International Journal of Emerging Research in Engineering and Technology
Pearl Blue Research Group | Volume 7 Issue 2 PP 253-261, 2026
ISSN: 3050-922X | https://doi.org/10.63282/3050-922X.IJERET-V7I2P131

Original Article

A Robust Machine Learning Architecture for Accurate Malware Classification in Large-Scale Cyber Threat Dataset

Dinesh Rajasekharan
Senior Director of Product Management, Vellore Institute of Technology USA.

Received On: 12/04/2026 ***Revised On: 11/05/2026*** ***Accepted On: 19/05/2026*** ***Published On: 25/05/2026***

Abstract *- Malware is a significant threat to information systems and software security as the technology progresses. As malware continues to evolve, it is essential to have quick and precise detection systems to deal with any potential threats. Machine learning might be a solution, as it is challenging for standard detection technologies to keep up with the rapid evolution of malware. This work introduces the EMBER dataset as the base for a machine learning malware classification system. Data pretreatment approaches and exploratory data analysis are used in order to make the data more consistent and improve the feature representation. Random Forest (RF) and Multi-Layer Perceptron (MLP) classifiers are used to develop and evaluate classifiers based on accuracy, precision, recall, F1-score and Area Under the ROC Curve (AUC). The experimental outcomes reveal that the MLP model exhibits a higher accuracy of 97.63%, recall of 95.43%, and F1 score of 95.44% and AUC of 99.13% compared with RF model with 94.66% accuracy and 92.25% AUC. The effectiveness of the proposed framework is further substantiated through a comparative study with other ML and DL approaches in terms of prediction power and generalization ability. The proposed framework is expected to offer strong malware classification capabilities, enabling cybersecurity decision-making in the evolving threat landscape, and to offer scalable and reliable detection capabilities.*

Keywords *- Attack, Malware Detection, Internet-Of-Things (Iot), Cybersecurity, Machine Learning, Malware Classification, EMBER Dataset.*

1. Introduction

This potential attack space has become significantly larger today, with the Exponential Growth of Digital Technologies, Cloud Computing, mobile platforms, and IoT systems [1]. As a result, malware is now a significant and pervasive threat in the cyber world. Modern malware families are extremely sophisticated and can rapidly evolve to bypass conventional protection measures and are far from being basic viruses. Malicious software has infiltrated critical industries such as healthcare, energy, and finance, causing significant breaches of data, financial losses, and service disruptions[2][3]. The majority of antivirus systems that use signatures to detect malware do so because these systems match malware with known patterns [4]. Although this method works well for threats that have already been recognized, it is unable to identify novel and unidentified malware variants, sometimes known as zero-day malware [5][6]. Cybercriminals constantly alter malware via packaging, encryption, polymorphism, and metamorphism, making it easy for malicious software to avoid signature-based detection [7][8]. Therefore, given the current threat scenario, depending just on traditional detection systems is no longer enough [9].

A large portion of today's malware is able to identify sandbox environments and change its behavior to seem harmless, evading detection. Malware detection and classification systems have shown great potential in the use of AI and ML [10][11]. AI allows for the automated examination of massive information, uncovering anomalies and hidden patterns that can point to the existence of harmful actions. ML algorithms in particular have the potential to be taught to distinguish between harmless and harmful actions, which might lead to more effective and accurate malware detection systems. For previously unseen malware variations, particularly zero-day assaults, the conventional signature-based methods that depend on recognized patterns of infection are useless [12]. More sophisticated obfuscation strategies like polymorphism and encryption further diminish the chances of detection. Meanwhile, the proliferation and ubiquity of Cloud Computing, IoT devices, and digital services have dramatically increased the potential consequence of cyber events[13][14]. These limitations also motivate the research of new intelligent approaches in malware detection which are able to learn complex activities and adapt to new kinds of malware. ML algorithms for malware classification have become crucial in the fight against cyber threats due to their ability to quickly and efficiently classify malware, boost detection accuracy, and enhance cybersecurity resilience.

This paper represents the following contributions to the field of detecting malware:

- Demonstrates improved malware discrimination capability through robust learning of complex executable feature patterns.
- Enhances classification reliability by reducing misclassification between benign and malicious software samples.
- Provides comparative evidence on the effectiveness of conventional and neural network-based approaches for cybersecurity applications.

- Supports scalable and accurate malware identification, addressing challenges posed by evolving and sophisticated cyber threats.
- Contributes toward the development of intelligent cybersecurity systems with improved generalization and threat detection capability[15].

