

A Novel Optimized Approach to Detect Complex and Unknown Malware using Deep Neural Networks

Deepika Sharma^{1,2}, Manoj Devare¹

¹Amity Institute of Information and Technology, Amity University, Navi Mumbai, India.

²Department of Engineering and Technology, Bharati Vidyapeeth(Deemed to be University), Off Capmus, Navi Mumbai.

Corresponding Email: sharmadeepika817@gmail.com

ARTICLE INFO

Received: 19 Dec 2024

Revised: 10 Feb 2025

Accepted: 22 Feb 2025

ABSTRACT

Since new generations of malware become more diverse and complex nowadays the conventional approaches and methods of malware detection and categorization do not always prove to be efficient. In this work, the combined and optimized CNN, and RNN architecture for the Firefly Algorithm is suggested to accurately classify and detect complex and unseen malware. To detect subtle patterns depicting the behavior of the malware, the framework utilizes a comprehensive feature extraction process while reducing the complexity as much as possible. In our future work, we aim to expand the proposed idea to use a deeper model utilizing CNN in combination with GANs, which we would fine-tune with firefly algorithm, on the similar dataset to bolster the detection performance. The optimization algorithm enhances effectiveness and decreases the consumption of resources while retaining high detection capability. Experimental results demonstrate that the model achieves a training and validation accuracy exceeding 99% after a few epochs, showcasing its suitability for real-world, large-scale cybersecurity applications. This novel approach provides a high-accuracy solution for malware detection, incorporating innovative features for both whitelist and blacklist classification.

Keywords: Malware Detection, Deep Neural Networks (DNNs) Signature Based Techniques Cyber Security.

INTRODUCTION

Malware has continued to be on the top of the most dangerous threats in the cybersecurity space due to its continually changing and broad nature. More conventional approaches for identifying malware, such as signature based or by structural analysis do not work in the case of the newly emerging unknown, unknown and polymorphic malware. With increased advancement in the kind of threats being experienced online, there is a call for better means through which one can easily detect the different kinds of malware, with high efficiency and accuracy.

Malware detection is thus a problem that has been targeted for deep learning solutions due to the capability of deep learning in complex pattern recognition of big data. By using deep neural network one can extract the meaningful features from the behavior of the malware and, therefore, have better accuracy of the classification. However, to apply such models there are challenges that include the following: computational complexity as well as the consumption of the resources. Reducing running time of these models and, in particular, the resources they consume when being run can be essential for the application of these models in real-life cybersecurity settings.

This paper presents an overall framework to tackle these problems. It particularly concentrates on proposing and deploying an optimized deep learning architecture for enhanced learning capability to detect and classify malware efficiently with a feature extraction mechanism to minimize computing costs. In addition, optimization is being implemented to improve the effectiveness of the framework when run through an optimization algorithm to reduce the number of computations needed to find the vector. The work described here also enhances the detection capability for sophisticated and previously unseen malware and does so in a manner that can be expanded and optimized for other tasks, making it ideal for a range of cybersecurity applications. This paper outlines the problems with current approaches to malware detection, details how the proposed framework would be put in to practice and assesses the findings from experiments involving the frame. The results suggest that this approach can significantly enhance the capabilities of current methods to fight against malware by proposing a stable, fast, and resource-conscious approach. In figure.1 bar chart depicts the comparison of various Anti-Malware Techniques and results reveal the accuracy by deep learning

methods (90% whereas the old school methods like signature-based detection have an accuracy of 50%). It underlines one of the developments in the deep learning techniques in the detection of malware.

Decision trees, Support Vector Machines (SVMs), and Random Forest are many of the most typical algorithms in the context of malware detection. Both supervised and unsupervised learning models determine which files are malicious early use known data characteristics. Although the ML models derived here offer reasonable accuracy classification, their dependence on selected features restricts feature space scalability and transferability to other unknown variations of malware (Ucci et al., 2019; Ye et al., 2017). The problem with these methods is that they are fairly static – not the best approach given the continuously changing world of malware threats.

CNN models and RNN models are even more attractive as opponents of machine learning and automated feature extraction for better detection of new strands of malware. These models have been applied to measure the difference in the distance of paths obtained from executable traces, API calls and binary code representations. The studies reveal that the advancement of deep neural networks outperforms other approaches of traditional ML in aspects of detection rate (LeCun et al., 2015; Pascanu et al., 2015). However, they incur a high computational cost and high demand for resources and thus are not easily deployable in constrained environments. Other research has tried some variations of the CNNs, for example integrating them with recurrent neural networks or trying the transformer-based models which improve both accuracy and performance (Venkatraman et al., 2019; Ullah et al., 2022).

The advanced feature of XAI brings a significant aspect of fairness in the detection process since it breaks the “Dark box” challenges that deep learning present in the malware detection process. In their research, both Barredo Arrieta et al (2020) and Das & Rad (2020) focused on how interpretability should be included in AI systems even more in cybersecurity. XAI techniques give understanding of the eventual output of the model for malware classification and hence leads to better confidence and reliability. In Liu et al. (2022), the authors describe the role of XAI to provide insight on why models deliver good results so as to enhance feature selection and classification techniques.

Feature extraction is a critical step employed in designing malware detection systems, as it is designed to enhance the efficiency of the process by possibly eliminating data that contributes little to the process. Feature reduction methods like PCA and auto encoders are used to pre-process input data so that models pay attention to what is useful. To increase speed and utilization of other resources, excessive accuracy of malware detection is used modern optimization techniques such as genetic algorithm and swarm intelligence (Mustafa Majid et al., 2023; Vijayakumar et al., 2019). New studies investigate the possibilities of combining the two approaches, for instance, firefly algorithms to optimize the solution between computation time and detection accuracy. Table 1 shows the various approaches taken in past few years.

