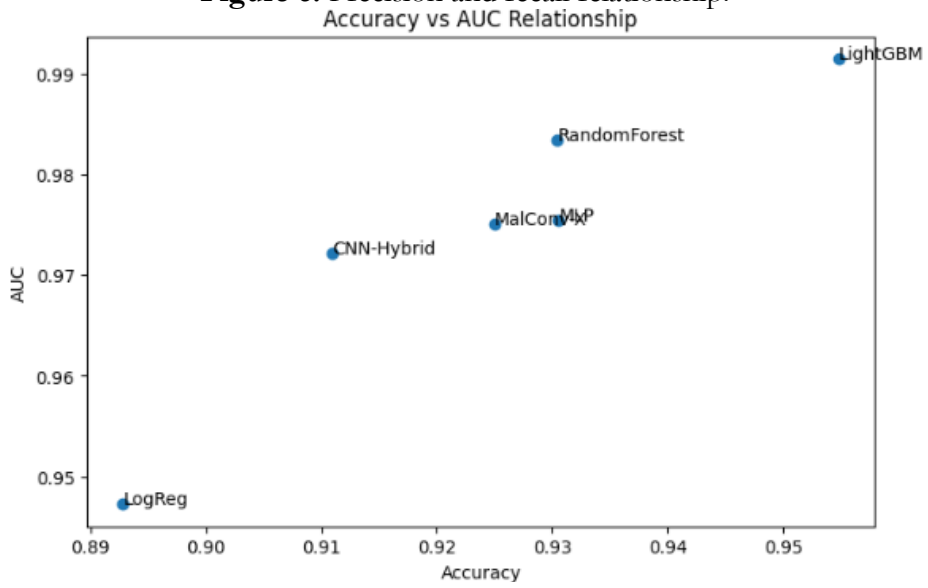

Figure 7. Accuracy vs AUC relationship.

Figure 8 shows the distribution of five evaluation metrics: Accuracy, Precision, Recall, F1-score, and AUC, for six classification models. The plot illustrates variability and central tendency across models for each metric: Accuracy, Precision, and F1-score have relatively compact distributions; Recall shows slightly greater variability, indicating differences across models in predicting positive samples; and the AUC values remain consistently high across all models, indicating strong discriminative capability. This box plot helps us judge the stability of the models under different metrics and their reliability.

Among all, LightGBM is the best-performing model with the highest value on Accuracy, Precision, Recall, F1-score, and AUC, which indicates very strong predictive performance and dependability in distinguishing between positive and negative classes. MLP and Random Forest also work very well and have quite balanced metrics. They could serve as very reliable alternatives in cases where deep learning and ensemble methods are not plausible. CNN-Hybrid has been performing moderately with some variance, especially in Recall, which means difficulty in consistently detecting positive instances. Logistic Regression has the overall

lowest metrics, which really pinpoint the limitations of linear models on complex patterns within the dataset.

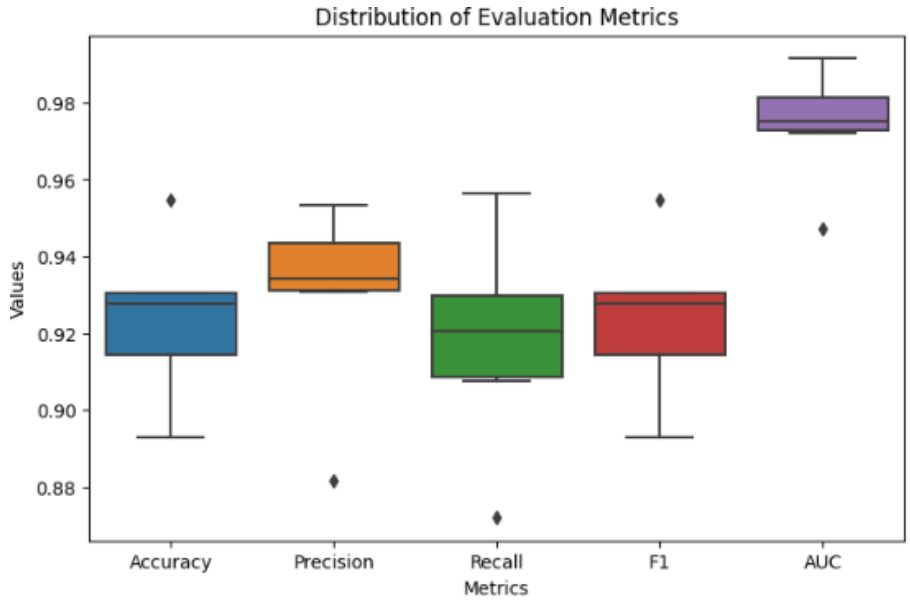

Figure 8. Distribution of evaluation metrics.