The novelty of the proposed work is in the effective learning of complex patterns from the features of executable files for improving classification performance of malware. The approach provides improved separation between good and bad instances, and enables the reliable identification of threats in cybersecurity environments. The proposed model is compared with traditional models that are less flexible and has proved to have better robustness and generalization. The study is justified because of the increase in complex attack of malware that is very risky for a digital system and network security. Hence, it is crucial to build accurate, scalable and efficient malware detection methods to boost cybersecurity infrastructure and minimize possible threats.

1.1. Structure of Paper

The framework of this document is as follows: Section II evaluates pertinent endeavors. The proposed methodology is comprehensively addressed in section III. Section IV contains the experimental results and a thorough examination of the observations. Section V serves as the paper's conclusion.

2. Literature Review

Researchers have recently proposed various malware detection and classification methods that rely on different malware analysis technologies. In this section, discuss the previous work done on each of them. R. S and C. K (2025) assesses how well SVM and Autoencoder performed on the "zero-day vulnerability" dataset for classification. With a recall of 30% and a precision of 20.83%, a f1score of 23%, and a specificity of 94%, the SVM classifier achieved an accuracy of 94.9%. In contrast, the autoencoder achieved 81.6% acc, 21% prec, 28% rec, 21.05% f1score, and 81% specificity. Results show that SVM outperforms autoencoder when it comes to dataset analysis in terms of accuracy [16].

Dickey, Hwang and Kim (2024) explores the use of a Convolutional Neural Network and various tree-based learning methods processing feature engineered from malware binaries. Overall model accuracy was found to be between 87% and 90%, and sufficient performance was obtained when collecting a certain amount of data above the threshold. It was also found that tuning some of the tree based model hyperparameters caused overfitting. Overall, the findings concluded that models that did not overfit performed comparably during both training and testing, representing consistent detection models [17].

Cassel and Majd (2024) offer a compact model that detects and efficiently categorizes safe traffic from various types of disguised malware. They used their training dataset, which was derived from the CIC-MaIMem2022dataset, and their hybrid DataAugmentation approach to construct a RF model. The experimental findings showed that the suggested model has 87.1% accuracy in detecting obfuscated malware, which is better than the state-of-the-art [18].

Sharma and Babbar (2023) some ML methods have been used to identify potential Android malware risks on the IoT. A set of Android malware samples and decent applications are used to construct an ML model using this strategy. The Android Malware dataset trains a number of ML algorithms, including NB, KNN, DT, and RF, which are used to detect malware in the IoT. The DT model has the best accuracy rate at 95%, followed by the NB model at 84%, KNN at 89%, and the RF model at 92% [19].

Bokolo, Jinad and Liu, (2023) accurately categorize malware into nine distinct families. RF, DT, SVM, KNN, SGD, LR, NB, and DL algorithms are used for the categorization. The DL model has a 96% success rate, which is higher than that of competing ML methods. The study's findings suggest that DL approaches might prove to be very beneficial in the detection and classification of malware [20].

Broome *et al.* (2022) showcase the first phase of their study that developed a very efficient SMS malware detection algorithm using supervisedML. To conduct this study, five ML algorithms were trained on a database consisting of 1,048,575 messages, each of which included 85 properties. Their final application can identify malware in SMS texts with 94.6% accuracy by combining the best features and algorithms [21]

Table 1: Comparative Analysis of Existing Malware Detection and Classification Techniques

Author	Techniques Used	Data	Results	Limitations	Future Work
R. S and C. K (2025)	SVM, Autoencoder	Zero-day vulnerability dataset	SVM achieved 94.9% accuracy; Autoencoder achieved 81.6% accuracy	Low precision and recall values	Improve detection performance for unknown threats
Dickey, Hwang and Kim (2024)	CNN, Tree-based learning models	Malware binary features dataset	Achieved 87%–90% accuracy	Hyperparameter tuning caused overfitting	Develop more generalized and scalable models
Cassel and Majd (2024)	Random Forest with hybrid augmentation	CIC-MaIMem2022 dataset	Achieved 87.1% accuracy for obfuscated malware classification	Limited classification accuracy	Enhance lightweight detection efficiency

Sharma and Babbar (2023)	NB, KNN, DT, RF	Android malware dataset	DT achieved highest accuracy of 95%	Limited to IoT Android malware	Extend detection to broader IoT environments
Bokolo, Jinad and Liu (2023)	RF, DT, SVM, KNN, SGD, LR, NB, Deep Learning	Malware family dataset	Deep learning achieved 96% accuracy	High computational complexity	Optimize deep learning for real-time detection
Broome et al. (2022)	Supervised ML algorithms	SMS dataset with 1,048,575 messages	Achieved 94.6% malware detection accuracy	Focused only on SMS malware detection	Improve detection for diverse communication threats

2.1. Research Gap

Despite the emergence of several MalwareDetection and classification techniques based on ML and DL methods, existing methods still suffer from overfitting, narrow generalization, poor precision and recall, and poor performance on complex or obfuscated malware sets. As shown in Table I, many studies focus on specific malware categories or datasets, which limits their applicability in real-world environments. Thus, there is a demand for efficient and robust malware classification framework having higher detection accuracy, good scalability and performance across various malware threats.

