Harnessing Machine Learning for Enhanced Malware Detection and Analysis

Andrés Pérez, Master Student
Polytechnic University of Puerto Rico
andres6perez@gmail.com

Rafael González, Master Student
Polytechnic University of Puerto Rico
regc1024@gmail.com

Roberto Vivas, Master Student
Polytechnic University of Puerto Rico
robertovivas2@gmail.com

Dr. Alfredo Cruz, Ph.D.
Polytechnic University of Puerto Rico
alcruz@pupr.edu

Abstract

Malware attacks have become increasingly sophisticated and widespread, posing a significant threat to global cybersecurity. To address this challenge, effective malware detection methods are crucial for identifying and mitigating potential threats. This research uses two machine learning models, Convolutional Neural Networks (CNN) and Random Forest (RF), to analyze and detect malware. The study aims to improve malware detection accuracy, minimize false positives, and enhance model efficiency. We implemented both models using a publicly available malware dataset and conducted a comparative analysis to evaluate their performance. CNN was leveraged for its feature extraction and pattern recognition strength, while RF was selected for its robustness and interpretability in decision-making. The results demonstrated that CNN achieved high precision in identifying complex patterns, while RF excelled in balancing accuracy and computational efficiency. This analysis provides valuable insights into the strengths and limitations of these models, contributing to the development of more effective malware detection systems. All experiments and analyses were conducted using Python and relevant machine-learning libraries in a controlled computational environment.

Keywords: Machine Learning, Malware Analysis, Cybersecurity, Convolutional Neural Networks, Random Forest, Supervised Learning, Threat Detection, Feature Extraction, Comparative Study, Python

Introduction

Malware poses a persistent and evolving threat to cybersecurity, rendering traditional detection methods such as signature-based and heuristic analysis insufficient against sophisticated and zero-day attacks (Smith et al., 2022). The increasing complexity of malware necessitates adaptive detection techniques capable of real-time threat identification. Machine learning (ML) offers a dynamic approach by analyzing large datasets and detecting patterns indicative of malicious activity (Brown et al., 2023).

This study investigates the effectiveness of Convolutional Neural Networks (CNNs) and Random Forest (RF) models in malware detection. CNNs, known for their pattern recognition capabilities, are evaluated against RF's interpretability and classification efficiency (Green et al., 2023). Using a publicly available malware dataset, we assess these models based on accuracy, precision, and recall to determine their

strengths, limitations, and potential synergies. The findings contribute to advancing ML-driven cybersecurity solutions and optimizing threat detection and mitigation strategies.

Evolution of Malware Detection: From Traditional Techniques to Machine Learning Models

Traditional malware detection relies on signature-based and heuristic methods, which compare files against known malware signatures or analyze suspicious behaviors. While effective against previously identified threats, these approaches struggle with zero-day attacks and polymorphic malware (Anderson, 2022).

Machine learning (ML) addresses these limitations by dynamically identifying known and novel malware variants. Supervised learning, particularly Convolutional Neural Networks (CNNs) and Random Forests (RFs), has shown significant promise in malware classification. CNNs, traditionally used in image processing, analyze malware binaries as visual representations, detecting complex patterns. RF, an ensemble-based model, offers robust classification with high interpretability (Djenna et al., 2023).

The integration of ML in malware detection has enhanced threat identification by reducing reliance on manual updates and adapting to evolving attack strategies. This shift represents a critical advancement in cybersecurity, improving accuracy, efficiency, and adaptability in combating modern threats.

Case Studies, Comparative Analysis, and Challenges

Real-world applications highlight the effectiveness of machine learning in malware detection. The Microsoft Malware Classification Challenge demonstrated the strengths of Convolutional Neural Networks (CNNs) in pattern recognition and Random Forests (RFs) in interpretability. Cylance's AI-based approach further showcased ML's ability to detect malware without signatures by analyzing behavioral patterns (Blackberry Cylance, 2019). At the same time, Google's VirusTotal integrates ML for real-time analysis of suspicious files and URLs, addressing threats like polymorphic malware (VirusTotal, 2025).