The combination of epoch-wise tables and various visualizations incorporating heatmaps, radar charts, scatter plots, and box plots offers great insights into the behavioral aspects of models. Such visualizations give ample scope for comparing different models, understanding tradeoffs between metrics like Precision and Recall, and gaining insight into stability regarding all measures. These results suggest that model selection should consider not just overall performance but also the balance concerning all metrics and reliability, especially in cases where either false positives or false negatives are critical. This analysis gives a very clear framework for the evaluation and comparison of classification models and supports decisions for practical deployment and future work.

A critical comparison with existing research is that the previous research focused on single models, domain-specific datasets, and a lack of standardized evaluation, but this work provides a unified multi-model pipeline that contains classical models, such as LightGBM, which performed well among many deep learning approaches under a consistent environment. The results are mapped with the objectives. The unified pipeline is validated through evaluation across all classical and deep learning models. The comparison between deep learning and machine learning shows that classical models named LightGBM and Random Forest performed better than others. Performance trade-offs are analyzed using multiple metrics and visualizations that highlight the differences in precision, recall, and AUC. After performing experiments, LightGBM is identified as the most suitable model for real-world deployment.

The proposed framework can be integrated into practical cybersecurity environments such as intrusion detection systems, endpoint protection platforms, and malware analysis pipelines. A unified evaluation approach enables consistent model selection, while the strong performance and efficiency of LightGBM make it suitable for real-time deployment.

Conclusion:

In this study, a unified multi-model learning framework is presented to evaluate several machine learning and deep learning approaches for static malware detection using high-dimensional features. The framework helped to examine the different models' behavior in the same conditions. The results show that LightGBM achieved the highest accuracy of 95.48 and

an AUC of 0.9915 as compared to others. Deep learning models such as MLP and MalConv-X show stable and strong results with F1-scores above 0.92, while the CNN-hybrid model showed a strong precision value that indicates fewer false positive values. These findings show that each model has different advantages depending on whether the priority is accuracy, precision, recall, or balanced performance. The experimental results further discover that there is smooth training behavior among the neural models, and all algorithms show consistent metric values. The results support the reliability of the framework and illustrate that combining multiple models in a single structured pipeline gives comprehensive details about the asset and its limitations. This research is helpful for security officials to choose the most suitable model for real-world malware detection tasks based on their specific needs. This research will act as a strong base to develop more dependable and adaptable detection systems that can meet new cybersecurity challenges.

Acknowledgement: Acknowledgements are considered necessary.

Author's Contribution: The author is responsible for conceptualization, methodology, data collection, analysis, and manuscript preparation.

Conflict of Interest: The author declares that there are no conflicts of interest.

Project Details: Not applicable.

References:

[1] Matthew Chin, Roberto Corizzo, "Continual Semi-Supervised Malware Detection," *Mach. Learn. Knowl. Extr.*, vol. 6, no. 4, pp. 2829–2854, 2024, doi: https://doi.org/10.3390/make6040135.

[2] Tiezhu Sun, Nadia Daoudi, "Temporal-Incremental Learning for Android Malware Detection," *ACM Trans. Softw. Eng. Methodol.*, vol. 34, no. 4, pp. 1–30, 2025, [Online]. Available: https://dl.acm.org/doi/10.1145/3702990

[3] M. Gopinath, Sibi Chakkaravarthy Sethuraman, "A comprehensive survey on deep learning based malware detection techniques," *Comput. Sci. Rev.*, vol. 47, p. 100529, 2023, doi: https://doi.org/10.1016/j.cosrev.2022.100529.