Table 1. Comparison of CNN and RNN-Based Approaches for Malware Detection (2020–2023)

Study	Year	Model	Focus/Findings	Strengths	Limitations
Venkatraman et al.	2020	CNN	Hybrid image-based malware detection using CNN for spatial feature extraction.	High detection accuracy for image-based data.	High computational cost for feature extraction.
Barredo Arrieta et al.	2020	RNN + XAI	Integration of RNNs with explainable AI (XAI) to interpret sequential data for malware classification.	Improved interpretability of results.	Limited focus on large-scale datasets.
Ullah et al.	2022	Transformers + CNN	Explored CNN and transformers for visual representation of malware combined with transfer learning.	Effective in detecting obfuscated malware.	Complex model with high resource needs.
Liu et al.	2022	CNN + RNN + XAI	Combined CNN for spatial features and RNN for temporal features, integrating XAI for model explanations.	High accuracy and transparency.	High complexity and training time.
Mustafa Majid et al.	2023	CNN + Optimized Models	Applied CNN with heuristic optimization for faster malware classification.	Improved efficiency using optimization.	Focused only on static datasets.
Vijayakumar et al.	2023	CNN + RNN	Explored hybrid models combining CNN and RNN for detecting unknown malware using sequential behavior.	Balanced spatial and temporal feature analysis.	Increased training complexity.

Ansari et al.	2023	CNN + RNN + GAN	Combined CNN and RNN with GAN for generating synthetic malware data and improving training robustness.	Effective for imbalanced datasets.	GANs add computational overhead.
---------------	------	-----------------	--	------------------------------------	----------------------------------

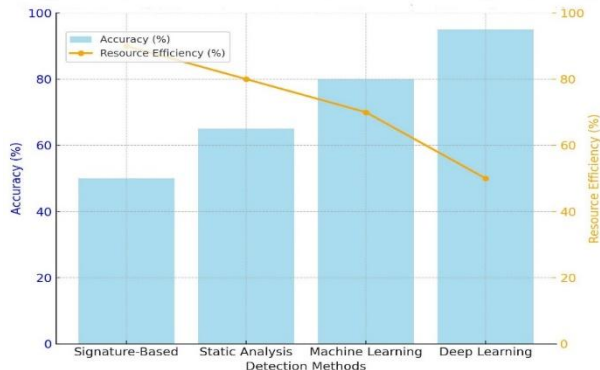


Figure 1. Comparison of Malware Detection Methods: Accuracy vs. Resource Efficiency

Figure 1 shows similarities in accuracy and resource use between different malware detection techniques. Although deep learning yields the best result, it is associated with a higher consumption of resources than other alternatives, implying an inherent compromise of precision and resource usage.

EXISTING METHODOLOGY AND ARCHITECTURES

2.1 Traditional Signature-Based Detection

Signature-based malware detection is one of the simplest and most traditional approaches, relying on identifying threats based on known patterns or signatures stored in a database. This process includes extracting called functions or byte sequences and hash-values and comparing them with a signature database a priori. Although effective for detecting known threats, this approach is therefore strongly categorized as weak when dealing with newer, unknown, or polymorphic malware primarily because it is incapable of generalization other than from previously identified patterns hence leading to high false negatives for new and emerging threats.

2.2. Heuristic-Based and Behavioral Analysis

Some of the methods are heuristic based and behavioral analysis in that they tend to identify malware based on the functions that the malware performs. These encompass observing subtle characteristics related to running malware including system call traces, memory and network behavior which are normally noticed in a sandbox context. Though useful in identifying suspicious activity, these techniques are computationally expensive because they require time-tested real-time monitoring and sandbox execution of files, and hence not ideal for massive use.

2.3. Machine Learning-Based Malware Detection

In supervised machine learning, the discrete attributes of malware like opcode sequences, system calls and permission usage, static and dynamic attributes are extracted and trained into classifiers such as Random Forests, SVMs, or gradient boosting decisions to classify benign and malicious software. Implementations of these models are learned with labeled data to predict whether a given data point is of the same class or belongs to another class. However, this approach has the following restraining factors; fresh features are hard to find and can be untrustworthy, the process of feature engineering is time-consuming, and the approaches cannot cope up with the dynamic nature of malware where static features alone are insufficient for proper detection.

2.4. Deep Learning for Malware Detection

Traditional methods included use of rule-based or signature matching, where a specific sequence of instructions or algorithm was used to detect malware, deep learning models like CNN and RNN automatically detect features from the raw data without human interjection apart from programming the autoencoder. All these methods have been proved to hold the potential to enhance the accuracy of the detection of malware.

2.5. Convolutional Neural Networks (CNNs)

Malware can be given grayscale images, or it can be byte plotted for spatial features to be classified. Organizing the malware by the visual representation of these visual architectures allows for easy tracking of patterns and structures present. However, a major drawback of this method is that standard analysis tools based on the Convolutional Neural Networks (CNNs), which are used for frequency feature extraction, consider only spatial properties and do not take into account temporal characteristics inherent in the malware or in the sequence of its actions. This limitation is the reason why there is the need to add methods that capture both the spatial and temporal patterns for a better analysis of the malware.

2.6. Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) are used in malware classification to process sequence data including opcode sequences or API calls to capture temporal characteristics of malware. However, the usage of RNNs has some problems: the vanishing gradient problem, and the inability to process long sequences. The integration of deep learning models to learn spatial features from the CNN models and temporal features from the RNN models make the hybrid approaches to gain better detection performance. However, these hybrid models are time-consuming in execution and may not recommend themselves for use in environments characterized by limited resources because of their complexity.

2.7. Optimization Techniques in Malware Detection

In feature selection and hyperparameter optimization, Genetic Algorithms (GA) and Particle Swarm Optimization (PSO) algorithms are utilized for malware detection to find out decision margins to enhance the classifier. These optimizations begin by working out the few features or parameters that are most informative for the detection models and therefore improving the level of accuracy in their functioning. Initially, they easily over-fit their models and are not very generalized in dealing with unseen and diverse forms of malware samples.

PROPOSED METHODOLOGY

3.1. Data Preparation and Preprocessing

Data cleaning and processing is the process without which data has to go through the foundational step of being cleaned, organized and cleaned for easy use in training deep learning models. Using the IoT-23 dataset, which contains comprehensive benign and malicious network traffic data, the data is first downloaded and read into a panda DataFrame for exploration and preprocessing. Libraries are used to load data from multiple files, standardize column names, and merge datasets into a unified structure using pandas, enabling consistent and efficient analysis. All labels except "Benign" are simplified to "Malicious" using lambda functions. This binary classification simplifies the detection task. When it comes to IP addresses, they are hashed out with SHA 256 to change the actual addresses while still keeping them unique and private. To make the computational work optimized, the number of dimensions of hashed IP addresses is reduced in order to involve groups of them like 1000 buckets and so on.

