

Enhancing malware detection capabilities using deep learning with advanced hyperparameter tuning

Walid El Mouhtadi, Yassine Maleh, Soufyane Mounir

LaSTI Laboratory, National School of Applied Sciences Khouribga, Sultan Moulay Slimane University, Beni Mellal, Morocco

Article Info

Article history:

Received May 23, 2024

Revised Sep 16, 2024

Accepted Sep 29, 2024

Keywords:

ANN

CNN

Data balancing

Deep learning

Malware detection

Optimization

RNN

ABSTRACT

As the threat landscape evolves with sophisticated malware and advanced persistent threats (APTs), the need for effective detection solutions increases. Traditional methods, such as signature-based and heuristic analysis, struggle to keep up with rapidly changing malicious activities. While machine learning offers a promising approach, it often falls short due to the manual extraction and selection of features, leading to time-consuming and error-prone processes. This research introduces a novel malware detection solution leveraging deep learning and focusing on portable executable (PE) file analysis to address these weaknesses. By customizing the hyperparameters of artificial neural networks (ANN), convolutional neural networks (CNN), and recurrent neural networks (RNN), the proposed approach enhances detection capabilities. The primary objective is to overcome the limitations of traditional and machine learning methods by tailoring these deep learning algorithms. The methodology includes a comparative study to demonstrate the advantages of the customized approach over conventional methods. Key findings reveal the proposed solution's superior performance, accuracy, and adaptability in combating evolving cyber threats. This research contributes to the development of robust and adaptive malware detection solutions.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Walid El Mouhtadi

LaSTI Laboratory, National School of Applied Sciences Khouribga, Sultan Moulay Slimane University

Casablanca, Morocco

Email: walid.elmouhtadi@usms.ac.ma

1. INTRODUCTION

The constantly changing threat landscape [1], characterized by advanced malware [2], requires innovative detection solutions. Traditional methods [3], such as signature-based [4] and heuristic analysis [5], are increasingly inadequate. Signature-based techniques [6], although initially effective, fail to address novel threats due to their reliance on known patterns [7]. Heuristic methods attempt to detect malicious behavior [8] but are prone to false positives and require constant updates.

Researchers have turned to machine learning techniques [9], which analyzes patterns in large datasets for malware detection. While machine learning has improved detection capabilities [10], it typically relies on manual feature extraction [11], which is labor-intensive and error-prone [12]. Recent advances in deep learning [13] offer promising solutions by automating feature extraction and adapting to evolving threats, particularly when dealing with PE files [14], which are a common target for malware [15]. Significant contributions in this area include Abualhaj *et al.* [16] used k-nearest neighbors (KNN) algorithm, which focuses on data transformation and normalization to enhance performance. Their MW-KNN model addresses the limitations of traditional signature-based methods, particularly for polymorphic and metamorphic malware. Another notable work is MalNet by Yan *et al.* [17] which discusses a deep learning

method for malware detection that analyzes Windows executable files through grayscale image conversion for CNN and opcode sequence analysis using LSTM networks. This dual approach improves detection accuracy and automates feature learning. Yılmaz and Bakır [18] used a system based on machine learning for android devices to optimize performance through methods like SMOTE and ClusterCentroids for imbalanced datasets, achieving a detection accuracy of 98.98%. ALGorain and Clark [19] investigated hyperparameter optimization techniques for malware detection, particularly focusing on Windows PE files, and demonstrated the importance of tailored parameter tuning. Lad and Adamuthe [20] proposed a deep learning model for static PE files malware detection, achieving high accuracy of 97.53% and 94.09% for the 2017 and 2018 datasets

Despite these advancements, challenges remain, particularly in automating feature extraction and adapting to new threats. Hyperparameter tuning in deep learning is a critical yet often underexplored aspect that can significantly impact performance [21]. This study builds upon these existing works by focusing on hyperparameter tuning across multiple deep learning models ANN, CNN, and RNN specifically applied to PE files. By customizing these parameters, our approach aims to surpass current detection accuracy levels and improve the detection of evolving malware threats, offering a more robust and comprehensive solution than previously proposed methods.

This research aims to advance malware detection by developing a highly optimized deep learning framework that enhances detection accuracy and improves precision to raise the adaptability of detection to new and sophisticated malware strains. Doing so, we seek to contribute a more resilient solution to the cybersecurity community and provide a foundation for future security research. Our approach is designed to readily apply to other security solutions, ultimately raising the maturity of the malware detection field.

The present paper is structured as follows: section 2 offers a detailed explanation of the adopted methodology and the proposed framework design, demonstrating how our approach addresses the challenges in current methods. Section 3 covers the findings, examinations, discussion, and assessment methodology, showing the effectiveness of our optimized models. Finally, section 4 delineates and summarizes the study's overall conclusion, along with the planned future work, highlighting our research's contributions and potential impact.

2. METHOD

2.1. Data collection

This research may lead to a new approach to deep learning, we've developed an efficient and optimized architecture and a strong workflow to address a specific challenge. We aim to enhance data processing and workflow by tackling weaknesses through methods like data balancing and hyperparameter tuning. This helps avoid issues caused by complex models, preventing overfitting for more effective results.

