Harnessing Unconventional Malware Detection Techniques to Equip Proactive and Resilient Cyber Defense Strategies Against the Constantly Changing Landscape of Sophisticated Cyber Threats

Raad Ali Hashmi, Muhammad Majid Adeel, Awais Khan
Department of Computer Science, Bahria University, Islamabad, Pakistan
*Correspondence: raadalihashmi@gmail.com, mmajidad33l@gmail.com, kkawaiskhan@yahoo.com

Citation | Hashmi. R. A, Adeel. M. M, Khan. A, "Harnessing Unconventional Malware Detection Techniques to Equip Proactive and Resilient Cyber Defense Strategies Against the Constantly Changing Landscape of Sophisticated Cyber Threats", IJIST, Vol. 07 Issue. 04 pp. 2527-2535, October 2025
Received | August 23, 2025 **Revised |** September 17, 2025 **Accepted |** September 19, 2025 **Published |** October 22, 2025.

The rapidly changing intricacy of malware, specifically in the case of highly secured air-gapped networks, necessitates proactive and resilient detection mechanisms that are highly capable of detecting sophisticated, modern, and obfuscated malware. Instant work focuses on a model for malware detection that is vibrant and resilient and uses deep learning models to look at Windows API call patterns that come from executable files. The original dataset from Kaggle had a big class imbalance (malicious: 42,797; benign: 1,079), but the SMOTE approach helped balance the training data. In this regard, a comparison of seven deep learning models, including Simple ANN, MLP, DropConnect Improved ANN, Residual ANN, DenseNet ANN, RBF Network, and hybrid CNN-LSTM, has been conducted over both 50 and 150 training epochs on various metrics such as recall, accuracy, F1-score, precision, and ROC-AUC. As a result, the CNN-LSTM model, enhanced by an attention mechanism, exhibited superior efficacy in differentiating between benign and malicious samples. In this context, the accuracy improvement is minimal at +0.08%, but the most substantial increase in Class 0 recall is +4.1%, and the F1-score shows an enhancement of +2.7%. The most significant contribution of this study is the attention-augmented architecture that apparently diminishes interpretability and enhances focus on significant behavioral attributes.

Keywords: Deep Learning, Malware Detection, API Call Patterns, Attention Mechanism, CNN-LSTM, SMOTE, Cybersecurity

Introduction:

As we see changes caused by the digital revolution, cyber-attacks have increased, with malware being among the biggest threats. Different types of malware can spread in systems & networks and are classified as viruses, worms, Trojan horses, ransomware, spyware, adware, rootkits, key-loggers and botnets. While malware grows more sophisticated and prevalent, intricate methods of detecting and analyzing the same have emerged. Malware analysis is the process of analyzing malicious software to gain knowledge about its functionality, origin, behavior, and effect. The primary objective is to identify, neutralize, and avoid future attacks through effective countermeasures. Various categories of techniques utilized in malware analysis include static analysis, dynamic analysis, and hybrid analysis.

Static analysis analyzes malware code or binary without running it, but is useless against obfuscated or packed malware. Dynamic analysis monitors malware activity in a controlled environment, but involves execution and is resource-consuming. To get the best information, auditors conduct hybrid analysis by using both static and dynamic auditing methods, but this approach needs more effort and resources. Some methods we use for malware detection are by signature, heuristics, behavior, machine learning, and hybrid analysis. When the malicious code's byte pattern does not match a known signature, signature-based detection fails. Such detection methods go beyond specific threat signatures and inspect a process's structure and logic. Behavior-based detection monitors a program as it is running, but it is costly to implement and carries serious security risks. When training machine learning, you use both malicious and clean files; however, many diverse files are needed for it to be beneficial. Being able to detect threats in multiple ways, hybrid malware detection guarantees results, but adds more costs.

Methods for locating malware, including signatures, behavior, heuristics, and examination of code, are studied in the research. Although signatures work quickly for detected malware, they must be updated often and still fail to protect against anything new. Using heuristic testing, malware is found by examining typical and suspicious code, although it also causes many more false alarms since its rules are not as accurate as they should be. As you click on commands, these watches will change their behavior, though if sandboxing isn't used, their actions might be risky because they use many resources. To detect malware using machine learning, we rely on models that have received a lot of information and use many computer resources. To improve detection, hybrid detection relies on analyzing code at runtime, scanning it with computer programs, and using machine learning, yet it is both complex and demanding on computer resources.

