Dijlah Journal of Engineering Science (DJES) Vol. 3, No. 3, August, 2026, pp.463-472
ISSN: Printed: **3078-9656, Online: 3078-9664**, paper ID: 219

A Cybersecurity-Oriented Framework for Windows Malware Detection via Neural Architecture Search Using Advantage Actor-Critic

Muthana S. Mahdi[1]
[1] Department of Computer Science, College of Science, Mustansiriyah University, Baghdad, Iraq

Article Info

Article history:

Received May,04,2026
Revised Jul.,15,2026
Accepted Aug.,20,2026

Keywords:

Windows Malware Detection
Intelligent Agent
Neural Architecture Search
Advantage Actor-Critic
Deep Learning
Cybersecurity

ABSTRACT

The continuous growth of Windows-based systems has been accompanied by a parallel increase in sophisticated malware threats, posing serious risks to digital infrastructures. Traditional detection mechanisms, particularly signature-based and manually designed learning models, often struggle to cope with rapidly evolving malware variants and complex evasion techniques. This limitation highlights the need for adaptive and automated solutions capable of improving detection performance without relying heavily on expert-driven design. This study proposes a novel framework for Windows malware detection based on neural architecture search guided by an Intelligent Agent using the Advantage Actor-Critic approach. The proposed method formulates architecture design as a sequential decision process, enabling the agent to construct an optimized neural model by selecting appropriate components from a structured search space. This strategy reduces manual intervention while enhancing the model's ability to generalize across diverse malware patterns. The proposed framework was evaluated on the EMBER dataset, a widely recognized benchmark for static malware analysis. Experimental results demonstrate strong performance, achieving an accuracy of 0.971, along with superior results in AUC, precision, recall, and F1-score compared to existing approaches. These results confirm the effectiveness of the automatically discovered architecture in capturing meaningful feature interactions while maintaining computational efficiency. Overall, the integration of reinforcement learning with neural architecture design provides a scalable and practical solution for modern malware detection challenges.

Corresponding Author:

Muthana S. Mahdi
Department of Computer Science, College of Science, Mustansiriyah University, Baghdad, Iraq
Email: muthanasalih@uomustansiriyah.edu.iq

1. INTRODUCTION

Emerging computing environment has rendered cybersecurity a paramount issue of the new computing looks [1]. Malware is a dynamic and continually evolving threat on Windows platforms that aims at individuals, businesses and critical infrastructure. Malware developers are exploring the use of such techniques like polymorphism, obfuscation, and zero-day attacks to get around the traditional detection methods [2]. This has implied that the process of identifying the malware has been complicated and demands intelligent and adaptive methods that can be able to handle large and dynamic environments [3].

Traditional models of detection of malware like signature based methods are normally employed as a first line solution. They may be successful in identifying known threats, but are unable to detect the new and emerging malware variants [4]. One booby prize in solving this problem will be that the domain knowledge applies to systems of rules which determine whether to issue a warning or not [5]. To mitigate these weaknesses, machine learning (ML) and deep learning (DL) methods have been extensively used to extract features and detect them. A number of these methods enable the learning models to efficiently learn complex features, which in turn encourage them to achieve higher generalization accuracies than other traditional methods [6].

However, the current approach of learning is not free of restrictions as well. Neural designs are usually hand-designed and might not be specially suited to the fluctuating and fluctuating malware [7]. These models may be very sensitive to both the architecture and the hyperparameters employed and have to be carefully tuned and expert knowledge [8]. Moreover, there are models that conduct dynamic analysis, i.e., sandboxing potentially malicious files to execute them [8]. This provides useful information but increases the computational costs and limits the possibility of real-time applications. Additionally, some approaches have tried to enhance the malware detection accuracy using ensemble models and attention models, which have a higher computational overhead and slow down the detection process [9].

Another key consideration is the trade-off between accuracy and efficiency. While complex models can yield excellent results, they can be complex to maintain and scale. Alternatively, simpler models might be insufficient in capturing complex patterns in malware data [10]. Additionally, existing methods may struggle to cope with the ever-changing nature of malware, given that they are based on fixed model configurations that fail to adapt to the changing data characteristics. These observations suggest the need for more adaptable and smarter approaches that can automatically search for the best possible model [11].

To overcome these limitations, the present paper introduces a new approach for Windows malware detection, using neural architecture search with an Advantage Actor-Critic (A2C) intelligent agent. The motivation is to cast the design of neural network structures as a sequence of actions undertaken by an agent, which learns to build a successful model by making decisions about the components to be chosen at each step. Instead of relying on manual design, the new approach automatically searches a structured space for architectures that achieve high detection rates while being efficient.