This research project focuses on analyzing a substantial number of malicious and legitimate files, leveraging datasets sourced from the Kaggle [22]. The testing phase incorporates diverse samples from Virustotal [23] and Virusshare [24]. This study aims to develop an effective deep learning algorithm for malware detection by utilizing a robust dataset comprising 137,000 files with 57 features, including 96,000 malicious and 41,000 legitimate samples. This approach ensures the effectiveness of data distribution in both the learning and testing phases, mitigating bias that could compromise the accuracy of results. A meticulous hyperparameter tuning process is conducted for the ANN, CNN, and RNN to optimize the performance of deep learning algorithms. This configuration aims to improve the confusion matrix values and enhance each model's accuracy and precision. Each deep learning algorithm undergoes training and evaluation using the prepared dataset. Evaluation metrics are computed to assess each model's accuracy, precision, and optimized confusion matrix. The best-performing model is selected based on its effective and robust confusion matrix and high accuracy and precision. This model is saved for future use in classifying unseen files.

When an unseen file requires analysis for classification, the header of its PE is extracted [25], feature selection is automatically performed based on importance, and the previously saved best model is utilized for classifying the file. The result of the classification (malicious or legitimate) is determined by the saved model, enhancing the detection capability and improving the quality and speed of file classification.

The examples below in Figure 1 detail the model optimization workflow, outlining each step in refining data processing and model training to maximize detection accuracy. Figure 2, in turn, illustrates the architecture designed for the accurate classification of previously unseen files, highlighting the notable impact of this approach on improving malware detection capabilities.

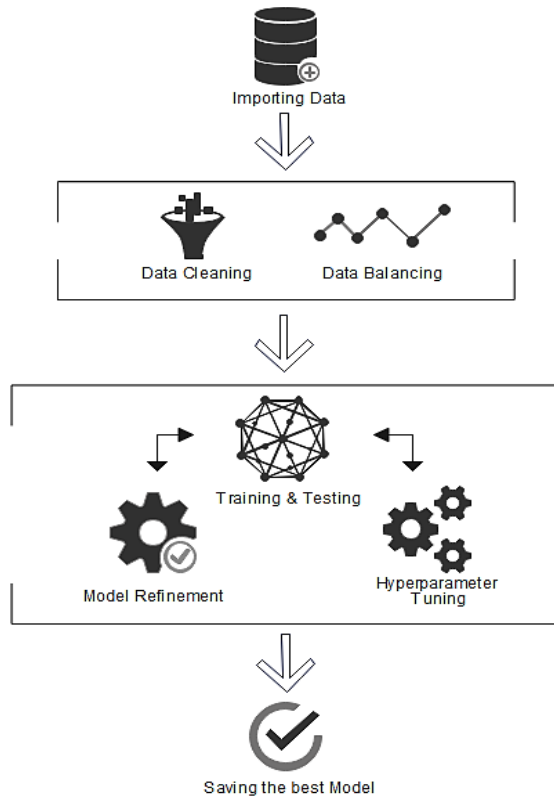


Figure 1. Model optimization workflow and data processing

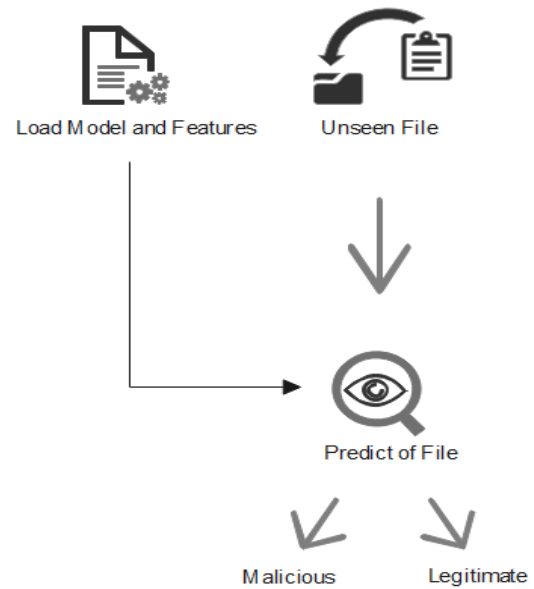


Figure 2. Advanced architecture for accurate classification of unseen files

2.2. Data fairness through data balancing

This step of data balancing ensures that the training data is evenly spread across all classes, preventing any bias in the model. Balancing the data is crucial, as an uneven distribution can lead to a distorted model. The Figures 3 and 4 show how the class distribution changes before and after balancing the data.