The research will design and assess an ML-based system for high-level malware detection, with an emphasis on a sample of benign and malicious files, dataset sanitization, and design of the ML-based system, along with comparison of its performance through statistical metrics. The work contributes to proactive threat discovery and provides a comparison of different ML methods for upcoming security systems. The dataset considered for this research is available in the public domain, but its high computational demand restricts its applicability.

The rest of the paper is organized in a way that Section II reviews the work done previously on malware detection techniques. Section III describes the research methodology. Section IV presents the experimental setup and results, and its associated steps. Section V concludes the paper.

Related Work:

Researchers have proposed a wide range of techniques for malware detection, transitioning from traditional approaches to more sophisticated machine learning (ML) and deep learning (DL) strategies. Various techniques proposed by previous researchers have been elaborated as under:

Wei et al. [1] proposed an LSTM-attention-based model using API call sequences to detect APT malware, achieving 99.2% accuracy. The study also used transfer learning to overcome the challenge of limited APT samples.Tahir [2] emphasized the limitations of traditional detection techniques such as signature and heuristic-based methods, advocating for hybrid approaches and intelligent methods like ML to combat evolving malware. Gibert et al. [3] presented a comprehensive review of ML applications in malware classification, highlighting feature selection challenges and the potential of DL and multimodal approaches to enhance detection accuracy. Alsmadi and Alqudah [4] traced the evolution of malware detection from classic signature-based methods to ML-based techniques, stressing the value of AI in identifying zero-day threats.

El Merabet and Hajraoui [5] categorized ML algorithms for static, dynamic, and hybrid detection and emphasized the importance of adaptive and intelligent models to counter obfuscated threats. Li et al. [6] surveyed feature selection methods, classifying them into filter, wrapper, and embedded categories. They stressed aligning selection techniques with data characteristics to optimize performance. Aslan and Samet [7] discussed both traditional and emerging detection strategies, including deep learning, mobile, IoT, and cloud-based methods, while outlining the limitations in creating general detection systems. Dong Shu and Nie [8] combined CNN and DNN architectures for Android malware detection, demonstrating superior accuracy in classifying malicious applications.

Yao et al. [9] employed Bi-LSTM and API call sequences to achieve 93.68% detection accuracy on a large mobile malware dataset. Chen and Cao [10] introduced VMCTE, which converts malware binaries into grayscale images and uses fine-tuned CNN models (e.g., ResNet50) with ensemble classifiers to achieve high performance on the Malimg dataset.

Yadav et al. [11] utilized EfficientNet-B4 CNNs and ensemble models for Android malware detection, achieving 95.7% binary classification accuracy.Azad et al. [12] developed DEEPSEL, a deep learning system that uses particle swarm optimization and behavioral features to detect Android malware, reporting an 83.6% accuracy rate. Kinkead et al. [13] improved the explainability of CNNs using opcode sequences and LIME-based interpretations, successfully highlighting critical malware features. Euh et al. [14] proposed feature dimensionality reduction using WEM images and low-dimensional attributes, demonstrating that ensemble methods like XGBoost yielded the highest accuracy and AUC. Bilot et al. [15] presented a review on malware detection using graph neural networks (GNNs), showcasing their ability to model complex malware behavior and resist adversarial attacks.

Gaber, Ahmed, and Janicke [16] conducted a systematic literature review on AI-based malware detection, emphasizing the role of high-quality datasets and the limitations imposed by evasive malware. Bensaoud, Kalita, and Bensaoud [17] evaluated DL-based detection across multiple operating systems and discussed the importance of interpretable models and adversarial defense strategies. Qureshi et al. [18] reviewed DL applications in IoT malware detection and forensics, noting critical gaps in real-time detection capabilities and anti-forensic resilience. Maniriho, Mahmood, and Chowdhury [19] examined DL techniques for desktop and mobile platforms, focusing on feature extraction, optimization methods, and future improvements in detection efficiency. Yunmar et al. [20] advocated for hybrid static-dynamic analysis using system call sequences and ML models to improve Android malware detection. Kim et al. [21] developed a compact data design strategy for ML training, reducing data usage by 57% while maintaining 99% accuracy.