Comparative studies reinforce these findings. The 2018 Tabular EMBER Dataset, widely used in malware research, has been the basis of numerous evaluations comparing deep learning and traditional machine learning approaches. One notable study by Gale and Steele (2020) was selected due to its rigorous methodology and clear contrast between model performances. It was identified by reviewing works that directly benchmarked traditional models against neural networks using standardized datasets. Their analysis revealed that a Random Forest (RF) classifier achieved an accuracy of 92.03% and an AUC of 0.979, significantly outperforming a neural network's 62.29% accuracy and 0.615 AUC. Using a randomly selected subset of 50,000 entries from the EMBER dataset and default hyperparameters from the Scikit-learn library, their study highlighted that deep learning does not inherently offer superior results-emphasizing the critical role of appropriate model selection in malware detection.

Table 1 Comparative Analysis of Previous Machine Learning Techniques

Study	Model	Precision	Recall	AUC	Accuracy
(Gale & Steele, 2020)	Random Forest	0.93	0.92	0.98	0.92
	Neural Network	0.75	0.69	0.62	0.62

Research Methodology

This research analyzes genuine and malicious activities using Random Forest (RF) and Convolutional Neural Networks (CNNs) based on their malware detection effectiveness using precision, recall, F1 score,

ROC AUC, and accuracy evaluation metrics. Random Forest was selected due to its strong performance in past research, while CNN was chosen after a standard Neural Network (NN) was initially tested but performed poorly on the EMBER 2018 dataset. With that being said, the evaluation metrics: (1) Precision, which is the percentage of the correct positive predictions out of all the positive predictions made by a given model; (2) Recall, the percentage of true positives that the model correctly found out of all the actual positives; the (3) F1 Score, which is a combination of the precision and recall values into a single number, is a harmonic mean of both results that is useful when the false positives and negatives of a given model must be considered equally; the (4) ROC - AUC is a graphical metric used to evaluate how models distinguish between positive and negative classes, like, in this case, whether a file contains malware or not; and (5) Accuracy, which is the overall percentage of correct predictions made by a given model, these were to present a more concrete and accurate picture on the models performances. The alternative, which is using Accuracy, does not consider factors of the models, like the false positives and negatives.

Figure 1 outlines the complete workflow for training and evaluating machine learning models, focusing on 2D CNN and Random Forest. The process begins with dataset preprocessing, followed by data splitting (80% training, 20% testing). The training phase is divided into two parallel paths: training the 2D CNN and Random Forest models, each producing a trained model. These models are then tested, and their results undergo evaluation to determine malware detection accuracy.

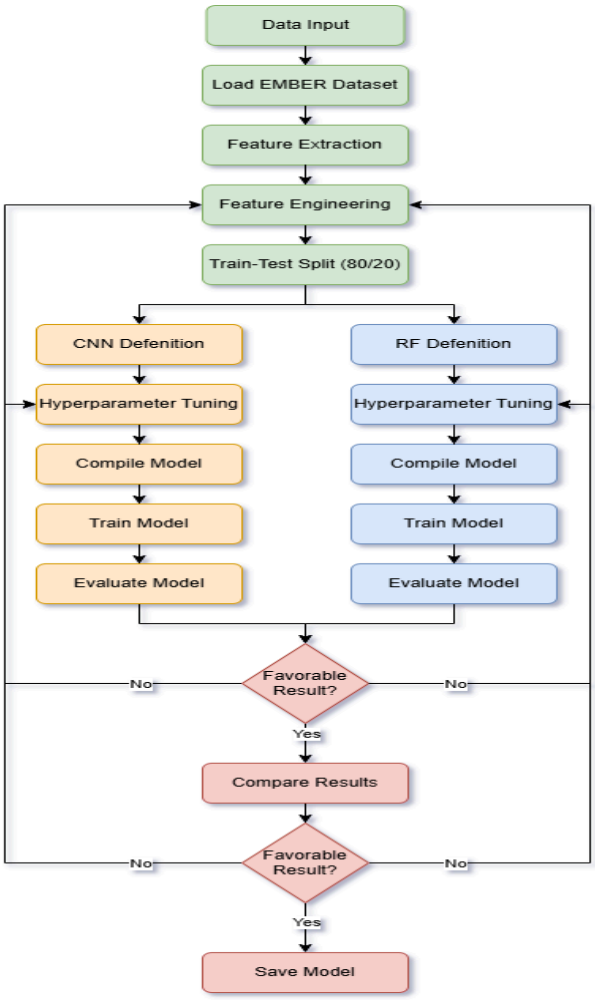


Figure 1 ML Workflow for Malware Detection using CNN (2D) and Random Forest models