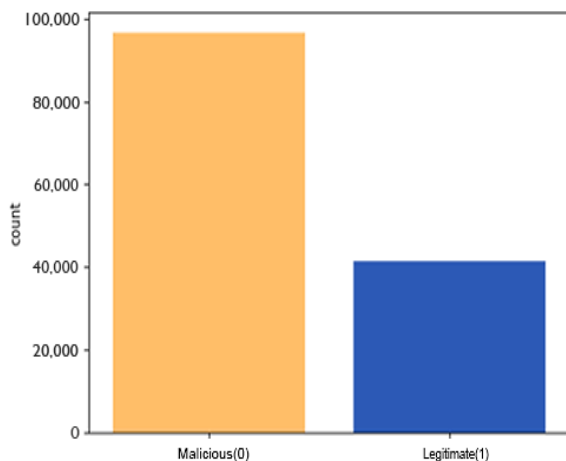


Figure 3. Before balancing

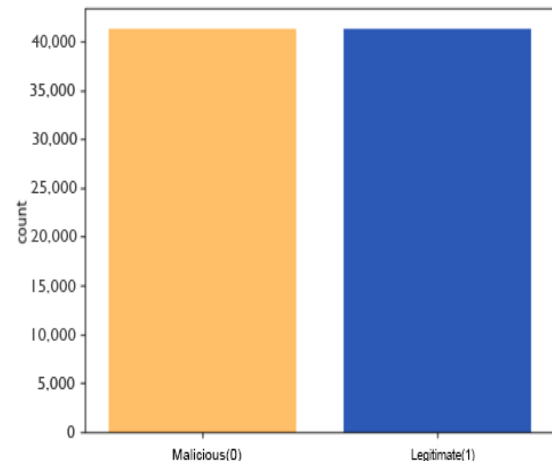


Figure 4. After balancing

Identifying key characteristics is essential for accurately categorizing files as either malicious or benign. After preprocessing the dataset by removing columns such as Name of the file, MD5, and label, a total of 54 features were retained from the original 57 columns, the meticulous selection of the most informative features becomes imperative to circumvent overfitting [26], preserve model interpretability, and

reduce computational expenses. The abundance of features can detrimentally affect model performance, introducing complexity and impeding pattern recognition. Furthermore, excess features extend the computational time required for training and prediction [27]. Thus, a comprehensive feature selection process becomes essential to streamline the model, enhance efficiency [28], and focus on the most pertinent features. In this study, feature selection performed by deep learning algorithms played a crucial role in aiding file classification, pinpointing 13 features essential for accurate categorization. The utilization of these key features significantly bolstered classification accuracy, underscoring the effectiveness of the deep learning algorithms in identifying pivotal features for file classification. Figure 5 presents the relevant features selected by deep learning and their importance.

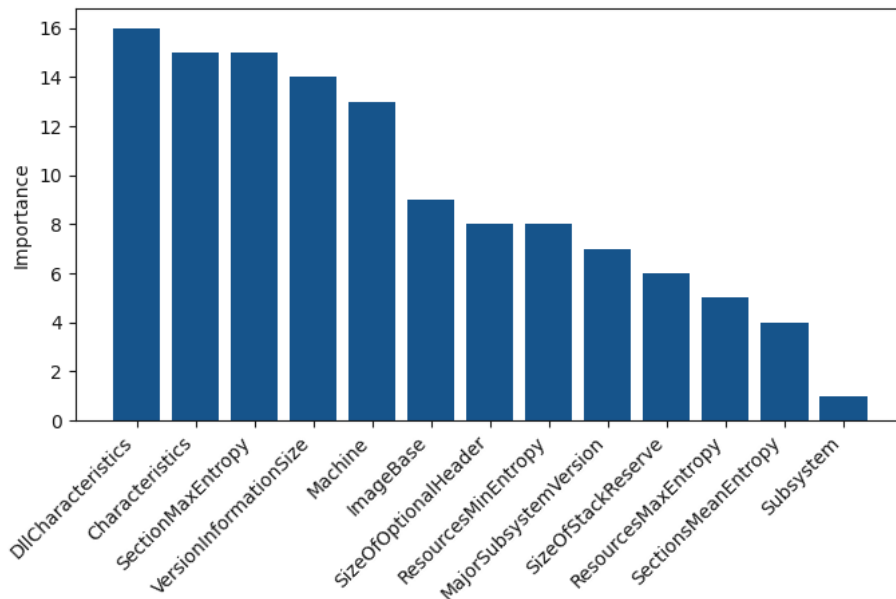


Figure 5. Important relevant feature

2.3. Experimental setup and execution environment

This research project carried out trials on a computing system equipped with an Intel Core i5 central processing unit (CPU) boasting 8 Gigabytes (GB) of random-access memory (RAM) and an Intel® UHD graphics processing unit (GPU) featuring 4.1 Gigabytes of dedicated video memory. The utilized operating system was Windows 10. The trials were executed using the Python programming language, specifically version 3.8.5, and an array of libraries, including NumPy, Pandas, Scikit-learn, PeFile, and TensorFlow. The data employed in the trials were locally stored as files.

3. PROPOSED DEEP LEARNING FOR MALWARE DETECTION FRAMEWORK

Once the fundamental attributes have been identified, the subsequent stage involves training and testing various classification algorithms using these attributes. This can be achieved by partitioning the data into a training set and a test set and utilizing the training set for the model's training, while the test set is employed to appraise the model's performance. The data division was conducted to guarantee the assessment of the accuracy of the deep learning algorithms. The training set comprised "X train" and "y train." "X train" encapsulates all the feature values for each row, excluding the label column, while "y train" exclusively contains the label values for each row. In the experiment, three deep learning algorithms were incorporated following the data division, including ANN, CNN, and RNN, the trained models were subsequently utilized to categorize the data in the test set "X test" and "y test" and assess their accuracy. The outcomes of this evaluation yield insights into the efficacy and performance of each deep learning algorithm.

Following common research practices, it is recommended to follow a specific series of steps. First, train and assess the computational methods. After completing this process, it's crucial to identify the most effective framework by calculating various metrics such as precision, false positives, and false negatives.