3. Methodology

The proposed methodology starts with the collection of the EMBER data set, followed by Exploratory Data Analysis (EDA), invalid sample removal and feature organization, and concludes with the preparation of the data for the next step of the methodology. The data is then scaled to a common standard with Standard Scaler and split into training and test sets in an 80:20 ratio. Then, a RF and a MLP classifiers are used to train the models to detect malware. At last, precision, accuracy, recall, F1-score, and AUROC are calculated and compared to measure the effectiveness of classification. Propose system pipeline shows in Fig. 1.

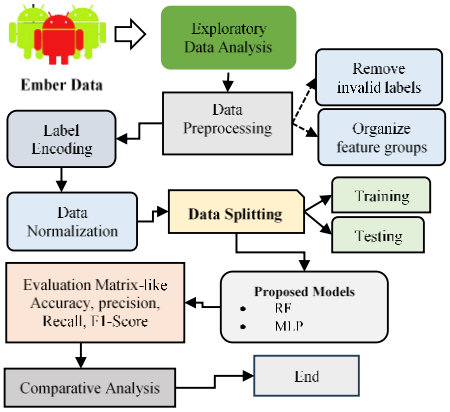

Fig 1: Proposed Flowchart for Malware Classification and Prediction

Overall discussion of propose flowchart is given below:

3.1. Data Collection

The cybersecurity field is served by Endgame Inc.'s Endgame Malware Benchmark for Research (EMBER). 1.1 million Windows Portable Executable (PE) files with benign or malicious labels are included in the dataset as feature vectors. The dataset contains 800,000 Training Samples and 200,000 Testing Samples, each labeled as either benign (0) or malicious (1). It used EDA on the EMBER dataset to look for trends in the distribution of classes, patterns in the features, and important attributes. The results of the analysis reflected that a balance of both benign and malicious samples are identified and the primary feature groups that impact the malware classification are identified. This information is useful for the correct pre-processing and modelling of the data.

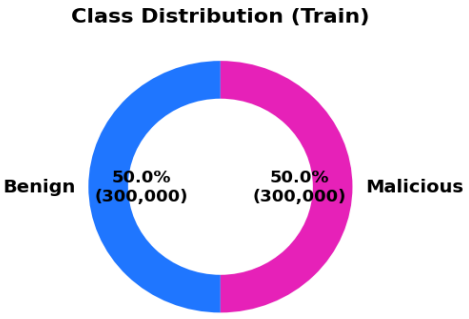

Fig 2: Donut Chart for Class Distribution

The distribution of the training samples in the two classes (Benign and Malicious) is balanced as shown in Fig. 2. There are 300,000 instances each class has, thus 50% of the data. Equal proportions ensure an unbiased training of the model and highlight that this dataset is suitable for testing the classification performance fairly.

Fig. 3 shows the average values for each of these six feature groups: Byte Hist, Entropy, PE Header, Sections, Imports and Strings. The mean value of the Sections group is the only one with a significant value, with values almost at 5 × 10 6, and all other values are close to zero. The green bar shows that the data set is mainly composed of features related to the sections. This indicates that the contribution of the features from section-level attributes is the highest in the model.