Dataset and Feature Analysis

The EMBER 2018 dataset was selected for this study due to its widespread adoption in academic and industry malware research, its open availability, and its balanced composition of benign and malicious Windows Portable Executable (PE) files. Unlike proprietary or less-documented alternatives such as VirusShare or Malicia, EMBER offers well-structured, labeled data and a comprehensive set of pre-extracted features—making it ideal for benchmarking machine learning models in malware detection. Each sample in the dataset is described by several core features that capture the structural and statistical properties of PE files, including Histogram, ByteEntropy, Sections, Imports, Exports, Header, General, and Strings. As outlined by Anderson and Roth (2018), these features are divided into two categories: parsed and format-agnostic. Parsed features include: (1) General—file size and basic PE header data, such as virtual size, number of imported/exported functions, and presence of a debug section; (2) Header—COFF header fields like timestamp, target machine, and DLL characteristics; (3) Imports—function information per library; (4) Exports—function information similarly organized; and (5) Sections—details on each section's name, size, entropy, virtual size, and section flags. Format-agnostic features are: (1) Histogram—256-byte frequency counts; (2) ByteEntropy—entropy estimates via joint distribution p(H, X) where H is entropy and X is a byte value; and (3) Strings—statistics on printable strings between 0x20 and 0x7F of at least five characters in length, including counts and average lengths. Figure 2 illustrates histograms of selected features such as Entropy, Numstrings, AVLength, and Printables. The full dataset contains 1,200,000 entries, typically partitioned into 1,000,000 for training and 200,000 for testing, enabling scale and reproducibility in model evaluation.

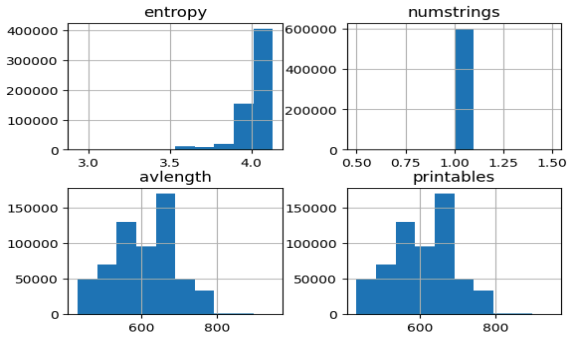


Figure 2 Histograms of features (entropy, numstrings, avlength, printables), showing their distributions.

EMBER 2018 Dataset Data Processing, Feature Extraction, and Feature Selection

The dataset was split into 80% training and 20% testing, following best practices (Gholamy et al., 2018) to handle overfitting problems. The training data was used to learn model parameters, while the testing data assessed accuracy and robustness. Studies suggest that this data division achieves optimal performance by balancing model learning and evaluation reliability.

Due to the characteristics of the EMBER dataset and the nature of the Random Forest (RF) model, specific preprocessing guidelines and feature extraction recommendations were followed to enhance model performance and maintain consistency. In particular, the work of Saxe and Berlin (2015) emphasized the value of combining parsed and format-agnostic features for static malware detection, while Roth (2022) provided implementation guidance for deriving high-dimensional feature sets from PE files. Additionally, Akhtar and Feng (2023) highlighted practical considerations for scaling large datasets without compromising feature integrity. These informed the approach used in this study, which expanded the original EMBER dataset from its initial two types of features to 2,350 features—representing a 29,275% increase. This expansion significantly increased computational demands, which led to a decision not to apply further feature selection for the RF model due to a limitation in Random Access Memory (RAM). A subset of these expanded features is shown in the figure below.

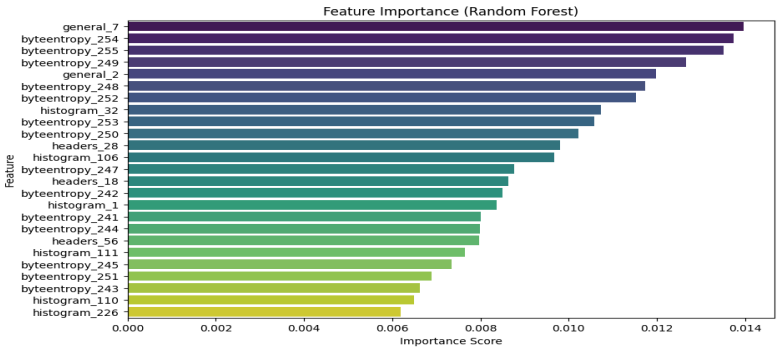


Figure 3 Feature Importances of Expanded EMBER 2018 Features for Random Forest

For the Convolutional Neural Network (CNN) model, feature selection was manually defined by selecting a fixed subset of features: histogram, byteentropy, numstrings, avlength, size, and vsize. The histogram and byteentropy features were stacked and reshaped into 32×16 grayscale grids to serve as 2D input for the CNN. Feature values were normalized to the [0,1] range. No automated feature selection methods were applied, as the CNN architecture extracts relevant patterns through its convolutional layers during training.