3.2. Exploratory Data Analysis (EDA) and Feature Selection

Exploratory Data Analysis (EDA) is very important in determining important details regarding data distribution and patterns while the feature selection makes it certain that the model only considered the right details. While performing data exploration statistical summaries include mean, median value and standard deviation for numerical variables which can be displayed as `df.describe()`. Gaps and twofold values are detected and treated, and condition of the dataset's form and organization is checked. To do so, different visualizations such as histograms, scatter plots and heatmaps are then used to single out patterns in the data.

A Random Forest classifier contains a method that outputs feature importance levels. This classifier selects and sorts out the ten essential features out of the features in the model depending on their respective feature importance scores. Based on SMOTE Synthetic Minority Oversampling Technique, synthetic examples are created to have a balanced quantity of samples for the minority class.

3.3. Optimization Techniques in Malware Detection

In feature selection and hyperparameter optimization, Genetic Algorithms (GA) and Particle Swarm Optimization (PSO) algorithms are utilized for malware detection to find out decision margins to enhance the classifier. These optimizations begin by working out the few features or parameters that are most informative for the detection models and therefore

improving the level of accuracy in their functioning. Initially, they easily over-fit their models and are not very generalized in dealing with unseen and diverse forms of malware samples

PROPOSED METHODOLOGY

4.1. Data Preprocessing

Using the IoT-23 dataset, which contains comprehensive benign and malicious network traffic data, the data is first downloaded and read into a panda DataFrame for exploration and preprocessing. Libraries are used to load data from multiple files, standardize column names, and merge datasets into a unified structure using pandas, enabling consistent and efficient analysis. The percentage of missing data in each column is then calculated to determine whether to impute the missing values or drop the columns entirely. The percentage of missing values in each column of the Data Frame highlights that some columns, such as service, duration, and local_orig, have significant missing values (e.g., 99.91% for service). The missing values were calculated after replacing '-' with NaN. All labels except "Benign" are simplified to "Malicious" using lambda functions. This binary classification simplifies the detection task. IP addresses are hashed out with SHA 256 to change the actual addresses while still keeping them unique and private.

Let the dataset D consist of N samples, where each sample has M features as shown in equation 1:

$$D = \{(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)\}, \quad x_i \in \mathbb{R}^M, y_i \in \{0, 1\} \dots (1)$$

where:

- x_i represents the feature vector for the i-th sample.
- $y_i=0$ for benign and $y_i=1$ for malicious labels.

After balancing via SMOTE, D becomes balanced with equal benign and malicious samples:

$$N_{\text{benign}} = N_{\text{malicious}}$$

4.2 Feature Selection Using Random Forest and Model Building

The model selects and sorts out the ten essential features out of the features in the model depending on their respective feature importance scores. Based on SMOTE Synthetic Minority Oversampling Technique, synthetic examples are created to have a balanced quantity of samples for the minority class. In current research, CNNs and RNNs are integrated in the hybrid model enabling better detection of malware. This is particularly helpful in dictating hierarchical features from the data through applying convolutional filters to enhance the identification of spatial relations in malware data. Max Pooling layers continue to further down-sampling and limiting feature dimensions while preserving important features keeping computation efficient. Since temporal characteristics of malware behavior need to be captured to model sequential nature of malware actions, Simple RNN layer is included. The extracted features are then passed into fully connected Dense layers for final probability predictions with SoftMax or sigmoid activation function. Meanwhile, in model compilation, Adam optimizer is used for optimizing the weight update and binary cross entropy loss is used for both the binary classifications.

Feature importance I_j for each feature j is calculated based on the decrease in impurity in the Random Forest:

$$I_j = \sum_{t=1}^T \frac{\Delta G_t(f_j)}{\text{Total nodes in } t} \dots (2)$$

where:

- T is the total number of trees in the Random Forest.
- $\Delta G_t(f_j)$ is the decrease in Gini impurity for feature f_j at a given split in tree t.

In equation 3 the top k features are selected based on the highest I_j :

$$F_{\text{selected}} = \{f_1, f_2, \dots, f_k\}, \quad k < M \dots (3)$$

Convolutional Layer:

The output of the l-th convolutional layer is:

$$h^{(l)}_{i,j} = \sigma \left(\sum_{m=1}^k w_m \cdot x_{i+m_j}^{(l-1)} + b^{(l)} \right) \dots (4)$$

where:

- w_m are convolutional filter weights.
- $x^{(l-1)}$ is the input to the l-th layer.
- $b^{(l)}$ is the bias term.
- Σ is the activation function (e.g., ReLU).

Max Pooling:

Max pooling reduces dimensionality:

$$p_{i,j} = \max (h_{i:i+k,j:j+k}^{(l)}). \dots \dots \dots (5)$$

Recurrent

Layer

(SimpleRNN):

The RNN processes sequential data:

$$h_t = \sigma (W_h \cdot h_{t-1} + W_x \cdot x_t + b_h). \dots \dots \dots (6)$$

where:

- h_t is the hidden state at time t.
- W_h are weight matrices for hidden and input states, respectively.
- b_h is the bias term.

Output

Layer:

The final Dense layer outputs probabilities for the two classes using the sigmoid activation:

$$y^{\wedge} = \sigma (W_o \cdot h + b_o). \dots \dots \dots (7)$$

where:

- $y^{\wedge} \in [0,1]$ represents the probability of the sample being malicious.

Loss Function

Binary Cross-Entropy (BCE) loss is used to optimize the model:

$$L = -\frac{1}{N} \sum_{i=1}^N [y_i \log (y_i^{\wedge}) + (1-y_i) \log (1-y_i^{\wedge})]. \dots \dots \dots (8)$$

4.3 Optimization Technique

To improve the efficiency and effectiveness of the model, Firefly Algorithm is included as a component in the proposed hybrid model. This algorithm designs a method to select appropriate hyperparameters like the number of neurons in the CNN and other RNN layers following the manner of fireflies. Firefly Algorithm compares a number of scenarios and chooses the best one; this optimizes model quality to computational resource usage, and therefore enhances the general performance.