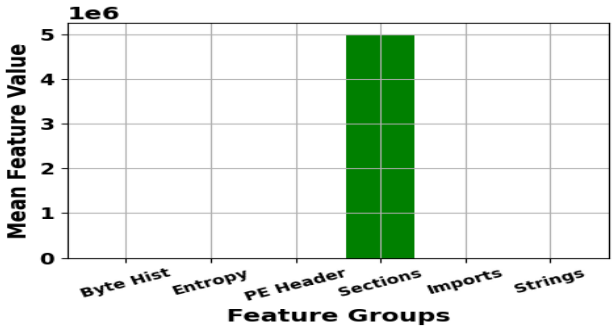

Fig 3: Mean Feature Value Across Feature Groups

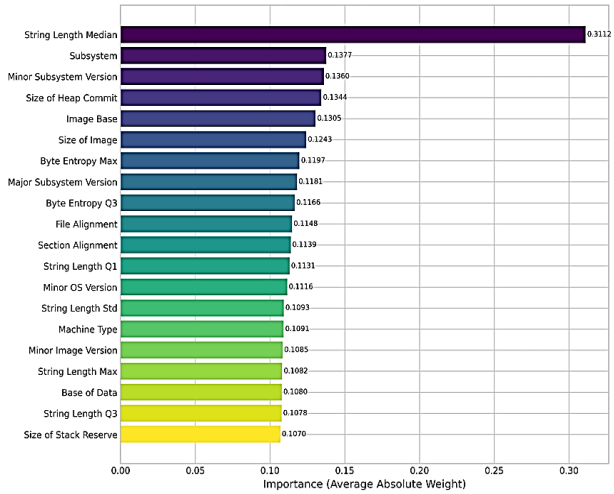

Fig 4: Top 20 Most Important Features score

Fig. 4 is the rank of the top 20 which contributes the most to the model's in predicting the rank, according to average absolute weight importance. The most influential feature is the string length Median (0.3112), followed by Subsystem and Minor Subsystem Version. The Size of Heap Commit, Image Base and Byte Entropy Max are some of the other parameters. The color gradient (purple to yellow) is clearly visible, marking the relative importance of the various parameters: string related and subsystem parameters are the most important in the decision-making of the model.

3.2. Data Pre-Processing

The first step in the pre-processing is to load the data files into NumPy arrays to easily handle the feature vectors with 2381 dimensions. Samples marked with −1 are removed to obtain a well-balanced training set consisting of 600K samples and a test set consisting of 200K samples. The features are then organized into 6 logical groups (Byte Hist, Entropy, PE Header, Sections, Imports, Strings) for targeted analysis and temporary arrays are emptied for optimization of memory for model training. This provides a clean, balanced and repeatable basis for subsequent and model training.

3.3. Label Encoding

Label Encoder transforms the malware and benign class labels into numbers which can be utilized with ML models. The original labels are mapped to integer values and class distribution is maintained in this process. The labels are encodings and the number of classes and balance ratios are displayed to be aware of the imbalance in the dataset in training a model.

3.4. Data Scaling

It used a common range to scale numerical quantities for improved performance[22]. Feature variance occurs when one feature predominates over other characteristics in the dataset due to improper scaling of the features. Data standardization was executed to evade this issue. Equation (1) is used to determine the standard score of x in the standard scaler:

$$z = \frac{z-\mu}{s} \qquad (1)$$

Where, u= Mean of training samples, s= Standard deviation of training samples. Subsequently, the neural network model is provided with this scaled data for additional training.

3.5. Data Splitting

The cleaned EMBER dataset was then split into two parts, training set and testing set in the ratio of 80:20. The 20% of the samples are used for the test and performance evaluation, and the 80% of samples are used for training the models. This division enables better model generalization and unbiased evaluation by learning on unseen malware samples.

3.6. Propose Classifiers

The proposed malware classification system uses two ML classifiers: RF and MLP to differentiate between benign and malicious executable files. RF is used because of its robustness and its ability to deal with high dimensional features, while MLP learns complex nonlinear patterns using the architecture of neural networks.

3.6.1. Random Forest (RF)

RF is a supervised ensemble ML method which gives higher classification accuracy and also prevents overfitting by using multiple DT [23]. In the training phase the model takes samples of data, called bootstraps, and randomly selects features to build different trees and the prediction is made using the majority voting. The RF model is trained with 100 decision trees (n_estimators = 100) having a maximum depth of 20 (max_depth = 20), Gini index criterion, and random_state = 42. The parameters min_samples_split = 2 and min_samples_leaf = 1 are used to enhance the stability and the classification performance of the model.

3.6.2. Multi-layer perceptron

MLP is a feed-forwardANN that has InputLayer, HiddenLayer and OutputLayer with connected weighted neurons. The model learns complex nonlinear relations with activation functions and iterative optimization of the weights. The features of input are propagated forward through the hidden layers, and the prediction errors are minimized by the application of the backpropagation algorithm and gradient-based optimization during training. The network updates weights and biases continuously over numerous

epochs to ensure a more accurate learning. The MLP model is trained via the following hyper parameters:

- HiddenLayerSizes (hidden_layer_sizes) = (128, 64)
- ActivationFunction = ReLU
- Solver = Adam
- LearningRate = 0.001
- BatchSize = 32
- Maximum Iterations (max_iter) = 200
- Alpha (Regularization Parameter) = 0.0001
- Early Stopping = True
- Random State = 42

These hyperparameters improved convergence speed, reduced overfitting, and enhanced classification accuracy for the proposed system.