[4] E. Rodríguez, M. Fukkink, S. Parkin, M. van Eeten and C. Gañán, "Difficult for Thee, But Not for Me: Measuring the Difficulty and User Experience of Remediating Persistent IoT Malware," *2022 IEEE 7th Eur. Symp. Secur. Priv. (EuroS&P), Genoa, Italy*, pp. 392–409, 2022, doi: 10.1109/EuroSP53844.2022.00032.

[5] U.-E.-H ; Tayyab, F B ; Khan, "A Survey of the Recent Trends in Deep Learning Based Malware Detection," *J. Cybersecur. Priv*, vol. 2, no. 4, pp. 800–829, 2022, [Online]. Available: https://www.mdpi.com/2624-800X/2/4/41

[6] I. A. Khan, N. Moustafa, D. Pi, K. M. Sallam, A. Y. Zomaya, and B. Li, "A New Explainable Deep Learning Framework for Cyber Threat Discovery in Industrial IoT Networks," *IEEE Internet Things J.*, vol. 9, no. 13, pp. 11604–11613, Jul. 2022, doi: 10.1109/JIOT.2021.3130156.

[7] Martin Kinkead, Stuart Millar, "Towards Explainable CNNs for Android Malware Detection," *Procedia Comput. Sci.*, vol. 184, pp. 959–965, 2021, doi: https://doi.org/10.1016/j.procs.2021.03.118.

[8] Zhi Lu, Vrizlynn L.L. Thing, "'How Does It Detect A Malicious App?' Explaining the Predictions of AI-based Android Malware Detector," *arXiv:2111.05108*, 2021, [Online]. Available: https://arxiv.org/abs/2111.05108

[9] N. G. Ambekar, N. N. Devi, S. Thokchom, and Yogita, "TabLSTMNet: enhancing android malware classification through integrated attention and explainable AI," *Microsyst. Technol. 2024 313*, vol. 31, no. 3, pp. 695–713, Mar. 2024, doi: 10.1007/s00542-024-05615-0.

[10] Jeff Mitchell, Niall McLaughlin, "Generating sparse explanations for malicious Android opcode sequences using hierarchical LIME," *Comput. Secur.*, vol. 137, p.

103637, 2024, doi: https://doi.org/10.1016/j.cose.2023.103637.

[11] S. K. Smmarwar, G. P. Gupta, and S. Kumar, "XAI-AMD-DL: An Explainable AI Approach for Android Malware Detection System Using Deep Learning," *Proc. - 2023 IEEE World Conf. Appl. Intell. Comput. AIC 2023*, pp. 423–428, 2023, doi: 10.1109/AIC57670.2023.10263974.

[12] J. D. Herath, P. P. Wakodikar, P. Yang, and G. Yan, "CFGExplainer: Explaining Graph Neural Network-Based Malware Classification from Control Flow Graphs," *Proc. - 52nd Annu. IEEE/IFIP Int. Conf. Dependable Syst. Networks, DSN 2022*, pp. 172–184, 2022, doi: 10.1109/DSN53405.2022.00028.

[13] Mohd Saqib, Samaneh Mahdavifar, "A Comprehensive Analysis of Explainable AI for Malware Hunting," *ACM Comput. Surv.*, vol. 56, no. 12, 2024, [Online]. Available: https://dl.acm.org/doi/10.1145/3677374

[14] Ferhat Demirkıran, Aykut Çayır, "An ensemble of pre-trained transformer models for imbalanced multiclass malware classification," *Comput. Secur.*, vol. 121, p. 102846, 2022, doi: https://doi.org/10.1016/j.cose.2022.102846.

[15] Farhan Ullah, Amjad Alsirhani, "Explainable Malware Detection System Using Transformers-Based Transfer Learning and Multi-Model Visual Representation," *Sensors*, vol. 22, no. 18, p. 6766, 2022, doi: https://doi.org/10.3390/s22186766.

[16] Rubab Roshan, Irfan Ali Bhacho, "Comparative Analysis of TF–IDF and Hashing Vectorizer for Fake News Detection in Sindhi: A Machine Learning and Deep Learning Approach," *Eng Proc*, vol. 46, no. 1, p. 5, 2023, doi: https://doi.org/10.3390/engproc2023046005.