RESULTS AND DISCUSSION

The Figure 1 shows multiple histograms representing the distribution of different variables or features in a dataset. Here's a breakdown of what this likely represents:

Each subplot in the histogram corresponds to a feature (or column) in the dataset, illustrating how its values are distributed. The x-axis represents the range of values for each feature, while the y-axis shows the frequency (or density) of values within each bin. Observations from the histograms reveal that some distributions are skewed, with a heavy concentration on one side, while a few variables may display outliers or extreme values, visible as spikes or long tails. Additionally, uniform distributions or features with low variance appear as flat histograms or narrow peaks. These insights suggest that skewed data may require normalization or transformation, uniform or constant features might be irrelevant for predictive models, and outliers may need to be addressed depending on the specific use case.

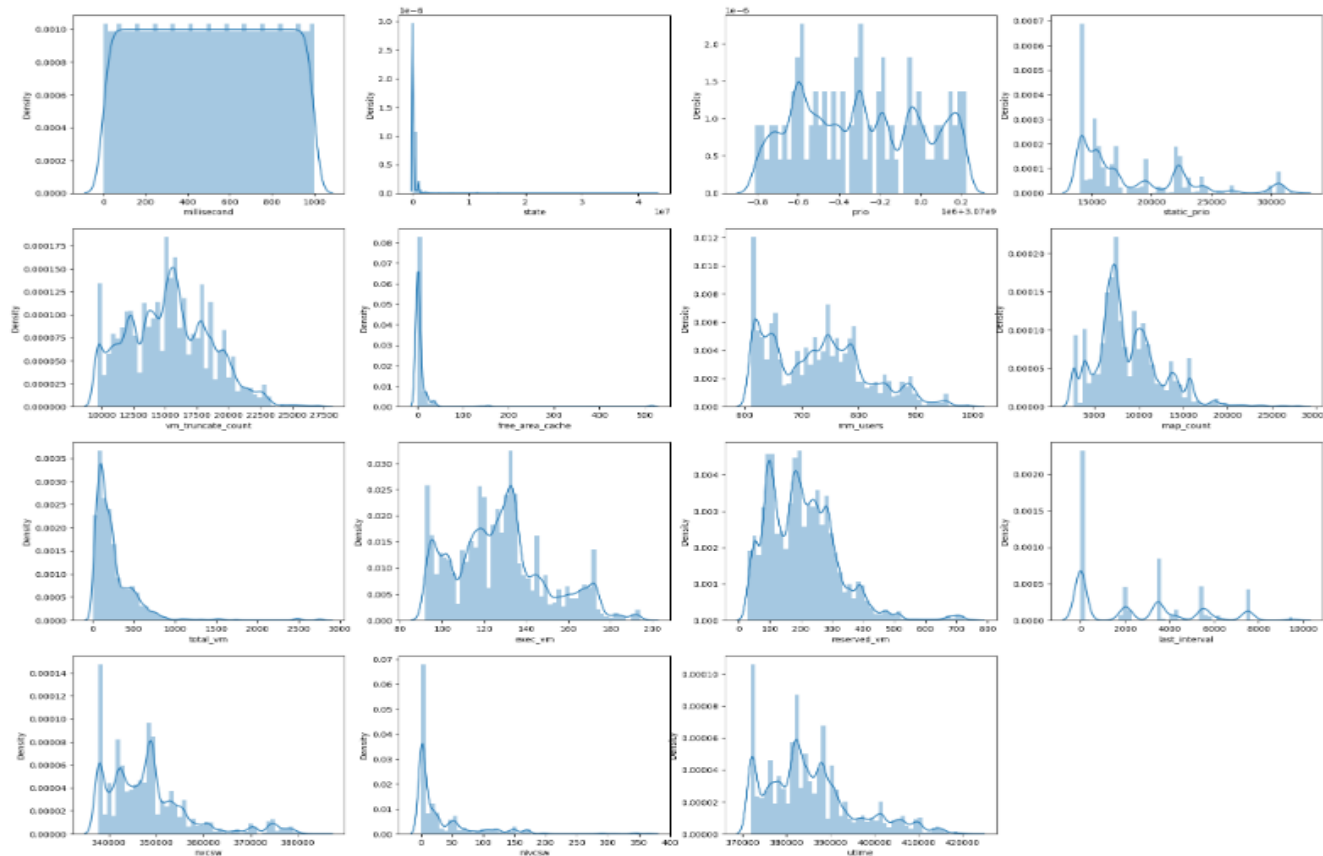


Figure 1. Feature Density Grid

Table 2 below depicts the feature observation from the dataset.

Table 2. Summary of Feature Distributions from Density Plots

Feature	Range	Distribution Characteristics	Observations
millisecond	(e.g., 0–1100)	Uniform distribution	High frequency across the range with no clear peaks.
state	(e.g., ~0–3e7)	Right skewed	Most values concentrated near 0.
pro	(e.g., -4e-5–2e-5)	Multi-modal	Several distinct peaks in the distribution.
static_pro	(e.g., ~0–35000)	Skewed	Higher density at lower values, with some distinct peaks.
syn_truncate_count	(e.g., ~10000–27000)	Multi-modal	Multiple peaks observed in the range.
tree_area_cache	(e.g., ~0–100)	Right-skewed	Majority of values near 0.
ram_users	(e.g., ~600–800)	Multi-modal	Concentration in certain ranges, possibly groups/clusters.
map_count	(e.g., ~2000–30000)	Skewed	Higher density near lower values.
total_vm	(e.g., ~0–3000)	Right skewed	Most values near 0, with a rapid drop-off.
sec_vm	(e.g., ~120–380)	Multi-modal	Clear clusters or repeated values in the range.
mserver_vm	(e.g., ~0–800)	Skewed	Most density at lower values.
last_interval	(e.g., ~0–10000)	Right skewed	Most values concentrated near 0.
rcrw	(e.g., ~340000–360000)	Multi-modal	Noticeable peaks, indicating recurring values.
ltime	(e.g., ~0–42000)	Right skewed	Concentration of values in a smaller range.