Methodology:

The section describes how researchers adopted a systematic process to detect malware through API call sequence datasets. The approach involves data preprocessing, handling imbalanced datasets, model construction, and evaluation using standard

performance metrics. The dataset consists of malware and benign samples. After preprocessing data for integrity and eliminating unnecessary identifiers, the dataset was divided into training and test sets based on stratified sampling. Since there existed a class imbalance, the SMOTE was used on the training set to create synthetic samples of benign images and acquire class balance. Seven deep learning algorithms were trained based on binary cross-entropy loss and the Adam optimizer, with training for 50 and 150 epochs. Performance was evaluated with measures like accuracy, precision, recall, F1-score, ROC-AUC, and confusion matrix-based findings, noting both malware and benign classification accuracy.

Malware and benign samples obtained through API call sequences are included in the dynamic behavior-based malware collection collected from Quo Vadis Malware Dataset (Kaggle), featuring 306 API calls. In malware analysis, API call sequences are well known for their capacity to capture actual executable file behavior patterns. Dynamic API-based features are more difficult to work with and more accurately represent the purpose and functioning of the code than static features like byte-level signatures, which can be disguised. They are therefore perfect for proactive malware threat identification. Through API invocations, each record in the dataset depicts the behavior of a program during runtime.

An 80/20 split served to divide the dataset into training and evaluation groups in the deep learning process. The training of models used an 80% portion of the available dataset, while testing occurred through the evaluation of the 20% subset. The 20% of data reserved in the Testing Set enables evaluation of model generalization for unseen data. Random sampling occurred through eh split process to establish subsets capable of maintaining equal classes in both partitions. A representative combination of benign and malicious samples exists throughout all subsets due to this approach [22].

The experimental results obtained from training and evaluating the selected machine learning models on the malware dataset are presented in this section. The best-performing approach was identified based on a comprehensive comparison of the evaluation metrics across all models. Based on the evaluation metrics, the CNN-LSTM model demonstrated the most promising results for malware classification on this dataset. It consistently achieved high scores across accuracy, precision, recall, and F1-score, indicating a strong ability to correctly classify both benign and malicious samples. The combination of CNN for feature extraction and LSTM for sequence learning likely contributed to its superior performance. The following models were evaluated:

Simple ANN Model
MLP Model
Enhanced Drop Connect ANN Model
Residual ANN Model
DenseNet ANN Model
RBF Network Model
CNN-LSTM model

The performance report for the CNN-LSTM model shows high performance in classifying malware (class 1) with a high precision of 0.9918, recall of 0.9970, and F1-score of 0.9944. Performance on benign samples (class 0) is lower, with a recall of 0.6890 and an F1-score of 0.7647, indicating that the model wrongly classifies some benign samples as malware (88 false positives). The overall accuracy is 98.91%, and the weighted average metrics indicate class imbalance, biased towards the majority malware class. The confusion matrix establishes that although the model captures most malware correctly, it compromises some benign classification correctness, which can be justifiable in security-critical applications where false negative minimization (i.e., missed malware detection) is a concern. Figure 1 shows the CNN_LSTM model classification and confusion matrix against 50

epochs. Moreover, training and validation loss and accuracy of the CNN-LSTM model against 50 epochs are also delineated in Figure 2.

```
Classification Report for Cnn-Lstm_model:
              precision    recall  f1-score   support

           0     0.8590    0.6890    0.7647       283
           1     0.9918    0.9970    0.9944     10686

    accuracy                         0.9891     10969
   macro avg     0.9254    0.8430    0.8796     10969
weighted avg     0.9884    0.9891    0.9885     10969


Confusion Matrix for Cnn-Lstm_model:
[[  195    88]
 [   32 10654]]
```

Figure 1. Classification report and confusion matrix of CNN LSTM Model: 50 epochs