[17] Syed Khurram Jah Rizvi, Warda Aslam, Muhammad Shahzad, Shahzad Saleem & Muhammad Moazam Fraz, "PROUD-MAL: static analysis-based progressive framework for deep unsupervised malware classification of windows portable executable," *Complex Intell. Syst.*, vol. 8, pp. 673–685, 2022, [Online]. Available: https://link.springer.com/article/10.1007/s40747-021-00560-1

[18] J. Jeon, B. Jeong, S. Baek, and Y. S. Jeong, "Static Multi Feature-Based Malware Detection Using Multi SPP-net in Smart IoT Environments," *IEEE Trans. Inf. Forensics Secur.*, vol. 19, pp. 2487–2500, 2024, doi: 10.1109/TIFS.2024.3350379.

[19] Faitouri A. Aboaoja, Anazida Zainal, "Malware Detection Issues, Challenges, and Future Directions: A Survey," *Appl. Sci.*, vol. 12, no. 17, p. 8482, 2022, doi: https://doi.org/10.3390/app12178482.

[20] "Ember-2018-V2-features." Accessed: Mar. 19, 2026. [Online]. Available: https://www.kaggle.com/datasets/dhoogla/ember-2018-v2-features

[21] M. A. Abid, S. Ullah, M. A. Siddique, M. F. Mushtaq, W. Aljedaani, and F. Rustam, "Spam SMS filtering based on text features and supervised machine learning techniques," *Multimed. Tools Appl. 2022 8128*, vol. 81, no. 28, pp. 39853–39871, May 2022, doi: 10.1007/S11042-022-12991-0.

[22] Muhammad Adeel Abid, Madiha Amjad, "IoT-Based Smart Biofloc Monitoring System for Fish Farming Using Machine Learning," *IEEE Access*, vol. 1, no. 24, 2024, doi: 10.1109/ACCESS.2024.3384263.

[23] G. Naidu, T. Zuva, and E. M. Sibanda, "A Review of Evaluation Metrics in Machine Learning Algorithms," *Lect. Notes Networks Syst.*, vol. 724 LNNS, pp. 15–25, 2023, doi: 10.1007/978-3-031-35314-7_2.

[24] Tuan Van Dao, Hiroshi Sato, "MLP-Mixer-Autoencoder: A Lightweight Ensemble Architecture for Malware Classification," *Information*, vol. 14, no. 3, p. 167, 2023, doi: https://doi.org/10.3390/info14030167.

[25] M. A. Abid, M. F. Mushtaq, U. Akram, B. Mughal, M. Ahmad, and M. Imran, "Recommending Domain Specific Keywords for Twitter," *Adv. Intell. Syst. Comput.*,

vol. 978 AISC, pp. 253–263, 2020, doi: 10.1007/978-3-030-36056-6_25.

[26] O. Kargarnovin, A. M. Sadeghzadeh, and R. Jalili, "Mal2GCN: a robust malware detection approach using deep graph convolutional networks with non-negative weights," *J. Comput. Virol. Hacking Tech. 2023 201*, vol. 20, no. 1, pp. 95–111, Sep. 2023, doi: 10.1007/s11416-023-00498-7.

[27] P. J. Alphine, B. P. Alapatt, and J. P. George, "Enhancing Malware Detection Through Hybrid Deep Learning Techniques," *Proc. 6th Int. Conf. Intell. Commun. Technol. Virtual Mob. Networks, ICICV 2025*, pp. 478–483, 2025, doi: 10.1109/ICICV64824.2025.11085589.

[28] Eric J. Michaud, Ziming Liu, Max Tegmark, "Precision Machine Learning," *arXiv:2210.13447*, 2022, [Online]. Available: https://arxiv.org/abs/2210.13447

[29] Carina Clemente, Gracinda R. Guerreiro, "Modelling Motor Insurance Claim Frequency and Severity Using Gradient Boosting," *Risks*, vol. 11, no. 9, p. 163, 2023, doi: https://doi.org/10.3390/risks11090163.

[30] Qihao Zhao, Fuwei Wang, Weimin Wang, Tianxin Zhang, Haodong Wu & Weijun Ning, "Research on intrusion detection model based on improved MLP algorithm," *Sci. Rep.*, 2025, [Online]. Available: https://www.nature.com/articles/s41598-025-89798-0