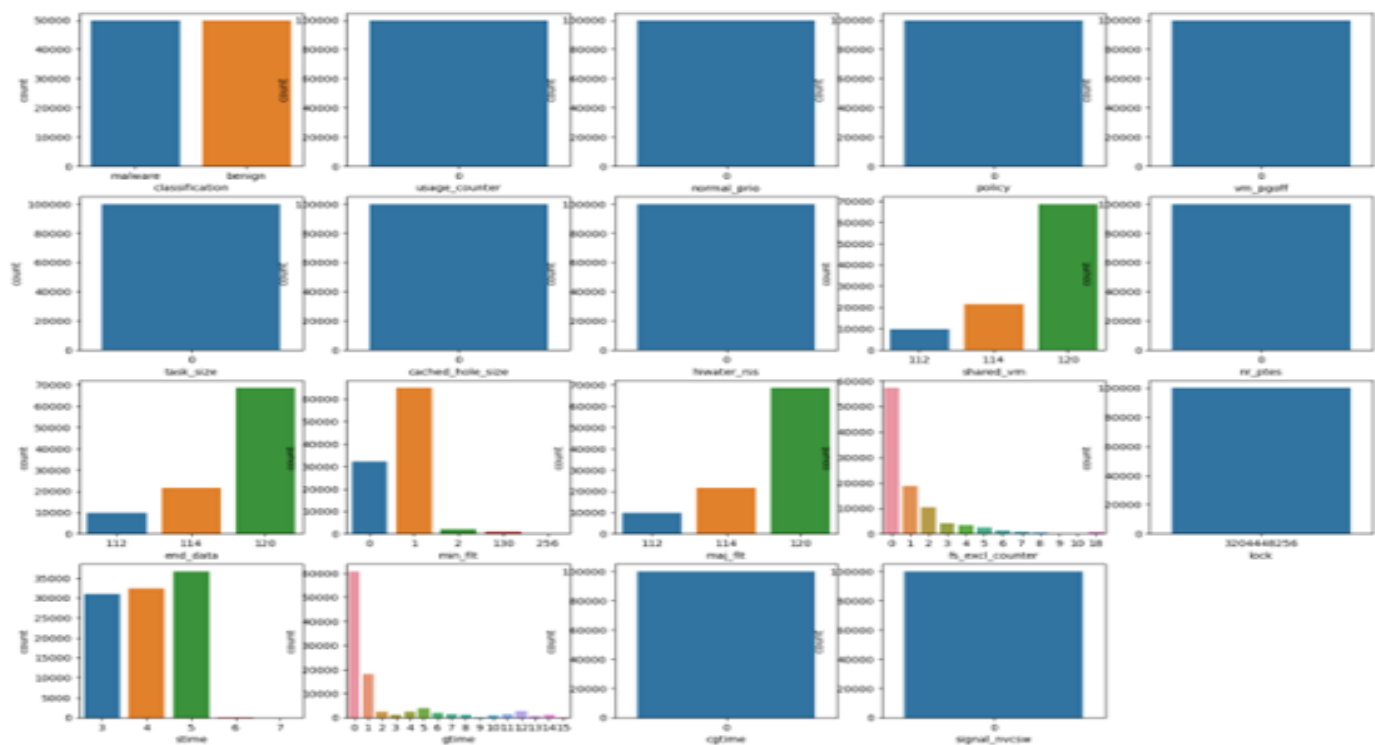


Figure 2. Feature Count Grid

This image presented in figure 2 shows a grid of bar charts, which could be the distribution of some categorical variable or the count of some feature in data set. Table 3 below list out the features and their value distribution.

Table 3. Histogram Feature Distributions

Feature Name	Distinct Values	Comments
classification	2 (malware, benign)	Balanced distribution
usage_counter	1	Single value
normal_prio	1	Single value
policy	1	Single value
vm_pgoff	1	Single value
task_size	3 (112, 114, 120)	Skewed towards higher values
cached_hole_size	Multiple (0-256)	Most data concentrated at lower values
hwater_rss	3 (112, 114, 120)	Similar pattern to task_size
shared_vm	Multiple (0-10)	Highly skewed; most at lower range
nr_ptes	1	Single value
lock	1 (3,046,412,536)	Single value
end_data	3 (112, 114, 120)	Similar distribution to task_size
mm_rss	Multiple (0-10+)	Highly skewed distribution
ma_rss	3 (112, 114, 120)	Similar pattern to task_size
gtime	3 (4, 6, 7)	Balanced
ctime	1	Single value
signal_nvcsw	1	Single value

Figure 3 displays the training and validation accuracy of a machine learning model over multiple epochs. The x-axis represents the number of epochs (training iterations), while the y-axis represents accuracy. Here's an analysis:

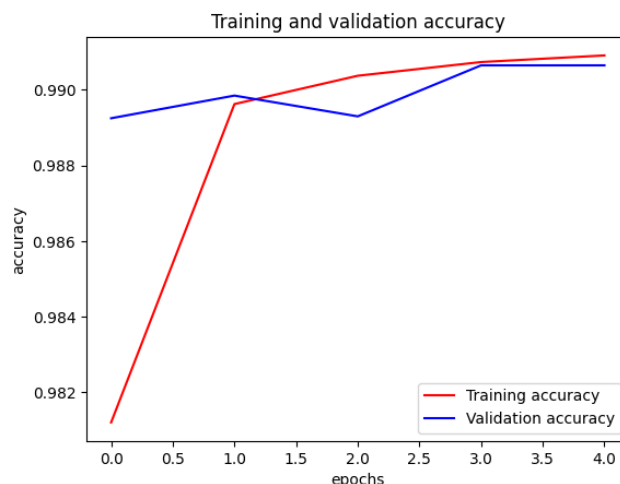


Figure 3 Training and Validation Accuracy Over Epochs

The training accuracy for the model is shown by the red line of accuracy trends and as evidenced by the curve it starts low but increases progressively in the first epoch and later stabilizes at very high training accuracy of 0.99+ as indicated above. The blue line represents the validation accuracy, and just as with the training accuracy, the validation accuracy is above the initial 50% and increases at a slower rate reaching a steady state that is almost the same as the training accuracy in a few epochs. After about 3-4 epochs, both training and validation accuracy are consistent suggesting that the model has probably identified the underlying patterns within the data. Most importantly, the training accuracy is almost paralleled by validation accuracy without any steep drops or gaps between these two which imply that the network is staying generalized to unseen data and there is no overfitting in the present model. It is clear that the accuracy for both training and validation set are high and relatively stable, this suggests that the model is well fits the training data and it is also generalized well to the validation set. This phenomenon indicates that the model architecture, the applied regularization techniques, as well as the quality of the dataset that has been used in this work are suitable for this task.