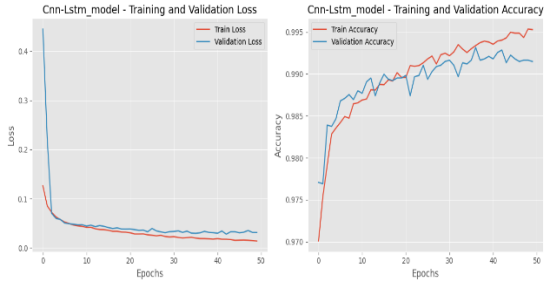

Figure 2. Training, validation loss & accuracy of CNN LSTM model: 50 epochs

Similarly, classification and confusion matrix in conjunction with training and validation loss and accuracy of each model when checked against 50 epochs are delineated in Figure 3 – Figure 14.

```
Classification Report for DenseNet_ANN_Model:
              precision    recall  f1-score   support

           0     0.9213    0.4134    0.5707       283
           1     0.9847    0.9991    0.9918     10686

    accuracy                         0.9840     10969
   macro avg     0.9530    0.7062    0.7813     10969
weighted avg     0.9831    0.9840    0.9810     10969


Confusion Matrix for DenseNet_ANN_Model:
[[  117   166]
 [   10 10676]]
```

Figure 3. Classification report and confusion matrix of DenseNet ANN: 50 epochs

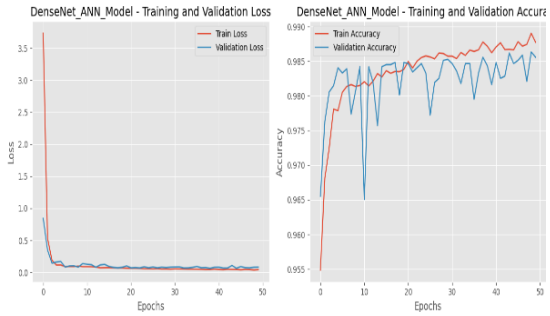

Figure 4. Training, validation loss & accuracy of DenseNet ANN: 50 epochs

```
Classification Report for Enhanced_DropConnect_ANN_Model:
              precision    recall  f1-score   support

           0     0.9490    0.3286    0.4882       283
           1     0.9825    0.9995    0.9910     10686

    accuracy                         0.9822     10969
   macro avg     0.9658    0.6641    0.7396     10969
weighted avg     0.9817    0.9822    0.9780     10969


Confusion Matrix for Enhanced_DropConnect_ANN_Model:
[[   93   190]
 [    5 10681]]
```

Figure 5. Classification report and confusion matrix Enhanced_DropConnect ANN: 50 epochs

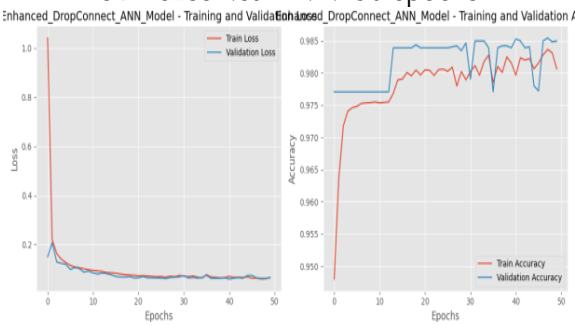

Figure 6. Training, validation loss & accuracy of Enhanced_DropConnect ANN: 50 epochs

```
Classification Report for Residual_ANN_Model:
              precision    recall  f1-score   support

           0     0.2319    0.7032    0.3488       283
           1     0.9917    0.9383    0.9643     10686

    accuracy                         0.9323     10969
   macro avg     0.6118    0.8208    0.6565     10969
weighted avg     0.9721    0.9323    0.9484     10969


Confusion Matrix for Residual_ANN_Model:
[[  199    84]
 [  659 10027]]
```

Figure 7. Classification report and confusion matrix of Residual ANN: 50 epochs

Figure 8. Training, validation loss & accuracy of Residual ANN: 50 epochs

```
Classification Report for MLP_Model:
              precision    recall  f1-score   support

           0     0.9286    0.4134    0.5721       283
           1     0.9847    0.9992    0.9919     10686

    accuracy                         0.9840     10969
   macro avg     0.9566    0.7063    0.7820     10969
weighted avg     0.9832    0.9840    0.9810     10969


Confusion Matrix for MLP_Model:
[[  117   166]
 [    9 10677]]
```

Figure 9. Classification report and confusion matrix of MLP: 50 epochs

Figure 10. Training, validation loss & accuracy of MLP: 50 epochs

```
Classification Report for RBF_Network_Model:
              precision    recall  f1-score   support

           0     0.9487    0.3922    0.5550       283
           1     0.9842    0.9994    0.9917     10686

    accuracy                         0.9838     10969
   macro avg     0.9664    0.6958    0.7734     10969
weighted avg     0.9832    0.9838    0.9805     10969


Confusion Matrix for RBF_Network_Model:
[[  111   172]
 [    6 10680]]
```