3.7. Model Performance

To assess the competency of the classifiers, assessment measures including AUROC, f1-score, recall, accuracy, and precision are used. Confusion matrixes are employed to manipulate this measurement, which are matrix-like representations of the predicted class in comparison to the actual class. A variety of evaluation metrics are adjusted, as provided in Equations (2) through (5):

$$Accuracy = \frac{\mathrm{TP+TN}}{\mathrm{TP+Fp+TN+FN}} \quad (2)$$

$$Precision = \frac{\mathrm{TP}}{\mathrm{TP+FP}} \quad (3)$$

$$Recall = \frac{\mathrm{TP}}{TP+FN} \quad (4)$$

$$F1-score = 2 \times \frac{Precision \times Recall}{Precision+Recall} \quad (5)$$

Accuracy is the overall correct rate and Precision and Recall are the numbers of false/absolute positives, respectively. The F1score combines the Prec and Rec in a harmonic mean. Finally, the Area Under the ROC Curve (AUROC) is assessed at any threshold of classification by the model.

4. Results and Discussion

The tests are carried out on a state-of-the-art computing system comprising: NVIDIA GPUs with TensorCores (NVIDIA, SantaClara, CA, USA) and 64 GB of RAM, which are crucial for effectively executing complicated DL models and managing big datasets. The software stack consists of Python 3.13 and libraries that are used for constructing models and running experiments successfully, such as scikit-learn, TensorFlow 2.16.1, and PyTorch 2.5.0. The experimental results of the proposed MLP and RF in terms of malware classification are presented in Table II. The accuracy of MLP model is 97.63% and AUC is 99.13%, whereas the accuracy of RF model is 94.66% and the AUC is 92.25%, which shows good performance in malware classification.

Table 2: Experiment Results oF Propose Models for Malware Classification

Measures	MLP	RF
Accuracy	97.63	94.66
Precision	98.75	90.33
Recall	95.43	88.67
F1	95.44	88.81
TPR	97.54	91.12
AUC	99.13	92.25

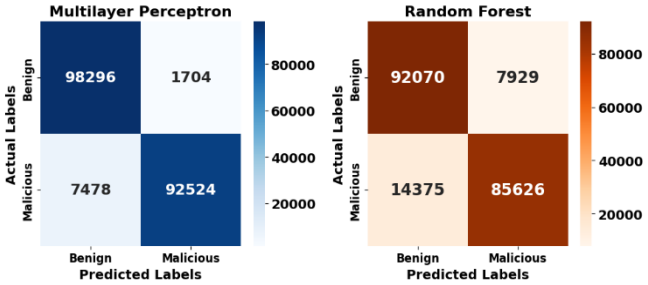

Fig 5: Confusion Matrices for MLP and RF Models

Fig. 5 displays the Confusion Matrices of the MLP and RF models that were used for malware classification. The MLP model is more successful in terms of classification, with 98,296 true benign predictions and 92,524 true malicious predictions, and fewer false predictions than the Random forest model. In contrast, the RF model showed relatively high levels of FP and FN, suggesting lower accuracy in detection. The outcomes show that the Multilayer Perceptron (MLP) model has greater power and reliability to differentiate between benign and malicious samples.

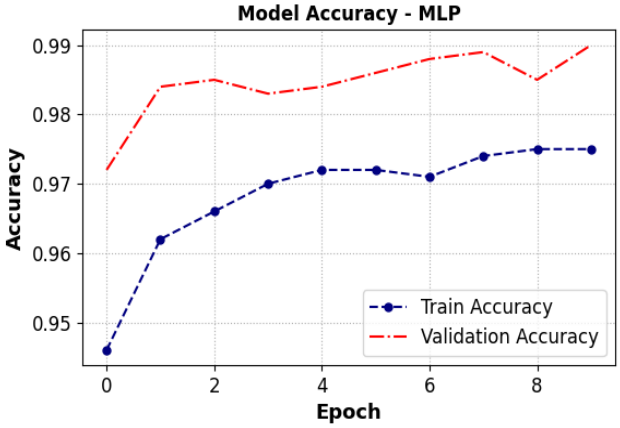

Fig 6: Training and Validation Accuracy Across Epochs for MLP

An accuracy of the MLP model during training and validation is shown over 10 epochs in Fig. 6. Train Accuracy (blue dashed line with circular markers) is close to 0.95 at the beginning and gradually raises to about 0.97 in the last epoch. The red dash-dotted curve is the Validation Accuracy, which is close to 0.97 in the beginning and fluctuates slightly between 0.98–0.99, showing good generalization and minimize overfitting. These two curves are quite similar in all parts of training; this implies that the model continues to work well and is stable in all areas of training.