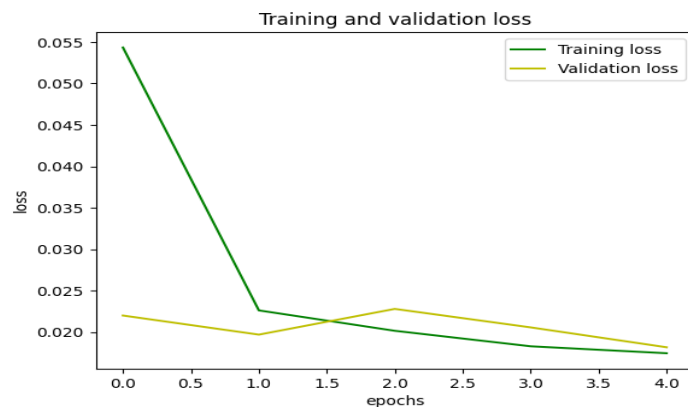


Figure 4 Training and Validation Loss Over Epochs

The plot in figure 4 denotes the training and validation loss in epochs which is a good measure of how a machine learning model is performing while training.

From the plotted figure, one can observe that the training loss is represented by the green color which clearly show that after the initial epoch of training it has consistently decreased gradually epoch after epoch, which clearly depicts that the training of the model has been good, and it is fitting well for the training data. The first plot represents the solid line which shows the validation loss, the yellow line gives fluctuations through slight increase and decrease before the later epochs of sharp decrease. This indicates some inconsistency in the performance of the model in the early episodes but there is a positive trend of improving generalization capability of the model. Insights from the loss curves reveal that the initial sharp decrease in training loss indicates the model learns quickly at the start. The validation loss being lower than the training loss in the early epochs could point to some regularization effect, such as dropout, that helps reduce overfitting early on. As the training progresses, both the training and validation losses converge, suggesting the model is generalizing well to unseen data. Overall, the model performs well with no significant signs of overfitting or underfitting. If the loss values plateau in subsequent epochs without large divergence, the training can be considered successful.

CONCLUSION

The proposed deep neural network-based framework for malware detection and classification demonstrates significant effectiveness and efficiency. The dataset used for training shows a balanced distribution between malware and benign classes, ensuring unbiased model training and evaluation. The feature distribution analysis focuses depicts the dispersion of several features where several of them are skewed and contain outliers which may impact recognition capability; these should be preprocessed. While some of the features are fairly consistent or grossly skewed, others provide good discriminant characteristics, as is evident from the histograms. Features could be removed or contained very little or irrelevant information which could be optimized to minimize the amount of computation done without any effects on its performance. In the training process, maximum accuracy values of model training and validation set are identified to be almost 99% after several epochs. This means that the session has been able to find the fundamental relations and features of the data input. The strong correspondence of validation accuracy to the training accuracy also indicates that the overall generalization ability of the model is excellent and does not suffer from overfitting. Moreover, the training loss keeps on reducing with epochs while the validation loss slightly oscillates and converges with the training loss which supports our proposition of good generalization capability of the model over the unseen data.

In total, the presented experimental results confirm the applicability of the proposed framework in terms of scalability, execution time, and safety for real-world cybersecurity scenarios. Here, efficient resource utilization and high detection rate make this approach a realistic and viable solution for addressing complicated and undiscovered malware. For example, future studies might build upon the various methods discussed in current literature to refine, feature selection, and feature optimization procedures to achieve better computational performance and flexibility.

FUTURE WORK

For future work, the integration of Convolutional Neural Networks (CNNs) with Generative Adversarial Networks (GANs) can be further explored to enhance malware detection systems. CNNs can extract intricate spatial features from malware representations, while GANs can generate synthetic datasets to address imbalances and expand the diversity of training data. Future studies may focus on optimizing the GAN-generated datasets to ensure the inclusion of realistic and representative malware variations, improving the generalization of CNN models. Additionally, combining GANs with advanced optimization techniques, such as reinforcement learning or meta-learning, could further enhance detection accuracy and reduce computational overhead. Research can also address the challenges of adversarial attacks and computational efficiency, ensuring the deployment of CNN-GAN architectures in real-time, resource-constrained environments.



REFERENCES

- [1] A. AliAhmad, D. Eleyan, A. Eleyan, T. Bejaoui, M. F. Zolkipli and M. Al-Khalidi, "Malware Detection Issues, Future Trends and Challenges: A Survey," 2023 International Symposium on Networks, Computers and Communications (ISNCC), Doha, Qatar, 2023, pp. 1-6, doi: 10.1109/ISNCC58260.2023.10323624.
- [2] S. G. Nayak, S. Kurup and A. J, "Malware Detection Employing Deep Neural Networks," 2024 10th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 2024, pp. 44-49, doi: 10.1109/ICACCS60874.2024.10717146.
- [3] A. A. Mustafa Majid, A. J. Alshaibi, E. Kostyuchenko, and A. Shelupanov, "A review of artificial intelligence-based malware detection using deep learning," Mater Today Proc, vol. 80, pp. 2678–2683, Jan. 2023, doi: 10.1016/J.MATPR.2021.07.012.
- [4] R. Pascanu, J. W. Stokes, H. Sanossian, M. Marinescu and A. Thomas, "Malware classification with recurrent networks," 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), South Brisbane, QLD, Australia, 2015, pp. 1916-1920, doi: 10.1109/ICASSP.2015.7178304.
- [5] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran and S. Venkatraman, "Robust Intelligent Malware Detection Using Deep Learning," in IEEE Access, vol. 7, pp. 46717-46738, 2019, doi: 10.1109/ACCESS.2019.2906934.
- [6] Dang, Quang-Vinh. (2022). Enhancing Obfuscated Malware Detection with Machine Learning Techniques. 10.1007/978-981-19-8069-5_54.