Model Selection and Training

The Random Forest model was selected for its efficiency in handling high-dimensional data and its ability to aggregate multiple decision trees for robust predictions (Breiman, 2001). Following the data preprocessing and feature extraction practices of Mirza (2009), Raman (2012), Saxe and Berlin (2015), Gale and Steele (2020), Roth (2022), and Akhtar and Feng (2023), like the conversion of the ByteEntropy and Histogram feature histograms to numpy arrays of 256 elements, representing 256 bits, and the acquisition of the size, virtual size and names of the sections in the PE files, to name a few of the things that were done, on the EMBER 2018 Dataset, as well as the preliminary results of the model itself, it was concluded that feature selection was not needed, yet recommended. With that being said, with the nominal hyperparameters of RF and the processed dataset, when evaluating RF at testing, it was possible to acquire a ROC - AUC score of 0.99, prompting for overfitting analysis, as discussed in the next section.

The Convolutional Neural Network (CNN) model was selected for its ability to extract spatial and hierarchical patterns from structured data, making it well-suited for malware detection (LeCun et al., 2015). The convolution operation, a fundamental component of CNNs, enables the model to identify complex structures by applying a sliding kernel over the input feature map. This operation computes a weighted sum of local regions, capturing essential patterns such as edges and textures that distinguish malware characteristics. The mathematical formulation of the convolution process is given by:

The convolution (cross-correlation) operation equation is given by:

$$S(i,j) = \sum_m \sum_n I\,(i + m,\ j + n) \cdot K(m,n) \tag{1}$$

where I represents the input feature map, K is the convolutional kernel, and (m,n) are indices iterating over the kernel dimensions. The resulting output feature map S(i,j) enhances critical representations necessary for classification. Backpropagation was employed to optimize learning, adjust network weights using gradient descent, ensure efficient model convergence, and improve malware detection accuracy.

Backpropagation optimizes the CNN by adjusting convolutional filter weights using gradient descent, ensuring efficient learning. The process computes the gradient of the loss function concerning each filter using the following equation.

The backpropagation for a convolution K equation is given by:

$$\frac{\partial L}{\partial K(m,n)} = \sum_{i,j} \delta(i,j) \cdot I(i + m, j + n) \quad (2)$$

where δ(i,j) represents the gradient of the loss concerning the convolution output, and I(i+m,j+n) is the input feature map. The summation aggregates contributions from all spatial positions where the filter was applied, allowing the model to update weights effectively. This process enables the backpropagation of errors by leveraging the chain rule refining convolutional filters to enhance malware detection accuracy.

Hyperparameter Tuning

The Random Forest model was tuned by adjusting n_estimators to 200, max_depth to 12, min_samples_leaf to 15, min_samples_split to 20, bootstrap to True, and max_samples to 0.70. A fixed random_state of 42 was used for reproducibility. Tuning was guided by the manual process of training, testing, and visualizing the RF model results with learning curves. Like with feature selection, automatized hyperparameter tuning procedures, like Randomized Search and Gradient Search, were not undertaken due to memory limitations.

For the CNN, the Adam optimizer was used with the default learning rate. Batch sizes of 32, 64, and 128 were tested, and training was capped at 50 epochs with early stopping (patience = 5) based on validation loss. Dropout was applied after dense layers with rates of 0.4, 0.3, and 0.2, and batch normalization was used after each convolution. The model used (3,3) kernels with filter depths of 32 and 64, and the best weights were saved using a validation loss checkpoint.

Model Evaluation

The Random Forest (RF) and 2D Convolutional Neural Network (CNN) models were evaluated using standardized performance metrics, including accuracy, precision, recall, F1 score, and ROC-AUC. These metrics provided a comprehensive view of each model's classification capabilities, covering the correctness of predictions and the balance between false positives and false negatives.