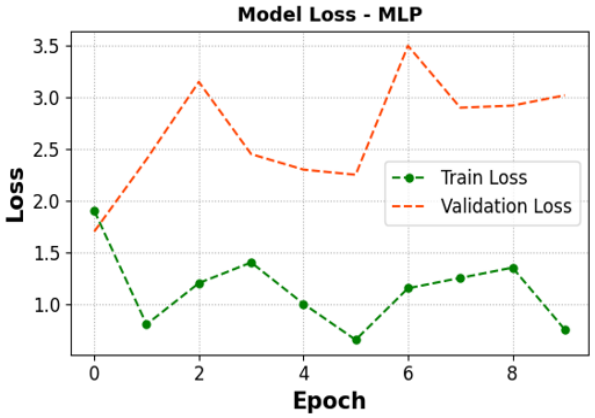

Fig 7: Training and Validation Loss of the MLP Model

Fig. 7 displays the training and validation values of the Multilayer Perceptron (MLP) model with the number of epochs. The overall trend of Training Loss shows a gradual decrease over time, reflecting the gradual learning process and optimization of model parameters. The model's capacity to generalize to new data is shown by the validation loss, which shows some slight fluctuations but is still within allowable bounds. The observed trend indicated the effectiveness of MLP model in reducing the classification errors in the process of malware detection.

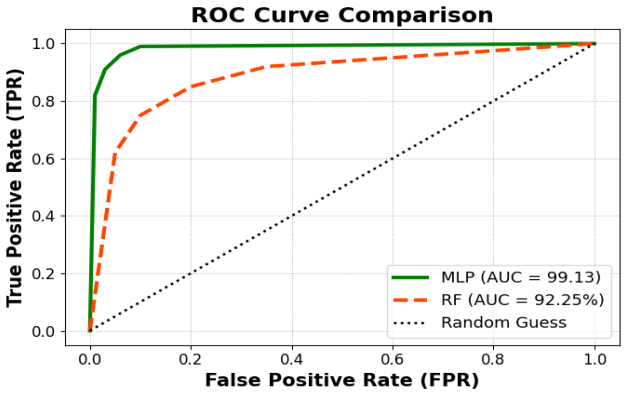

Fig 8: ROC Curve Comparison Between MLP and RF Models

Fig. 8 shows the ROC curve comparison between the MLP model and the LGBM model. The MLP model presented an AUC=99.13%, which resulted in a significant difference in the classification results compared to the other models, suggesting good discrimination ability between the classes. The RF model, achieved an AUC of 92.25%, which is comparatively lower in performance. The diagonal dotted line shows the baseline performance of the random classifier. The ROC analysis shows that the MLP model has a higher TPR in all the different FPRs, which makes it a good and robust model for prediction tasks with high accuracy

4.1. Comparative Analysis

Table III compares the performance of base and propose models. In base models, Decision Tree model achieved 94.7% accuracy with balanced prec, rec, and F1score values around 95%, indicating stable classification performance. The Logistic Regression algorithm had the lowest accuracy at 68.71% and F1 score at 68.40%, which indicates a lesser ability to deal with complex patterns. Within deep learning methods, LSTM attained accuracy of 94.58%, the BERT-GPT2 model gave an accuracy of 90.75% and 91% recall, exhibiting encouraging performances for advanced representation learning. The MLP and RF models outperform the standard and DL models in terms of accuracy, demonstrating superior prediction efficiency, generalizability, and classification capabilities.

Table 3: Comparative Performance Analysis of Ml and Dl Models

Models	Accuracy	Precision	Recall	F1
DT[24]	94.7	95	95	95
LR [25]	68.71	69.50	68.70	68.40
KNN[26]	94.2	94.6	93.8	94.2
BERT - GPT -2 [27]	90.75	-	91	91
SVM	82.64	82.81	82.03	81.74
LSTM [28]	94.58	95.41	94.54	94.97
MLP	97.63	98.75	95.43	95.44
RF	94.66	90.33	88.67	88.81

The proposed malware classification framework has demonstrated a better capability in detecting malware through effective preprocessing, feature analysis and optimized ML models. The approach improves classification accuracy, generalization and minimizes misclassification in the real world applications of cybersecurity. Moreover, it also serves as a benchmark dataset, which is also accessible to the public and thus does not use personal or sensitive data; this guarantees privacy, transparency and ethical compliance. The system under construction will be deployed for defense to aid in the safe and responsible detection of malware.

5. Conclusion and Future Work

The constant evolution of malware and its ability to evade traditional security measures remains a key challenge in the cybersecurity landscape. The results show that the MLP model is more accurate than the RF model, and has greater discriminatory power and the ability to classify files as dangerous or benign. The proposed model had an accuracy of 97.63% and AUC as 99.13% while the RF model gave an acc of 94.66% with an AUC of 92.25%. Moreover, its predictive capabilities and generalization power are compared with other models like LR, SVM, KNN, DT, LSTM and BERT–GPT2, showing superior predictive performance and generalization. The results confirm the feasibility of the proposed solution to improve the intelligence of the cybersecurity system and malware detection. But the framework is assessed against a benchmark set of malware, and that may not fully represent the diversity of malware or new threats in the wild. Additionally, as the attack patterns change, the performance of the model can

be affected in the long-term. Future research should focus on the creation of deployable real-time environments, explainable AI techniques, hybrid DL architectures, and adaptive learning approaches to enhance detection, robustness, and scalability of zero-day malware attacks.