The main contribution of this work is the combination of reinforcement learning and neural architecture design to develop an automated approach. The method eliminates the need for human input, simplifies the architecture to avoid potential overcomplexity, and enhances its ability to generalize to different malware samples. Moreover, the approach is based on static feature extraction, without the need for dynamic execution, while still being able to learn relevant information from the data. The model strikes balance between performance and efficiency making it feasible for real world applications.

In summary, this work provides an efficient and intelligent approach to malware detection to address the shortcomings of existing approaches. In this study, using reinforcement learning for automatically designing the architecture of the proposed detection framework boosts the performance and adaptability, and provides a solution for the cyber environment in the current era.

2. Related Work

Over the past few years, emphasis has been shifting towards data-driven and intelligent systems for Windows malware detection, namely, machine learning and deep learning. Early work in 2020 used neural networks for static analysis. Huang et al. [12] demonstrated the ability to classify the models based on extracted features of files. But these approaches were still unable to detect new families of malware due to their use of static features. Also, Amer and Zelinka [13] proposed dynamic analysis using API sequences. This provided improved contextual features at the expense of substantial computational overhead and the requirement for an isolated environment.

There was a shift to combined approaches in 2021. Wang et al. [15] proposed a hybrid IoT malware detection using static and dynamic techniques. This approach improved robustness, but it was unclear if this would work in the case of detection for Windows malware. Azeez et al. [16] proposed ensemble learning models by training two separate neural models to improve accuracy. Unfortunately, this also meant their model is more complicated and inefficient, making it difficult to implement.

In 2022, each of the modelling and algorithm optimisation techniques also advanced. Diener et al. [17] combined memory forensics with a big data pipeline for better detection. This approach was successful but needed extensive infrastructure to perform, which may be difficult. Ravi and Alazab [18] designed an attention-based convolutional model for feature extraction and classification, but it required heavy training. Algorain and Clark [19] implemented malware detection models using Bayesian optimization for better classification accuracy, but this method was based on a possible search space. Kudrekar and Rani [20] introduced a reinforcement learning-based classifier model that considered various behavioural traits of malware, but it was complex to implement and train. Kim et al. [21] performed a comprehensive study on the anti-analysis techniques of malware, providing information, rather than a detection model.

There were improvements in generalisation and analysis in 2023. Koçak et al. [22] adopted machine learning approaches with static features, achieving high accuracy on existing data but not generalising for new malware. Komarudin et al. [23] examined the use of artificial intelligence for malware identification, with improved accuracy but no consideration of dynamic features. Du et al. [24] employed a process-aware deep learning model,

using dynamic features. It performed well but required large-scale data with vulnerability to changes in the execution of code.

New advances have been made in 2024. Syeda and Asghar [25] were among the first to classify API calls to improve real-time detection, but it is still susceptible to some advanced obfuscation attacks. Zada et al. [26] benchmarked supervised machine learning models, while providing insights into model performance and selection, but not introducing a new method. Gururaja et al. [27] demonstrated the advantages of using ensemble methods to improve detection, but with higher model complexity and less adaptability. Imran et al. [28] studied adversarial attacks on machine learning-based detection, emphasising security but mostly in an adversarial setting. Khan and Nauman [29] introduced a transformer-based model to capture long-range dependencies in binary code to improve explainability and detection, but it is resource-intensive.

More recently, Mahdi [30] developed a deep neural network (DNN) model, combining convolutional neural networks (CNN) and recurrent neural networks (RNN) with attention, to enhance feature extraction. The approach has demonstrated good performance on several benchmarks with improved generalisation. But the structure of the model is determined by hand, and this makes the model more complex, which may slow down the speed.

These papers have greatly advanced the field of malware detection, but are not without limitations. It either requires manual intervention in architecture design, is resource-intensive, or lacks flexibility. Some of them employ intensive behavioral analysis or use ensemble techniques, which are difficult to implement.

To address these challenges, this study proposes a neural architecture search (NAS) approach using asynchronous advantage actor-critic (A2C) to automatically design a model to fit the dataset. This obviates the need for manual design, enhances generalization, and delivers performance comparable to the state-of-the-art with an efficient architecture that can be deployed in practice. Table 1 provides a summary of the related works with their strengths and weaknesses.