Visual evaluation methods were also incorporated to aid in model assessment. Confusion matrices were used to visualize true positives, true negatives, false positives, and false negatives, offering direct insight into misclassification patterns. Receiver Operating Characteristic (ROC) curves were generated to examine the trade-off between true positive rate and false positive rate across different decision thresholds, with the area under the curve (AUC) indicator of discriminative ability.

For CNN, additional evaluation was conducted on the influence of batch size and training epochs. Multiple configurations were tested to observe how changes in these hyperparameters affected training stability and generalization. Feature scaling and reshaping techniques were also evaluated to ensure compatibility with the CNN architecture, which processes inputs as 2D representations.

For RF, evaluation included model behavior under default settings and tuned hyperparameters. Visual tools such as linear learning curves were used to monitor for signs of overfitting. Preprocessing steps, including removing ambiguous labels and transforming input features, were assessed for their impact on model stability and performance consistency.

Experiment Results

This section presents the performance analysis of Random Forest (RF) and 2D Convolutional Neural Networks (CNNs) in malware detection using the EMBER dataset. The evaluation metrics comprehensively compare both models' effectiveness, including precision, recall, F1 score, ROC-AUC, and accuracy. Table 2 presents a detailed performance comparison of the Random Forest (RF) and Convolutional Neural Network (CNN) models across one million entries, broken down by class. The CNN

achieved consistently high performance across both classes, with precision, recall, and F1 scores all at or above 0.91. It reached 0.91 accuracy and a ROC-AUC of 0.97, demonstrating balanced and reliable classification for benign (class 0) and malware (class 1) samples.

Random Forest also performed well, with a precision of 0.92 for benign files and 0.88 for malware. Its recall showed the opposite trend—higher for malware (0.93) but lower for benign files (0.87). This imbalance suggests that while RF effectively detects threats, it may occasionally misclassify benign files. Still, its overall accuracy remained strong at 0.90, and like CNN, it achieved a ROC-AUC of 0.97, indicating robust class separation.

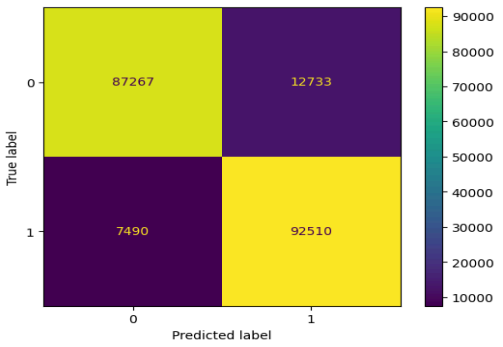
Although RF initially reached an AUC of 0.99 with default settings, further evaluation revealed this was due to overfitting the EMBER 2018 dataset. After removing ambiguous labels and applying tuning, the model's performance stabilized but slightly lagged behind CNN in consistency. CNN delivered uniform results across both classes, making it more dependable in high-stakes malware detection. Meanwhile, RF remains valuable for its interpretability and efficiency—especially when classifying structured data in less dynamic environments.

Table 2 Performance Analysis of Studied Models (1M Entries)

Model	Class	Precision	Recall	F1 Score	ROC - AUC	Accuracy
Random Forest	0	0.92	0.87	0.90	0.97	0.90
	1	0.88	0.93	0.90		
Convolutional Neural Network	0	0.92	0.91	0.91	0.97	0.91
	1	0.91	0.92	0.91		

Figure 3 presents the confusion matrix for the Random Forest model, showcasing its classification performance. The model correctly classified 87,267 instances as benign files (True Negatives) and 92,510 as malware (True Positives). However, it misclassified 12,733 instances as malware when they were benign (False Positives) and incorrectly identified 7,490 instances of malware as benign (False Negatives). These misclassifications highlight the trade-offs between precision and recall, showing that while the model performs well overall, there is still room for improvement in reducing false negatives.

Figure 4 displays the Receiver Operating Characteristic (ROC) curve, which illustrates the trade-off between the True Positive Rate (TPR) and False Positive Rate (FPR) at various decision thresholds. The area under the curve (AUC) of 0.97 confirms that the Random Forest model can differentiate between malware and non-malware samples. These results demonstrate the model's effectiveness in malware detection, providing a balance between accurate predictions and a manageable rate of false positives and negatives.