References

[1] R. rao Thallada and N. Alapati, "Privacy and Cybersecurity Convergence: GRC Controls for Data Protection," *J. Bus. Manag. Stud.*, vol. 8, no. 5, pp. 42–48, March, 2026, doi: 10.32996/jbms.

[2] S. Irfan, "Identification of Financial Fraud Transactions: A Cybersecurity via Machine Learning Methods," in *2026 IEEE International Conference on AI Engineering and Innovations (AIEI)*, NIT Jamshedpur, India: IEEE, 2026, pp. 1–6, May. doi: 10.1109/AIEI69164.2026.11497127.

[3] B. Mohan, V. R. Surasani, and R. Kumar, "Autonomous Data Stewardship: Multi-Agent AI for Real-Time Master Data Management in Financial Services," in *2026 IEEE 16th Annual Computing and Communication Workshop and Conference (CCWC)*, Las Vegas, NV, USA: IEEE, 2026, pp. 0689–0698, February. doi: 10.1109/CCWC67433.2026.11393832.

[4] B. Pargi, D. S. Degadwala, and M. Joshi, "Hybrid Malware Analysis using Static and Dynamic Techniques with Machine Learning," *Int. J. Sci. Res. Sci. Eng. Technol.*, vol. 13, no. 1, pp. 22–27, Jan. 2026, doi: 10.32628/IJSRSET2613101.

[5] V. K. Bollu, "Threat Landscape in Artificial Intelligence Systems: Taxonomy, Attack Vectors and Security Implications," *World J. Adv. Res. Rev.*, vol. 29, no. 1, pp. 285–294, January, 2026, doi: https://doi.org/10.30574/wjarr.2026.29.1.0007.

[6] J. B. Mehta, "Securing Test Automation in Zero Trust Architectures: A Framework for Continuous Verification," in *2025 International Conference on Computer and Applications (ICCA)*, IEEE, Dec. 2025, pp. 1–5. doi: 10.1109/ICCA66035.2025.11430950.

[7] R. Alguliyev, R. Aliguliyev, and L. Sukhostat, "Radon transform based malware classification in cyber-physical system using deep learning," *Results Control Optim.*, vol. 14, p. 100382, Mar. 2024, doi: 10.1016/j.rico.2024.100382.

[8] S. Priyadarshini, C. Althati, M. Tomar, K. R. Jinna, T. Pichaimani, and V. P. Rambabu, "A Scalable Digital Twin Architecture for Intelligent Cyber Physical Systems," in *2026 International Conference on Machine Learning and Autonomous Systems (ICMLAS)*, Bangkok, Thailand: IEEE, 2026, pp. 1841–1847, March. doi: 10.1109/ICMLAS67792.2026.11483673.

[9] J. Singh and J. Singh, "A survey on machine learning-based malware detection in executable files," *J. Syst. Archit.*, vol. 112, p. 101861, Jan. 2021, doi: 10.1016/j.sysarc.2020.101861.

[10] A. Pathak, U. Barman, and T. S. Kumar, "Machine learning approach to detect android malware using feature-selection based on feature importance score," *J. Eng. Res.*, vol. 13, no. 2, pp. 712–720, Jun. 2025, doi: 10.1016/j.jer.2024.04.008.

[11] D. Javaheri, H. Chizari, M. Fahmideh, M. H. Nadimi-Shahraki, and J. Hur, "DeepRadar: A cyber-defence interceptor for early warning and defusing malware injection attacks," *Knowledge-Based Syst.*, vol. 331, p. 114830, Jan. 2026, doi: 10.1016/j.knosys.2025.114830.

[12] J. B. Mehta, "Autonomous Patch Validation for Zero-Day Exploits in Enterprise Clouds," *Int. J. Appl. Math.*, vol. 38, no. 4s, pp. 1270–1285, Oct. 2025, doi: 10.12732/ijam.v38i4s.685.

[13] D. Bhattacharjee, "Design and Evaluation of Deep Generative AI Model for Intrusion Detection in Cyber Threat Monitoring," in *2025 7th International Symposium on Advanced Electrical and Communication Technologies (ISAECT)*, Mohali, Punjab, India: IEEE, 2025, pp. 1–6, December. doi: https://doi.org/10.1109/ISAECT68904.2025.11318752.