Figure 11. Classification report and confusion matrix of RBF Network: 50 epochs

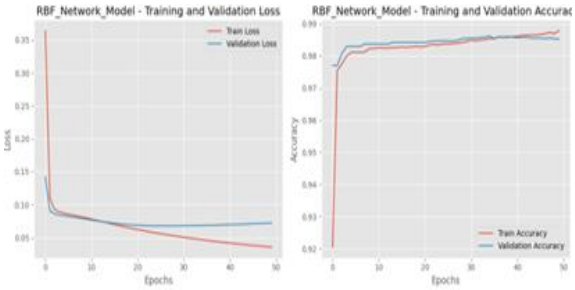

Figure 12. Training, validation loss & accuracy of RBF Network: 50 epochs

```
Classification Report for Simple_ANN_Model:
              precision    recall  f1-score   support

           0     0.9640    0.3781    0.5431       283
           1     0.9838    0.9996    0.9916     10686

    accuracy                         0.9836     10969
   macro avg     0.9739    0.6889    0.7674     10969
weighted avg     0.9833    0.9836    0.9801     10969


Confusion Matrix for Simple_ANN_Model:
[[  107   176]
 [    4 10682]]
```

Figure 13. Classification report and confusion matrix of Simple ANN: 50 epochs

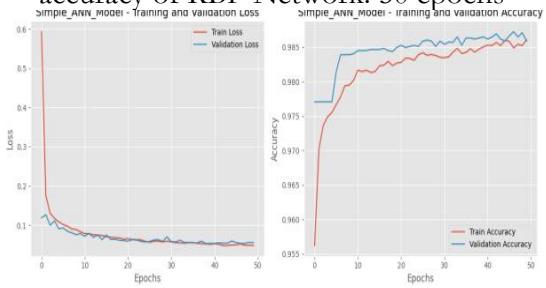

Figure 14. Training, validation loss & accuracy of Simple ANN: 50 epochs

Results and Analysis:

From the tested models' confusion matrices and classification reports against 50 epochs, it can be seen that all models are very good at identifying malware (class 1), but vary a lot in their performance in identifying benign samples (class 0) correctly. The CNN-LSTM model takes the lead with high balanced performance and overall accuracy (98.91%) with an F1-score of 0.7647 for benign samples and 0.9944 for malware. Conversely, Simple ANN and MLP models exhibit very high malware detection (recall 0.999), with low benign class recall (0.3887 and 0.3110, respectively), leading to higher false positives. DropConnect ANN with increased connectivity has failed on class 0 with zero precision, recall, and F1-score, detecting all samples as malware, which is probably due to severe class imbalance handling failure.

Residual ANN and RBF Network marginally enhance benign detection compared to MLP, whereas DenseNet ANN provides a better-balanced performance among the ANN-based models with 0.6334 F1-score for benign class and 0.9921 for malware. These findings indicate that deeper learning architectures with higher representational capacity, e.g., CNN-LSTM and DenseNet, are better at distinguishing both classes, especially in processing the minority benign class more robustly. Statistical comparison of these models against 50 epochs is shown in Figure 15:

	Model	Accuracy	F1-Score	...	False Positives (FP)	False Negatives (FN)	ROC-AUC
6	Cnn-Lstm_model	0.989060	0.994400	...	88	32	0.978354
4	DenseNet_ANN_Model	0.984593	0.992131	...	137	32	0.913760
0	Simple_ANN_Model	0.983499	0.991596	...	173	8	0.941046
5	RBF_Network_Model	0.983134	0.991406	...	170	15	0.858760
1	MLP_Model	0.980855	0.990256	...	195	15	0.890325
3	Residual_ANN_Model	0.980217	0.989907	...	173	44	0.786532
2	Enhanced_DropConnect_ANN_Model	0.974200	0.986931	...	283	0	0.932016

Figure 15. Models Performance: 50 epochs

The classification reports and confusion matrices against 150 epochs provide detailed insights into the performance of each deep learning model applied for malware detection. The Simple ANN model achieved a high accuracy of 98.4%, with excellent performance on the malware class (1), showing a precision of 0.9863 and a recall of 0.9975. However, its performance on the benign class (0) was weaker, with a recall of only 0.4770, indicating many benign samples were misclassified. The MLP model performed similarly, with a slightly lower benign recall (0.3993) but improved precision (0.9339). The Enhanced DropConnect ANN improved the balance, with higher precision and recall for the benign class (0.8812 and 0.4982, respectively), while still achieving near-perfect malware detection. The Residual ANN model showed poor overall performance, with a misleadingly high precision (0.9969) on class 1 but a drastically low recall (0.3609), leading to an overall accuracy drop to 37.6%.