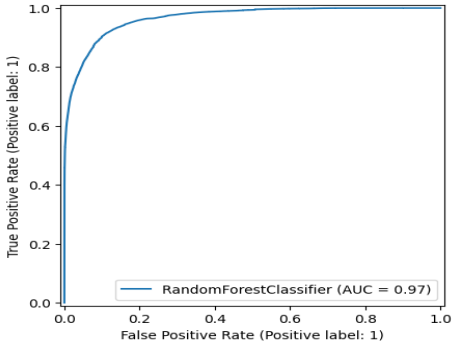


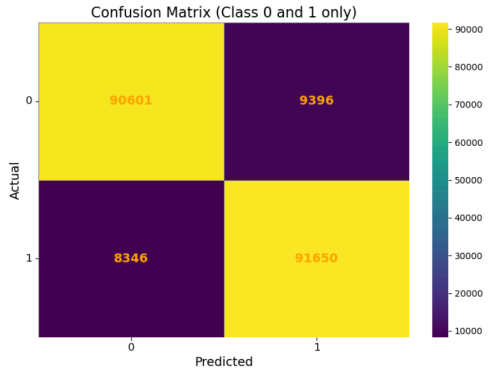
Figure 4 Confusion Matrix of Random Forest Results

Figure 5 ROC Curve of Random Forest Results

The 2D CNN was trained using batch sizes of 32, 64, and 128 over multiple epochs to evaluate how these parameters affect convergence and classification performance. Figure 5 shows the confusion matrix for classes 0 and 1, revealing strong predictive accuracy and class balance. The best-performing model reached a test accuracy of 91.13%, outperforming the Random Forest baseline. The CNN architecture included convolutional layers with 32 and 64 filters, followed by three fully connected layers with 256, 128, and 64 neurons, each with dropout rates of 0.4, 0.3, and 0.2, respectively. This configuration helped reduce overfitting and improved the model's generalization to unseen data.

A batch size of 64 offered the best trade-off between convergence speed and stability. Smaller batches introduced high variance during training, while larger ones slowed learning. Accuracy gains diminished after 30 epochs, suggesting the model had reached optimal convergence. Figure 6 presents the ROC curves for class 0 and class 1, both achieving an AUC of 0.9718. This indicates a high true positive rate and a low false positive rate across thresholds, reinforcing the model's reliability in distinguishing between classes. The near-identical AUC values for both classes also reflect CNN's balanced performance.

While the Random Forest model relied on structured, engineered features, the CNN extracted spatial and hierarchical representations from byte-level input. Although RF showed higher precision in some cases due to effective preprocessing, CNN demonstrated stronger adaptability to complex malware patterns, making it a powerful tool for large-scale detection tasks.



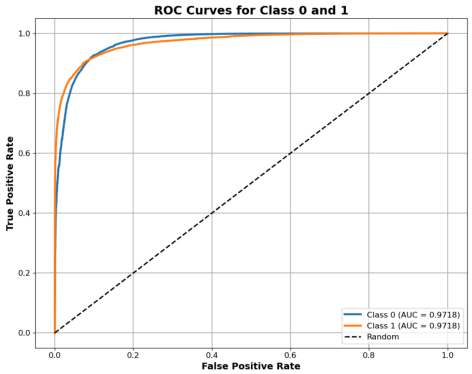


Figure 6 Confusion Matrix of CNN Results

Figure 7 ROC Curve of CNN Results

The study highlights the strengths and trade-offs of Convolutional Neural Networks and Random Forests in the context of malware detection. CNNs demonstrated superior capability in extracting deep patterns and handling unstructured or transformed data, making them well-suited for scenarios requiring sophisticated feature extraction. On the other hand, the Random Forest model excelled in handling structured data with interpretability and computational efficiency.

The modifications applied to the Random Forest model, including dataset preprocessing, removal of ambiguous cases (-1), advanced data transformations, and hyperparameter tuning, significantly improved its performance. Achieving an accuracy of 90%, the RF model exhibited strong predictive capabilities, with F1 Scores of 90% for benign files and 90% for malware files. However, the RF model demonstrated a bias toward True Negative cases, indicating better performance in predicting non-malware files, as evidenced by the confusion matrix, only after finely tuned for overfitting.

A hybrid approach leveraging the strengths of both methods offers a promising avenue for malware detection. CNNs could extract high-level, meaningful features from raw data, while Random Forests could be employed for final classification, balancing robustness, interpretability, and computational efficiency. This combination could enhance malware detection systems, ensuring adaptability to evolving threats and providing actionable insights for cybersecurity professionals.