[14] B. Madupati, M. M. Mohammed, L. Upadhyay, D. P. Guda, K. Kaushik, and M. Soni, "Integrating Artificial Intelligence with Cybersecurity for Resilient Wireless Communication Against Advanced Threats," in *2025 International Conference on Artificial Intelligence and Machine Vision (AIMV)*, IEEE, Aug. 2025, pp. 1–5. doi: 10.1109/AIMV66517.2025.11203666.

[15] S. K. Chintagunta and S. Amrale, "Enhancing Cloud Database Security Through Intelligent Threat Detection and Risk Mitigation," *Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol.*, vol. 8, no. 3, November-December, pp. 756–768, 2022, [Online]. Available: https://ijsrcseit.com/CSEIT22556

[16] P. R. S and S. B. C. K, "AI Driven Exploit Mitigation for Zero Day Vulnerability Using SVM and Autoencoder," in *2025 3rd International Conference on Advancements in Electrical, Electronics, Communication, Computing and Automation (ICAECA)*, 2025, pp. 1–4. doi: 10.1109/ICAECA63854.2025.11012512.

[17] K. Dickey, D. Hwang, and D. Kim, "Analyzing Various Machine Learning Approaches for Detecting Android Malware," in *Conference Proceedings - IEEE SOUTHEASTCON*, 2024. doi: 10.1109/SoutheastCon52093.2024.10500178.

[18] W. Cassel and N. E. Majd, "A Lightweight Obfuscated Malware Multi-class Classifier for IoT Using Machine Learning," in *2024 International Conference on Computing, Networking and Communications (ICNC)*, IEEE, Feb. 2024, pp. 239–243. doi: 10.1109/ICNC59896.2024.10555986.

[19] A. Sharma and H. Babbar, "An Analysis of Android Malware and IoT Attack Detection with Machine Learning," in *2023 3rd International Conference on Intelligent Technologies (CONIT)*, IEEE, Jun. 2023, pp. 1–5. doi: 10.1109/CONIT59222.2023.10205931.

[20] B. Bokolo, R. Jinad, and Q. Liu, "A Comparison Study to

Detect Malware using Deep Learning and Machine learning Techniques," in *2023 IEEE 6th International Conference on Big Data and Artificial Intelligence (BDAI)*, IEEE, Jul. 2023, pp. 1–6. doi: 10.1109/BDAI59165.2023.10256957.

[21] H. Broome, Y. Shrestha, N. Harrison, and N. Rahimi, "SMS Malware Detection: A Machine Learning Approach," in *2022 International Conference on Computational Science and Computational Intelligence (CSCI)*, IEEE, Dec. 2022, pp. 936–941. doi: 10.1109/CSCI58124.2022.00167.

[22] M. R. C. MUKKOLAKKAL, "IntelliStore: An Intelligent AI Agent Framework for Autonomous Storage and Database Optimization in Cloud-Native Microservices," *Int. J. Sci. Res. Mod. Technol.*, vol. 3, no. 12, pp. 243–250, Dec, 2024, doi: https://doi.org/10.38124/ijsrmt.v3i12.1024.

[23] D. R. Arikkat *et al.*, "DroidTTP: Mapping android applications with TTP for Cyber Threat Intelligence," *J. Inf. Secur. Appl.*, vol. 93, p. 104162, Sep. 2025, doi: 10.1016/j.jisa.2025.104162.

[24] R. D, A. T, and T. M, "Malware Classification Using Machine Learning and Deep Learning: A Comprehensive Approach," *Cureus J. Comput. Sci.*, Jul. 2025, doi: 10.7759/s44389-025-05024-y.

[25] M. Ababneh, A. Al-Droos, and A. El-Hassan, "Modern Mobile Malware Detection Framework Using Machine Learning and Random Forest Algorithm," *Comput. Syst. Sci. Eng.*, vol. 48, no. 5, pp. 1171–1191, 2024, doi: 10.32604/csse.2024.052875.

[26] V. R. and S. K.P., "DeepMalNet: Evaluating shallow and deep networks for static PE malware detection," *ICT Express*, vol. 4, no. 4, pp. 255–258, Dec. 2018, doi: 10.1016/j.icte.2018.10.006.

[27] D. B. D. G. Gabin, D. D. Jerome, K. Koffi, and O. Souleymane, "Innovation in Cyber Threat Detection: Transformer-Based Approach," *Int. J. Adv. Res.*, vol. 12, no. 11, pp. 1375–1389, Nov. 2024, doi: 10.21474/IJAR01/19953.

[28] H. Alkahtani and T. H. H. Aldhyani, "Artificial Intelligence Algorithms for Malware Detection in Android-Operated Mobile Devices," *Sensors*, vol. 22, no. 6, p. 2268, Mar. 2022, doi: 10.3390/s22062268.