Finally, save the framework with the highest effectiveness as a file for future use. The efficiency of each computational method is evaluated using metrics like precision, false positive ratio, and false negative ratio. Framework accuracy is assessed using the scoring function from the Python Scikit library. Calculating false positive and false negative ratios is done using the confusion matrix, and the error matrix. This matrix summarizes the performance of a binary categorization technique by comparing predicted values with actual values, as illustrated in Table 1.

Table 1. Confusion matrix

		Actual	
		Positive (1)	Negative (0)
Predicted	Positive (0)	True positive	False positive
	Negative (1)	False negative	True negative

Note: True positives (TP): the number of correctly predicted instances as positive. False positives (FP): the number of instances predicted as positive but were actually negative. True negatives (TN): the number of correctly predicted instances as negative. False negatives (FN): the number of instances predicted as negative but were actually positive.

The metrics TPR, TNR, precision, and accuracy can be calculated using the values in the confusion matrix, like illustrated in Table 2. In summary, the true positive rate (TPR) and true negative rate (TNR) indicate the algorithm’s capacity to correctly forecast positive and negative instances. Precision gauges the algorithm’s effectiveness in generating precise positive predictions, while accuracy serves as a comprehensive metric for evaluating the algorithm’s overall performance.

Table 2. Metrics for performance algorithm

Metric	Description	Calculation
True positive rate (TPR)	Sensitivity or Recall: Fraction of actual positives correctly classified as positive.	$TPR = \left(\frac{TP}{FN} \right)$
False positive rate (TNR)	Specificity: Fraction of actual negatives incorrectly classified as positive.	$FPR = \left(\frac{FP}{FP + TN} \right)$
Precision	Fraction of positive predictions that were actually positive.	$Precision = \left(\frac{TP}{TP + FP} \right)$
Accuracy	Fraction of instances correctly classified by the algorithm.	$Accuracy = \left(\frac{TP}{TP + FP} \right)$

4. RESULTS AND DISCUSSION

A series of case studies were conducted, and the performance metrics for each default parameter for each algorithm under consideration are presented below in Table 3. These case studies encapsulate real-world scenarios, illustrating the practical efficacy of our models in distinguishing between malicious and legitimate files.

Table 3. Before tuning

Metric	ANN before hyperparameter tuning	CNN before hyperparameter tuning	RNN before hyperparameter tuning
Accuracy	0.49897	0.91526	0.75141
Precision	0.50411	0.91537	0.75160
True positive	0	6892	5421
True negative	8333	8239	7003
False positive	0	94	1330
False negative	8197	1305	2776
TPR	0.00000	0.84080	0.90179
FPR	0.00000	0.01128	0.01464
Running time (in sec)	9.23649	0.72364	9.23649

4.1. Comparison of model creation and evaluation

Hyperparameter tuning is a critical aspect of optimizing neural networks, and the journey from an original implementation to a refined model involves adjusting key parameters. Tables 4-6 showcase the transformation of hyperparameters of ANN, CNN, and RNN from the original implementation to recommended settings.

4.2. Performance metrics after tuning

Performance evaluation is crucial for understanding how tuning affects algorithm performance. Adjusting hyperparameters is a key process in optimizing deep learning models, and the impact of these adjustments is evident in the performance metrics. By examining the performance criteria values, one can see how fine-tuning enhances model effectiveness. The improvements achieved through this process highlight the significance of hyperparameter optimization in refining performance. This analysis is essential for evaluating the success of tuning and guiding future enhancements. To prepare that, we will review Table 7, which displays the values of performance criteria for each algorithm after tuning.

Table 4. Comparative analysis of hyperparameter settings in ANN optimization

Hyperparameter	ANN original implementation	ANN recommended hyperparameter
Number of neurons (Layer 1)	128 (Dense layer)	'units1' (128, Dense layer)
Dropout rate (Layer 1)	0.2 (Dropout layer)	N/A (Not tuned in Code 2)
Number of neurons (Layer 2)	64 (Dense layer)	'units2' (64, Dense layer)
Dropout rate (Layer 2)	0.2 (Dropout layer)	N/A (Not tuned in Code 2)
Number of neurons (Layer 3)	4 (Dense layer)	'units3' (4, Dense layer)
Learning rate	'adam' optimizer used (no specific learning rate)	'learning_rate' (0.001, GridSearchCV)
Number of epochs	20	5 (Epochs for GridSearchCV)
Batch size	32	32 (Batch Size for GridSearchCV)
Verbose	1 (Prints progress bar)	0 (Verbose for GridSearchCV)

Table 5. Comparative analysis of hyperparameter settings in CNN optimization

Hyperparameter	CNN original implementation	CNN recommended hyperparameter
Number of filters (Conv1)	32 (Conv1D layer)	32 (Best value from GridSearchCV)
Number of filters (Conv2)	64 (Conv1D layer)	64 (Best value from GridSearchCV)
Kernel size (Conv1)	3 (Conv1D layer)	3 (Best value from GridSearchCV)
Kernel size (Conv2)	3 (Conv1D layer)	3 (Best value from GridSearchCV)
Pooling size (MaxPool1)	2 (MaxPooling1D layer)	2 (Best value from GridSearchCV)
Pooling size (MaxPool2)	2 (MaxPooling1D layer)	2 (Best value from GridSearchCV)
Dense units	1 (Dense layer)	1 (Best value from GridSearchCV)
Learning rate	N/A	0.001 (Best value from GridSearchCV)
Number of epochs	20	5 (Best value from GridSearchCV)
Batch size	32	32 (Best value from GridSearchCV)
Verbose	0	0 (Verbose for GridSearchCV)