Conclusions and Future Work

Future work can explore the development of a hybrid model that combines the strengths of Convolutional Neural Networks (CNNs) and Random Forests (RF), leveraging CNNs for high-level automated feature extraction and RF for efficient, interpretable classification. While CNNs excel in learning hierarchical representations from raw data, they require large labeled datasets and are computationally intensive—making them better suited for scenarios where abundant training data is available and model accuracy is critical, such as advanced threat hunting or cloud-based malware analysis. Conversely, RF models perform well on structured, tabular data and are faster to train and evaluate, making them ideal for low-latency environments or endpoint-based threat detection with limited resources. Feature selection can also be applied to the RF portion of the experiment to identify which of the 2,350 features in the expanded dataset contribute most significantly to performance, potentially reducing computational overhead. Enhancing the dataset with diverse and emerging malware samples would improve generalizability while implementing robust defense mechanisms could mitigate adversarial attacks. Real-time performance evaluation—latency, throughput, and resource usage—is crucial for practical deployment. Additionally, improving CNN explainability through methods like Grad-CAM or SHAP could yield more actionable insights for analysts. Leveraging transfer learning with pre-trained CNNs may also reduce training time and improve accuracy in cases with limited labeled data. Together, these directions aim to build scalable, resilient, and interpretable malware detection systems capable of adapting to evolving threats.

Acknowledgment

This material is based upon work supported by, or in part by, the National Science Foundation (NSF-SFS) under contract/award 2140638

References

Akhtar, M., & Feng, T. (2023). Evaluation of Machine Learning Algorithms for Malware Detection.

Sensors, *23*(2). https://doi.org/10.3390/s23020946

Blackberry Cylance. (2019). Artificial Intelligence: The Smarter Approach to Information Security [White

Paper]. In *Blackberry*.

https://www.blackberry.com/content/dam/bbcomv4/blackberry-com/en/products/resource-cen

ter/resource-library/ebooks/Replace-Your-AV-EBook.pdf

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32.

https://doi.org/10.1023/a:1010933404324

Brown, A., Gupta, M., & Abdelsalam, M. (2023, March 3). *Automated Machine Learning for Deep*

Learning based Malware Detection. arXiv.org. https://arxiv.org/abs/2303.01679

Djenna, A., Bouridane, A., Rubab, S., & Marou, I. M. (2023). Artificial Intelligence-Based Malware Detection, analysis, and Mitigation. *Symmetry*, *15*(3), 677. https://doi.org/10.3390/sym15030677

Gale, C., & Steele, R. (2020). Evaluating Performance Maintenance and Deterioration Over Time of Machine Learning - based Malware Detection Models on the EMBER PE Dataset. *Seventh International Conference on Social Networks Analysis, Management and Security (SNAMS).*

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

Raman, K. & Adobe Systems, Inc. (2012). Selecting Features to Classify Malware. *Product Security Incident Response Team (PSIRT).* http://2012.infosecsouthwest.com/files/speaker_materials/ISSW2012_Selecting_Features_to_Classify_Malware.pdf

S Anderson, H., & Roth, P. (2018). EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models. *ArXiv*, *abs/1804.04637*. https://doi.org/10.48550/arXiv.1804.04637

Saxe, J., & Berlin, K. (2015). Deep neural network based malware detection using two dimensional binary program features. *2015 10th International Conference on Malicious and Unwanted Software (MALWARE)*, 11–20. https://doi.org/10.1109/MALWARE.2015.7413680

Shafiq, M., Tabish, S., Mirza, F., & Farooq, M. (2009). A Framework for Efficient Mining of Structural Information to Detect Zero-Day Malicious Portable Executables. *FAST National University of Computer & Emerging Sciences, Islamabad, Pakistan.* https://www.researchgate.net/profile/Fauzan-Mirza/publication/242084613_A_Framework_for_Efficient_Mining_of_Structural_Information_to_Detect_Zero-Day_Malicious_Portable_Executables/links/0c96052e191668c3d5000000/A-Framework-for-Efficient-Mining-of-Structural-Information-to-Detect-Zero-Day-Malicious-Portable-Executables.pdf

Smith, J., Johnson, L., & Lee, K. (2022). A Comprehensive Review of Cross-Validation Techniques in Machine Learning Model Evaluation. *Journal of Machine Learning Research*, *15*, 123–145. https://www.scirp.org/reference/referencespapers?referenceid=3547271

VirusTotal. (2025). *VirusTotal*. https://www.virustotal.com/gui/intelligence-overview