Table 1. Summary of related works

Ref.	Approach	Strength Points	Weak Points
[12]	Deep learning with static features	Effective classification using file features	Limited adaptability to new malware
[13]	Dynamic API-based analysis	Captures runtime behavior	High computational cost
[14]	Behavioral sequence modeling	Detects evolving threats	Sensitive to obfuscation
[15]	Hybrid static-dynamic framework	Improved robustness	Limited validation in Windows
[16]	Ensemble neural networks	High detection accuracy	High complexity and cost
[17]	Memory-based big data analysis	Real-time insights	Requires a large infrastructure
[18]	Attention-based CNN	Strong feature representation	Resource-intensive training
[19]	Bayesian optimization	Improved parameter tuning	Dependent on the initial setup
[20]	Reinforcement learning model	Captures multiple behaviors	Complex and difficult to tune
[21]	Anti-analysis study	Insight into malware evasion	Limited detection application
[22]	ML-based static detection	High accuracy on known samples	Poor generalization
[23]	AI-based detection study	Improved performance	Lacks behavioral integration
[24]	Process-aware deep learning	Captures runtime features	Data-intensive and sensitive
[25]	API behavior classification	Enhances real-time detection	Struggles with obfuscation
[26]	Comparative ML evaluation	Comprehensive benchmarking	No novel contribution
[27]	Ensemble-based detection	Improved performance	High complexity, limited generalization
[28]	Adversarial analysis	Highlights robustness issues	Limited practical use
[29]	Transformer-based model	High interpretability	Computationally expensive
[30]	Hybrid CNN–RNN with attention	High accuracy and generalization	Complex, manually designed

3. Proposed Method

This work proposes a systematic approach to Windows malware detection using a dynamic and flexible framework that combines neural architecture search and an Advantage Actor-Critic (A2C) intelligent agent. This approach allows automatic learning of the best neural architecture suited for malware data, rather than relying on manual design or static hyperparameter optimization. The underlying concept is to formulate the architecture design as a decision-making process, where the intelligent agent builds the network step by step, choosing the best components to build the network.

This method overcomes several challenges mentioned in previous work, such as a lack of generalization, computational inefficiency, fixed network structures, and vulnerability to feature changes. The new approach dynamically optimizes architecture to achieve high detection performance and efficiency, as opposed to relying on fixed architectures or expensive ensemble learning.

The strategy will focus on how to learn effective representations of malware learnable features, using a lightweight but powerful architecture. Reinforcement learning can be applied to do away with brute-force search or human expertise to make design decisions. Also, through the application of convolutional and recurrent modeling capabilities, the network is able to model both local interactions and implicit structural properties of the executables.

Finally, the given approach is a balance of flexibility, performance, and efficiency. The architecture is not hard-coded, but is obtained through an intelligent search process, rewarding the performance. This enables more flexibility and adaptability to emerging types of malwares. Then there comes exposition of foundation blocks of proposed approach in the subsequent subsections.

3.1. Dataset Collection and Preparation

The proposed method leverages the EMBER dataset for fair and targeted evaluation. The rationale behind this choice is to remove the possibility of confounding variables, as well as the creation of a representative and workable benchmark in the Windows malware classification. The EMBER [31] is famous for providing rich statistical characteristics of Portable Executable files, both benign and malicious specimens, that can be used to carry out large-scale machine learning research.

The data set contains different feature types such as file headers, imported functions, byte histograms, and metadata, which, along with the various types of features, reflect the considerable importance of executable files. Such characteristics enable the model to learn discriminative patterns without necessarily running dynamic analysis, and, therefore, eliminate computation costs and allow for easy deployment.

Data is organized in a procedure pipeline to ensure a quality audience. The first one is that features are standardized to eliminate scaling effects and contribute to convergence. To guarantee the quality of data, imputation of incoherent values is done by applying the common methods. Dimensionality reduction is also used to remove unneeded and noisy features, and as much information as possible is retained.

Data is then split in a conventional manner in 70 percent training, 15 percent validation and 15 percent testing. This is to prevent leakage and in order to have a valid evaluation. In addition, we use slight data augmentation so that our model can better generalize by introducing small perturbations to feature values.

This method of data preparation provides the input to the machine-learning system to be well-cleaned, balanced and realistic. It also helps in achieving the goal of developing a fast and light malware detection system.

3.2. Neural Architecture Construction Using an A2C-Based Intelligent Agent

The suggested method frames neural architecture evolution as a game of reinforcement learning, in which an intelligent agent that uses the same reinforcement signal as the neural architecture is used to solve the game. Instead of specifying actions, parameters, and layers, the agent searches a given space of design, and gradually builds the network architecture by selecting actions that enhance detection performance.