- [7] A. Razgallah, R. Khoury, K. Khanmohammadi and C. Pere, "Comparing the Effectiveness of Static, Dynamic and Hybrid Malware Detection on a Common Dataset," 2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Honolulu, Oahu, HI, USA, 2023, pp. 2452-2458, doi: 10.1109/SMC53992.2023.10394198.
- [8] J. Zhao, S. Zhang, B. Liu and B. Cui, "Malware Detection Using Machine Learning Based on the Combination of Dynamic and Static Features," 2018 27th International Conference on Computer Communication and Networks (ICCCN), Hangzhou, China, 2018, pp. 1-6, doi: 10.1109/ICCCN.2018.8487459.
- [9] M. F. Ansari, B. Dash, P. Sharma, and N. Yathiraju, "The Impact and Limitations of Artificial Intelligence in Cybersecurity: A Literature Review," IJARCCCE, vol. 11, no. 9, Sep. 2022, doi: 10.17148/ijarccce.2022.11912.
- [10] Y. Ye, T. Li, D. Adjeroh, and S. S. Iyengar, "A Survey on Malware Detection Using Data Mining Techniques," ACM Computing Surveys, vol. 50, no. 3, pp. 1–40, Jun. 2017, doi: <https://doi.org/10.1145/3073559>.
- [11] G. Rjoub et al., "A Survey on Explainable Artificial Intelligence for Cybersecurity," Mar. 2023, doi: 10.1109/TNSM.2023.3282740.
- [12] F. Ullah, A. Alsirhani, M. M. Alshahrani, A. Alomari, H. Naeem, and S. A. Shah, "Explainable Malware Detection System Using Transformers-Based Transfer Learning and Multi-Model Visual Representation," Sensors, vol. 22, no. 18, Sep. 2022, doi: 10.3390/s22186766.
- [13] D. Alvarez-Melis and T. S. Jaakkola, "Towards robust interpretability with self-explaining neural networks," arXiv (Cornell University), vol. 31, pp. 7786–7795, Dec. 2018, [Online]. Available: <http://arxiv.org/pdf/1806.07538.pdf>
- [14] Y. -H. Chen, S. -C. Lin, S. -C. Huang, C. -L. Lei and C. -Y. Huang, "Guided Malware Sample Analysis Based on Graph Neural Networks," in IEEE Transactions on Information Forensics and Security, vol. 18, pp. 4128-4143, 2023, doi: 10.1109/TIFS.2023.3283913.
- [15] Barredo Arrieta et al., "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI," Information Fusion, vol. 58, pp. 82–115, Jun. 2020, doi: 10.1016/j.inffus.2019.12.012.
- [16] A. Das and P. Rad, "Opportunities and Challenges in Explainable Artificial Intelligence (XAI): A Survey," Jun. 2020, [Online]. Available: <http://arxiv.org/abs/2006.11371>
- [17] P. J. Phillips et al., "Four principles of explainable artificial intelligence," Sep. 2021. doi: 10.6028/NIST.IR.8312.
- [18] Y. Liu, C. Tantithamthavorn, L. Li, and Y. Liu, "Explainable AI for Android Malware Detection: Towards Understanding Why the Models Perform So Well?" Sep. 2022, [online]. Available: <http://arxiv.org/abs/2209.00812>
- [19] E. Venkata Pawan Kalyan, A. Purushottam Adarsh, S. Sai Likith Reddy and P. Renjith, "Detection of Malware Using CNN," 2022 Second International Conference on Computer Science, Engineering and Applications (ICCSEA), Gunupur, India, 2022, pp. 1-6, doi: 10.1109/ICCSEA54677.2022.9936225.
- [20] M. Ganesh, P. Pednekar, P. Prabhuswamy, D. S. Nair, Y. Park and H. Jeon, "CNN-Based Android Malware Detection," 2017 International Conference on Software Security and Assurance (ICSSA), Altoona, PA, USA, 2017, pp. 60-65, doi: 10.1109/ICSSA.2017.18.
- [21] Muhammad Saqib. (2025). Optimizing Spot Instance Reliability and Security Using Cloud-Native Data and Tools. In Journal of Information Systems Engineering and Management (Vol. 10, Issue 14s, pp. 720–731). Science Research Society. <https://doi.org/10.52783/jisem.v10i14s.2387>
- [22] M. U. Majid, M. A. Raza, and N. Khan, "Malware Detection Using Convolutional Neural Networks (CNN) and Image-Based Analysis," 2023 International Conference on Artificial Intelligence and Machine Learning (ICAIML), Karachi, Pakistan, 2023, pp. 123-128, doi: 10.1109/ICAIML57134.2023.10020014.
- [23] Tayyab, U.-e.-H.; Khan, F.B.; Durad, M.H.; Khan, A.; Lee, Y.S. A Survey of the Recent Trends in Deep Learning Based Malware Detection. J. Cybersecur. Priv. 2022, 2, 800-829. <https://doi.org/10.3390/jcp2040041>
- [24] R. Jones, M. Omar, D. Mohammed, C. Nobles and M. Dawson, "Harnessing the Speed and Accuracy of Machine Learning to Advance Cybersecurity," 2023 Congress in Computer Science, Computer Engineering, & Applied Computing (CSCE), Las Vegas, NV, USA, 2023, pp. 418-421, doi: 10.1109/CSCE60160.2023.00074.