In contrast, DenseNet ANN and RBF Network models showed more balanced improvements, especially in detecting benign instances (recalls of 0.5724 and 0.4488, respectively), while maintaining strong malware classification metrics. Finally, the CNN-LSTM hybrid model delivered the best overall results, achieving the highest accuracy of 99.05% and the best balance between both classes, with a benign class recall of 0.7527 and malware recall of 0.9968, showing the least number of misclassifications across the board. These results demonstrate that while most models are highly effective at detecting malware (class 1), only the CNN-LSTM model offered a robust balance by minimizing false positives and false negatives simultaneously. The summary of the performance against each model is described in Figure 16:

	Model	Accuracy	F1-Score	...	False Positives (FP)	False Negatives (FN)	ROC-AUC
6	Cnn-Lstm_model	0.990519	0.995142	...	70	34	0.970219
2	Enhanced_DropConnect_ANN_Model	0.985322	0.992510	...	142	19	0.913464
4	DenseNet_ANN_Model	0.984684	0.992166	...	121	47	0.892577
0	Simple_ANN_Model	0.984046	0.991858	...	148	27	0.913983
1	MLP_Model	0.983772	0.991734	...	170	8	0.904556
5	RBF_Network_Model	0.980217	0.989891	...	156	61	0.831111
3	Residual_ANN_Model	0.376333	0.529990	...	12	6829	0.729041

Figure 16. Model's Performance: 150 epochs

Conclusion:

Among the seven models evaluated for malware detection using API call sequences, the CNN-LSTM model achieved the highest accuracy (98.91%) and f1-score (0.9944), with the lowest false positive and false negative rates among all models. While simpler models such as Simple ANN and RBF performed well, they exhibited relatively higher false positive rates. The Enhanced DropConnect ANN failed to detect any benign samples, highlighting the impact of model architecture and training sensitivity on class imbalance. This reinforces the importance of robust architectures like CNN-LSTM for complex, sequential feature spaces such as API calls in dynamic malware analysis.

References:

[1] X. M. Chaoxian Wei, Qiang Li, Dong Guo, "Toward Identifying APT Malware through API System Calls," *Secur. Commun. Networks*, vol. 1, no. 1, 2021, doi: https://doi.org/10.1155/2021/8077220.

[2] Rabia Tahir, "A Study on Malware and Malware Detection Techniques," *Int. J. Educ. Manag. Eng.*, vol. 8, no. 2, pp. 20–30, 2018, doi: 10.5815/ijeme.2018.02.03.

[3] D. Gibert, C. Mateu, and J. Planes, "The rise of machine learning for detection and

classification of malware: Research developments, trends and challenges," *J. Netw. Comput. Appl.*, vol. 153, p. 102526, 2020, doi: https://doi.org/10.1016/j.jnca.2019.102526.

[4] T. Alsmadi and N. Alqudah, "A Survey on malware detection techniques," *2021 Int. Conf. Inf. Technol. ICIT 2021 - Proc.*, pp. 371–376, Jul. 2021, doi: 10.1109/ICIT52682.2021.9491765.

[5] S. H. Smita Ranveer, "A Survey of Malware Detection Techniques based on Machine Learning," *Int. J. Comput. Appl.*, vol. 120, no. 5, pp. 1–7, 2015, doi: 10.5120/21220-3960.

[6] K. C. Jundong Li, "Feature Selection: A Data Perspective," *ACM Comput. Surv.*, vol. 50, no. 6, pp. 1–45, 2017, doi: https://doi.org/10.1145/3136625.

[7] Ö. A. A. and R. Samet, "A Comprehensive Review on Malware Detection Approaches," *IEEE Access*, vol. 8, pp. 6249–6271, 2020, doi: 10.1109/ACCESS.2019.2963724.