Table 6. Performance after hyperparameter settings in RNN optimization

Hyperparameter	RNN original implementation	RNN recommended hyperparameter tuning
Number of units	64 (SimpleRNN layer)	64 (Best value from GridSearchCV)
Learning rate	N/A	0.001 (Best value from GridSearchCV)
Number of epochs	20	5 (Best value from GridSearchCV)
Batch size	32	32 (Best value from GridSearchCV)
Verbose	0	0 (Verbose for GridSearchCV)

Table 7. Performance of algorithm after tuning

Metric	ANN before tuning	ANN after tuning	CNN before tuning	CNN after tuning	RNN before tuning	RNN after tuning
Accuracy	0.49897	0.876	0.91526	0.950	0.75141	0.820
Precision	0.50411	0.883	0.91537	0.947	0.75160	0.831
True positive	0	6759	6892	7333	5421	6570
True negative	8333	8245	8239	8223	7003	7489
False positive	0	88	94	110	1330	844
False negative	8197	1438	1305	864	2776	1627
True positive rate	0.00000	0.824	0.84080	0.895	0.90179	0.801
False positive rate	0.00000	0.010	0.01128	0.013	0.01464	0.101
Running time (sec)	9.23649	9.500	0.72364	0.750	9.23649	9.800

Figures 6-8 present the confusion matrices for the ANN, CNN, and RNN models after hyperparameter tuning, highlighting the classification accuracy and the impact of tuning on true positives, false positives, true negatives, and false negatives. Figure 9 presents the ROC Curves for ANN, CNN, and RNN models after hyperparameter tuning. These curves illustrate the trade-off between the true positive rate

and false positive rate for each model, highlighting the effects of hyperparameter adjustments on classification performance.

The performance metrics provide compelling evidence supporting the initial hypothesis that prioritizing hyperparameter tuning would significantly enhance the effectiveness and accuracy of deep learning models in malware detection. Among the models tested, the CNN consistently demonstrated superior performance, with post-tuning accuracy improving from 0.915 to 0.950 and precision from 0.915 to 0.947. The substantial increase in the CNN TPR from 0.840 to 0.895 further underscores its exceptional ability to capture and differentiate complex patterns in the data. This is reflected in its highest area under the curve (AUC) of 0.92, confirming the hypothesis that the CNN, when optimized, would be the most robust and efficient solution for this application.