-
- [25] F. Ullah, A. Alsirhani, M. M. Alshahrani, A. Alomari, H. Naeem, and S. A. Shah, "Explainable Malware Detection System Using Transformers-Based Transfer Learning and Multi-Model Visual Representation," *Sensors*, vol. 22, no. 18, Sep. 2022, doi: 10.3390/s22186766.
 - [26] Y. Zhang and Y. Gu, "A Survey of Traditional and Machine Learning-based Malware Detection Techniques," 2023 IEEE 3rd International Conference on Information Technology, Big Data and Artificial Intelligence (ICIBA), Chongqing, China, 2023, pp. 1373-1378, doi: 10.1109/ICIBA56860.2023.10165632.
 - [27] A. AliAhmad, D. Eleyan, A. Eleyan, T. Bejaoui, M. F. Zolkipli and M. Al-Khalidi, "Malware Detection Issues, Future Trends and Challenges: A Survey," 2023 International Symposium on Networks, Computers and Communications (ISNCC), Doha, Qatar, 2023, pp. 1-6, doi: 10.1109/ISNCC58260.2023.10323624.
 - [28] N. Z. Gorment, A. Selamat, L. K. Cheng and O. Krejcar, "Machine Learning Algorithm for Malware Detection: Taxonomy, Current Challenges, and Future Directions," in *IEEE Access*, vol. 11, pp. 141045-141089, 2023, doi: 10.1109/ACCESS.2023.3256979.
 - [29] "What is Explainable AI (XAI)? | IBM." <https://www.ibm.com/watson/explainable-ai> (accessed Jul. 19, 2023).
 - [30] P. J. Phillips et al., "Four principles of explainable artificial intelligence," Sep. 2021. doi: 10.6028/NIST.IR.8312.
 - [31] I. Azhar and M. Sr, "Novateur Publications International Journal of Innovations in Engineering Research and Technology [ijert] Artificial Intelligence for Cybersecurity: a Systematic Mapping of Literature," Website: ijert.org VOLUME, vol. 7, 2020.
 - [32] S. M. Mathews, "Explainable Artificial Intelligence Applications in NLP, Biomedical, and Malware Classification: A Literature Review," in *Intelligent Computing - Proceedings of the Computing Conference*, 2019, pp. 1269–1292. doi: 10.1007/978-3-030-22868-2_90.
 - [33] M. J. Hossain Faruk et al., "Malware Detection and Prevention using Artificial Intelligence Techniques," in *Proceedings - 2021 IEEE International Conference on Big Data, Big Data 2021*, Institute of Electrical and Electronics Engineers Inc., 2021, pp. 5369–5377. doi: 10.1109/BigData52589.2021.9671434.
 - [34] H. Patel, "The Future of Cybersecurity with Artificial Intelligence (AI) and Machine Learning (ML)," 2023, doi: 10.20944/preprints202301.0115.v1.
 - [35] LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, doi: <https://doi.org/10.1038/nature14539>.
 - [36] D. Ucci, L. Aniello, and R. Baldoni, "Survey of machine learning techniques for malware analysis," *Computers & Security*, vol. 81, pp. 123–147, Mar. 2019, doi: <https://doi.org/10.1016/j.cose.2018.11.001>.
 - [37] L. Sasikala and C. Shanmuganathan, "Effective Malware Classification using Fine-tuned CNN Architecture: An Image-Based Approach," 2024 2nd International Conference on Networking and Communications (ICNWC), Chennai, India, 2024, pp. 1-6, doi: 10.1109/ICNWC60771.2024.10537444.
 - [38] Y. -s. Liu, Y. -K. Lai, Z. -H. Wang and H. -B. Yan, "A New Learning Approach to Malware Classification Using Discriminative Feature Extraction," in *IEEE Access*, vol. 7, pp. 13015-13023, 2019, doi: 10.1109/ACCESS.2019.2892500.
 - [39] S. Venkatraman, M. Alazab, and R. Vinayakumar, "A hybrid deep learning image-based analysis for effective malware detection," *Journal of Information Security and Applications*, vol. 47, pp. 377– 389, Jun. 2019, doi: 10.1016/j.jisa.2019.06.006.
 - [40] H. B. Nguyen, B. Xue, P. Andrae and M. Zhang, "Particle Swarm Optimisation with genetic operators for feature selection," 2017 IEEE Congress on Evolutionary Computation (CEC), Donostia, Spain, 2017, pp. 286-293, doi: 10.1109/CEC.2017.7969325.

BIOGRAPHY OF AUTHORS

	<p>Mrs. Deepika Sharma is an Assistant Professor at DET, BVDU specializing in the application of Artificial Intelligence and Machine Learning to real-world problems. Her background combines academic rigor with practical industry experience. She holds a Master's degree in Information Security (10 CGPA). She is currently pursuing her Ph.D (IT) from Amity Institute of Information Technology, focusing her research on malware detection and their applications in cybersecurity under the guidance of Dr. Manoj H Devare. Her research has been published as "Assessing Autonomic Level for Self-managed Systems – FAHP Based Approach" (Advances in Computing and Data Sciences. ICACDS 2018. CCIS, vol 905. Springer, Singapore). Prior to entering academia, Mrs. Sharma worked as an Assistant System Engineer at Tata Consultancy Services, developing automation solutions using Python and shell scripting, providing her with valuable practical insights that inform her teaching and research.</p>
	<p>Dr. Manoj Devare is a Professor of Computer Science at Amity University Mumbai, India. He received his BSc and MSc from North Maharashtra University and his PhD from Bharati Vidyapeeth University, all in Computer Science. His doctoral research focused on performance optimization and resource allocation in homogeneous and heterogeneous computer networks, earning him a "Best Paper Award" at the ICSCI 2008 international conference. Following his PhD, Dr. Devare held a Postdoctoral Fellowship at the Centre of Excellence on HPC at the University of Calabria, Italy, where he was recognized as a Young Researcher (Giovani Ricercatori Indiani). His postdoctoral work centered on multi-scale high-performance computing in grids, virtualization, and clouds, with applications to scientific problems. Notably, he originated the concept of "Desktop Clouds."</p> <p>Dr. Devare's research contributions include two granted Indian patents, nine international journal publications, six edited book chapters, and 15 conference papers. His current research interests encompass grid computing and GPU programming using Nvidia-Tesla architectures. He has also collaborated with Nimbus (Chicago) Science Clouds.</p>