[8] S. Dong, L. Shu, and S. Nie, "Android Malware Detection Method Based on CNN and DNN Bybrid Mechanism," *IEEE Trans. Ind. Informatics*, vol. 20, no. 5, pp. 7744–7753, May 2024, doi: 10.1109/TII.2024.3363016.

[9] Y. Z. Ye Yao, "Research on Malware Detection Technology for Mobile Terminals Based on API Call Sequence," *Mathematics*, vol. 12, no. 1, p. 20, 2024, doi: https://doi.org/10.3390/math12010020.

[10] J. C. Zhiguo Chen, "VMCTE: Visualization-Based Malware Classification Using Transfer and Ensemble Learning," *Comput. Mater. Contin.*, vol. 75, no. 2, pp. 4445–4465, 2023, doi: https://doi.org/10.32604/cmc.2023.038639.

[11] S. V. Pooja Yadav, Neeraj Menon , Vinayakumar Ravi, "EfficientNet convolutional neural networks-based Android malware detection," *Comput. Secur.*, vol. 115, p. 102622, 2022, doi: https://doi.org/10.1016/j.cose.2022.102622.

[12] J. A. Muhammad Ajmal Azad, Farhan Riaz, Anum Aftab, Syed Khurram Jah Rizvi, "DEEPSEL: A novel feature selection for early identification of malware in mobile applications," *Futur. Gener. Comput. Syst.*, vol. 129, pp. 54–63, 2022, doi: https://doi.org/10.1016/j.future.2021.10.029.

[13] M. Kinkead, S. Millar, N. McLaughlin, and P. O'Kane, "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.

[14] D. K. and D. H. S. Euh, H. Lee, "Comparative Analysis of Low-Dimensional Features and Tree-Based Ensembles for Malware Detection Systems," *IEEE Access*, vol. 8, pp. 76796–76808, 2020, doi: 10.1109/ACCESS.2020.2986014.

[15] A. Z. Tristan Bilot, Nour El Madhoun, Khaldoun Al Agha, "A Survey on Malware Detection with Graph Representation Learning," *ACM Comput. Surv.*, vol. 56, no. 11, pp. 1–36, 2024, doi: https://doi.org/10.1145/3664649.

[16] M. A. Matthew G. Gaber, "Malware Detection with Artificial Intelligence: A Systematic Literature Review," *ACM Comput. Surv.*, vol. 56, no. 6, pp. 1–33, 2024, doi: https://doi.org/10.1145/3638552.

[17] A. Bensaoud, J. Kalita, and M. Bensaoud, "A survey of malware detection using deep learning," *Mach. Learn. with Appl.*, vol. 16, p. 100546, 2024, doi: https://doi.org/10.1016/j.mlwa.2024.100546.

[18] F. U. Siraj Uddin Qureshi, Jingsha He, Saima Tunio, Nafei Zhu, Ahsan Nazir, Ahsan Wajahat, "Systematic review of deep learning solutions for malware detection and forensic analysis in IoT," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 36, no. 8, p. 102164, 2024, doi: https://doi.org/10.1016/j.jksuci.2024.102164.

[19] A. N. M. Pascal Maniriho, "A Survey of Recent Advances in Deep Learning Models

for Detecting Malware in Desktop and Mobile Platforms," *ACM Comput. Surv.*, vol. 56, no. 6, pp. 1–41, 2024, doi: https://doi.org/10.1145/3638240.

[20] W. and F. M. R. A. Yunmar, S. S. Kusumawardani, "Hybrid Android Malware Detection: A Review of Heuristic-Based Approach," *IEEE Access*, vol. 12, pp. 41255–41286, 2024, doi: 10.1109/ACCESS.2024.3377658.

[21] C. Y. Y. and S. N. S. -K. Kim, X. Feng, H. A. Hamadi, E. Damiani, "Advanced Machine Learning Based Malware Detection Systems," *IEEE Access*, vol. 12, pp. 115296–115305, 2024, doi: 10.1109/ACCESS.2024.3434629.

[22] Swapnaja Hiray, "Comparative Analysis of Feature Extraction Methods of Malware Detection," *Int. J. Comput. Appl.*, vol. 120, no. 5, pp. 1–7, 2015, doi: 10.5120/21220-3960.