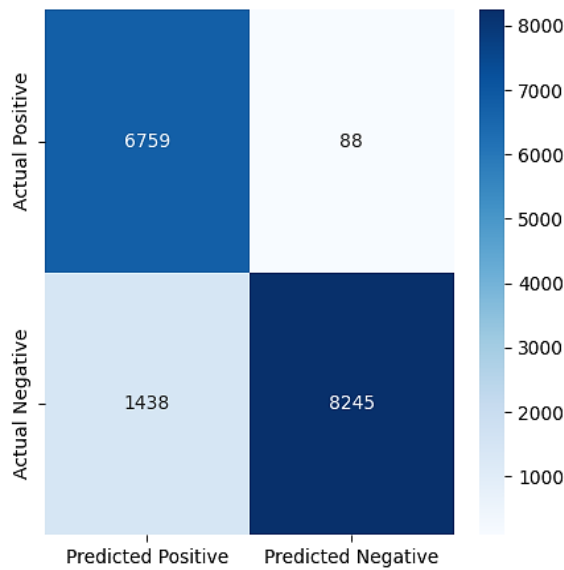


Figure 6. Confusion matrix for ANN after hyperparameter tuning

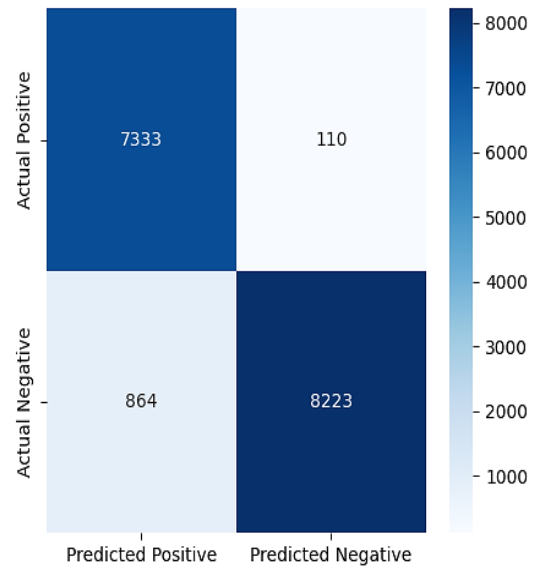


Figure 7. Confusion matrix for CNN after hyperparameter tuning

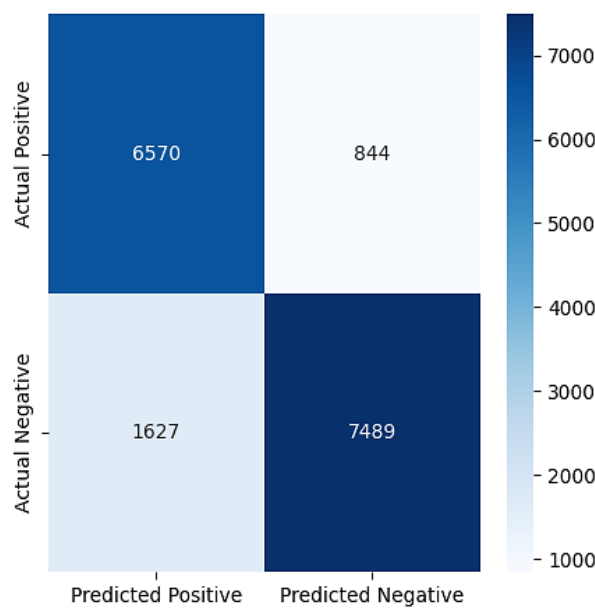


Figure 8. Confusion matrix for RNN after hyperparameter tuning

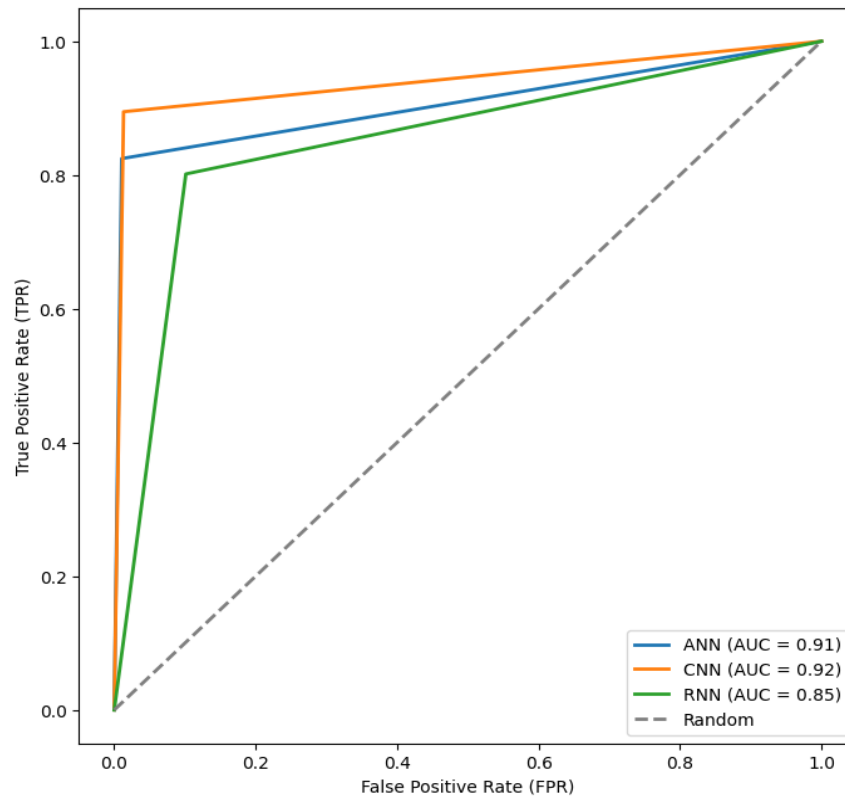


Figure 9. ROC curve for ANN, CNN, and RNN after hyperparameter tuning

Our choice to emphasize hyperparameter tuning as a critical component of the methodology is strongly supported by both theoretical foundations and empirical results. For instance, the significant improvements observed in the ANN, where accuracy jumped from 0.498 to 0.876 and precision from 0.504 to 0.883, validate the hypothesis that such tuning is essential for enhancing model performance. Theoretical insights in machine learning suggest that when properly tuned, models like ANN and CNN can better capture intricate relationships within data, leading to improved generalization and classification accuracy. This was evident as the ANN AUC improved to 0.91 post-tuning, aligning with our hypothesis regarding the potential of optimized models. In contrast, while the RNN also improved post-tuning with accuracy rising from 0.751 to 0.820 and precision from 0.751 to 0.831, it still lags behind the CNN in overall performance, achieving an AUC of 0.85. This outcome was anticipated in our hypothesis, which recognized that while RNN are well-suited for sequential data processing, they are less effective than CNN in capturing the complex spatial features critical for high-performance malware detection. This comparative analysis further solidifies the rationale behind our methodological choices, reinforcing the effectiveness of CNN in this domain.

In developing our approach, we prioritized models based on their strengths in different aspects of pattern recognition CNN for spatial data and RNN for sequential data while emphasizing the critical role of hyperparameter tuning. The results confirm that our initial hypothesis was correct, the CNN architecture, optimized through careful tuning, proves to be the most effective model for distinguishing between malware and non-malware samples. The significant gains across all models post-tuning also reaffirm the theoretical premise that optimal hyperparameter configurations are essential for unlocking a model's full potential. Our CNN model's superior performance, with an accuracy of 0.950 and an AUC of 0.92, surpasses KNN-based methods, complex hybrid models like MalNet, and feature-intensive machine learning approaches. Achieving these results through simpler, yet effective, hyperparameter tuning confirms our hypothesis that this approach would deliver higher precision and accuracy without requiring extensive pre-processing or complex configurations. The empirical results validate our methodology, showing that our work outperforms related methodologies in both accuracy and efficiency, making it the most reliable approach for malware detection as hypothesized. The applications and methods developed through this work are poised for adoption by researchers who are seeking ready-to-implement solutions, without the need for further energy-intensive investigations to enhance algorithm performance.

5. CONCLUSION

Our contribution to the evolution of deep learning algorithms ANN, CNN, and RNN has underscored the transformative power of hyperparameter tuning. This process emerged as a unifying force, guiding each algorithm toward enhanced performance by addressing the unique challenges inherent in different datasets. The significance of customization in model training was evident, as it enabled these models to achieve superior accuracy and precision. The divergence in performance among the algorithms became clear, with CNN standing out due to its exceptional ability to capture intricate patterns, demonstrating a nuanced strength distinct from ANN and RNN. Although ANN and RNN showed significant improvements, CNN ability to excel in malware detection tasks reaffirmed its position as the most robust model in this context. The iterative nature of hyperparameter tuning highlighted the adaptability and responsiveness of these deep learning models, contributing to their overall predictive capabilities. Looking ahead, future work will involve combining our recent contributions focused on refining malware detection using enhanced machine learning algorithms and hyperparameter tuning with this research to create a comprehensive anti-malware solution. This solution will incorporate the architecture outlined, where data samples undergo preprocessing and classification based on their type (byte or assembly). By leveraging techniques like opcode analysis for assembly code and multi-antivirus validation, this approach aims to enhance the accuracy and precision of detection. This comprehensive anti-malware framework could significantly contribute to future antivirus and endpoint detection and response (EDR) systems, offering a robust defense mechanism against evolving threats. Additionally, exploring ensemble methods to combine the strengths of multiple algorithms and leveraging transfer learning with pre-trained models will be crucial in advancing the field. Emphasis on interpretability and real-world applicability remains essential for widespread adoption, and continuous iteration to stay abreast of the latest developments in Deep Learning will be key to unlocking new possibilities.




REFERENCES

- [1] C. Do Xuan and M. H. Dao, "A novel approach for APT attack detection based on combined deep learning model," *Neural Computing and Applications*, vol. 33, no. 20, pp. 13251–13264, Oct. 2021, doi: 10.1007/s00521-021-05952-5.
- [2] *Data mining*. STYLUS Publishing, 2017.
- [3] S. K. Sahay, A. Sharma, and H. Rathore, "Evolution of Malware and its detection techniques," in *Advances in Intelligent Systems and Computing*, 2020, pp. 139–150. doi: 10.1007/978-981-13-7166-0_14.
- [4] J. Scott, "Signature based malware detection is dead," *Cybersecurity Think Tank, Institute for Critical Infrastructure Technology*, pp. 1–15, 2017.
- [5] M. Naseer et al, "Malware Detection: Issues and Challenges," *Journal of Physics: Conference Series*, vol. 1807, no. 1, p. 012011, Apr. 2021, doi: 10.1088/1742-6596/1807/1/012011.
- [6] R. Gutierrez, W. Villegas-Ch, L. Naranjo Godoy, A. Mera-Navarrete, and S. Luján-Mora, "Application of deep learning models for real-time automatic malware detection," *IEEE Access*, vol. 12, pp. 107742–107756, 2024, doi: 10.1109/ACCESS.2024.3436588.
- [7] Y. Fan, Y. Ye, and L. Chen, "Malicious sequential pattern mining for automatic malware detection," *Expert Systems with Applications*, vol. 52, pp. 16–25, Jun. 2016, doi: 10.1016/j.eswa.2016.01.002.
- [8] H. S. Galal, Y. B. Mahdy, and M. A. Atiea, "Behavior-based features model for malware detection," *Journal of Computer Virology and Hacking Techniques*, vol. 12, no. 2, pp. 59–67, May 2016, doi: 10.1007/s11416-015-0244-0.
- [9] A. V. Kozachok and V. I. Kozachok, "Construction and evaluation of the new heuristic malware detection mechanism based on executable files static analysis," *Journal of Computer Virology and Hacking Techniques*, vol. 14, no. 3, pp. 225–231, Aug. 2018, doi: 10.1007/S11416-017-0309-3/METRICS.
- [10] A. Ijaz et al., "Innovative machine learning techniques for malware detection," *Journal of Computing and Biomedical Informatics*, vol. 7, no. 1, 2024.
- [11] F. Nawshin, R. Gad, D. Unal, A. K. Al-Ali, and P. N. Suganthan, "Malware detection for mobile computing using secure and privacy-preserving machine learning approaches: A comprehensive survey," *Computers and Electrical Engineering*, vol. 117, Jul. 2024, doi: 10.1016/j.compeleceng.2024.109233.
- [12] J. Qiu, J. Zhang, W. Luo, L. Pan, S. Nepal, and Y. Xiang, "A survey of android Malware detection with deep neural models," *ACM Computing Surveys*, vol. 53, no. 6, 2021, doi: 10.1145/3417978.
- [13] F. Xiao, Z. Lin, Y. Sun, and Y. Ma, "Malware detection based on deep learning of behavior graphs," *Mathematical Problems in Engineering*, no. 1, Jan. 2019, doi: 10.1155/2019/8195395.
- [14] S. J. Kattamuri, R. K. V. Penmatsa, S. Chakravarty, and V. S. P. Madabathula, "Swarm optimization and machine learning applied to PE Malware detection towards cyber threat intelligence," *Electronics*, vol. 12, no. 2, Jan. 2023, doi: 10.3390/electronics12020342.
- [15] H. Bostani and V. Moonsamy, "EvadeDroid: A practical evasion attack on machine learning for black-box Android malware detection," *Computers and Security*, vol. 139, Apr. 2024, doi: 10.1016/j.cose.2023.103676.
- [16] M. M. Abualhaj, A. A. Abu-Shareha, Q. Y. Shambour, S. N. Al-Khatib, and M. O. Hiari, "Tuning the k value in k-nearest neighbors for malware detection," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 13, no. 2, pp. 2275–2282, Jun. 2024, doi: 10.11591/ijai.v13.i2.pp2275-2282.
- [17] J. Yan, Y. Qi, and Q. Rao, "Detecting Malware with an ensemble method based on deep neural network," *Security and Communication Networks*, pp. 1–16, 2018, doi: 10.1155/2018/7247095.
- [18] E. KAVALCI YILMAZ and H. BAKIR, "Hyperparameter tuning and feature selection methods for Malware detection," *Politeknik Dergisi*, vol. 27, no. 1, pp. 343–353, Feb. 2024, doi: 10.2339/politeknik.1243881.
- [19] F. T. ALGorain and J. A. Clark, "Bayesian hyper-parameter optimisation for malware detection," *Electronics*, vol. 11, no. 10, May 2022, doi: 10.3390/electronics11101640.




- [20] S. S. Lad. and A. C. Adamuthe, "Improved deep learning model for static PE Files Malware detection and classification," *International Journal of Computer Network and Information Security*, vol. 14, no. 2, pp. 14–26, Apr. 2022, doi: 10.5815/ijcnis.2022.02.02.
- [21] A. Djenna, A. Bouridane, S. Rubab, and I. M. Marou, "Artificial intelligence-based malware detection, analysis, and mitigation," *Symmetry*, vol. 15, no. 3, Mar. 2023, doi: 10.3390/sym15030677.
- [22] "Kaggle: your machine learning and data science community." Accessed: Aug. 11, 2024. [Online]. Available: <https://www.kaggle.com/>.
- [23] "VirusTotal-Home." Accessed: Aug. 11, 2024. [Online]. Available: <https://www.virustotal.com/gui/home/upload>.
- [24] "VirusShare.com." Accessed: Aug. 11, 2024. [Online]. Available: <https://virusshare.com/about>.
- [25] H. H. Al-Khshali and M. Ilyas, "Impact of portable executable header features on malware detection accuracy," *Computers, Materials and Continua*, vol. 74, no. 1, pp. 153–178, 2023, doi: 10.32604/cmc.2023.032182.
- [26] E. S. Alomari *et al.*, "Malware detection using deep learning and correlation-based feature selection," *Symmetry*, vol. 15, no. 1, Jan. 2023, doi: 10.3390/sym15010123.
- [27] D. Ö. Şahin, O. E. Kural, S. Akleylek, and E. Kılıç, "A novel permission-based Android malware detection system using feature selection based on linear regression," *Neural Computing and Applications*, vol. 35, no. 7, pp. 4903–4918, Mar. 2023, doi: 10.1007/s00521-021-05875-1.
- [28] K. Shaukat, S. Luo, and V. Varadharajan, "A novel deep learning-based approach for malware detection," *Engineering Applications of Artificial Intelligence*, vol. 122, Jun. 2023, doi: 10.1016/j.engappai.2023.106030.

BIOGRAPHIES OF AUTHORS






Walid El Mouhtadi    is a Ph.D. student at LaSTI Laboratory, ENSA Khouribga, part of Sultan Moulay Slimane University, Morocco, since 2022. He works as Manager Global SOC (Security Operations Center), Morocco, since 2024. He received Ingénieur d'État in Network and Telecommunications Engineering, from National School of Applied Sciences (École Nationale des Sciences Appliquées de Khouribga), from Sultan Moulay Slimane University, since 2021. His research interest includes cyber security, malware analysis, digital forensics, machine learning, and deep learning. He can be contacted at email: walid.elmouhtadi@usms.ac.ma.



Yassine Maleh    is an associate professor of cybersecurity and IT governance at Sultan Moulay Slimane University, Morocco, since 2019. He is a double Ph.D. in computer sciences and IT management. He is the founding chair of IEEE Consultant Network Morocco and founding president of the African Research Center of Information Technology and Cybersecurity. He is a senior member of IEEE. He has published over than 140 papers (international journals, book chapters and conferences/workshops), 27 edited books, and 5 authored books. He is the editor-in-chief of the International Journal of Information Security and Privacy. He can be contacted at email: y.maleh@usms.ma.



Soufyane Mounir    is associate professor in the National School of Applied Sciences of Sultan Moulay Slimane University, Beni Mellal, Morocco, since 2014. He got his Ph.D. in electronics and telecommunication, from University Hassan 1st, Morocco. His research is multidisciplinary that focuses on telecommunications, VoIP, signal processing, embedded systems and cyber security. He is an active member of LaSTI Laboratory, ENSA Khouribga. He can be contacted at email: s.mounir@usms.ma.