The agent starts with an empty network. In each step the agent is given the current state (the current network) and performs an action, which corresponds to a design decision, like adding a convolutional layer, specifying the number of filters, or adding a recurrent element. After a fully specified architecture, the network gets trained to obtain a fixed number of epochs, and the reward is the F1-score of the trained network that trained on the validation set. This reward assists the agent in coming up with better architectural decisions.

The search space is judiciously crafted. It uses the requisite architectural detail without being too detailed. The search over the number of convictions, filter size, activation functions, pooling, sequence modeling layers and regularization. This exploration was not accidental but through design decisions.

The A2C method enhances efficiency compared to conventional neural architecture search, which can be costly in terms of computational resources, by training a policy that favors good architectures. It also avoids overfitting by training various candidate models under identical conditions.

When the search process stalls, the optimal architecture is chosen for training. The performance of this architecture is discussed in Section 4. Table 2 provides the state–action space used by the A2C agent during architecture search.

Table 2: State–Action Space Used by the A2C Agent During Architecture Search

State (Decision Stage)	Description	Available Actions
Input Representation	Defines input reshaping strategy	Raw vector
Convolution Depth	Number of convolution layers	1, 2, 3
Convolution Filters	Number of filters per layer	32, 64, 128
Kernel Size	Size of the convolution kernel	3, 5
Activation Function	Non-linearity choice	ReLU, Leaky ReLU
Pooling Inclusion	Whether to include pooling	Yes, No
Pooling Type	Type of pooling	Max, Average
Pooling Size	Pooling window size	2, 3
Recurrent Layer Inclusion	Add sequence modeling layer	None, LSTM
LSTM Units	Size of LSTM layer	64, 128
Dense Layer Depth	Number of fully connected layers	1, 2
Dense Layer Size	Number of neurons	64, 128, 256
Dropout Inclusion	Apply dropout	Yes, No
Dropout Rate	Dropout value	0.3, 0.5
Output Layer	Final classification layer	Sigmoid

3.3. Final Selected Network Architecture

After network architecture search, the A2C agent selects a hybrid architecture that is efficient yet effective. The final network shows typical design principles discovered during the search, such as the tendency to form shallow convolutional models with a single recurrent layer and weak regularization. Figure 1 shows the final architecture chosen by the intelligent agent after the network architecture search.

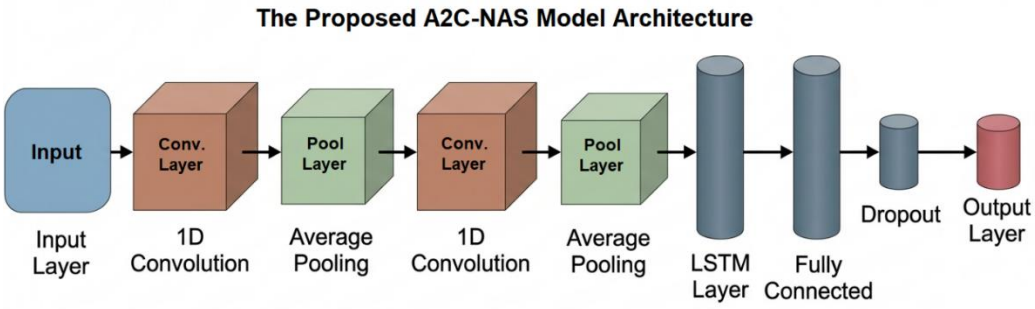

Figure 1. The architecture of the final model discovered by the A2C-based intelligent agent.

This architecture starts with an input layer that accepts the feature vector input. This is followed by two one-dimensional convolutional layers to model local interactions. The layers employ medium-sized filters and kernel sizes to capture patterns while keeping the model complexity low. The activation function is Leaky ReLU to enhance gradient propagation and prevent neurons from dying.

Average pooling is used after each convolutional layer to reduce the dimensionality of feature maps and enhance generalization to small feature variations. The agent chose Average pooling as it resulted in smoother feature pooling here.

Following feature extraction, a single LSTM layer is used to capture implicit relationships between feature groups. This enables modelling of dependencies between feature groups, even in non-sequential data representations.

The features are then fed into a fully connected layer that enhances the representation and is used for classification. One fully connected layer was deemed sufficient to avoid over-complexity. A dropout layer is added to avoid overfitting and improve the network's generalization. The dropout rate of 0.5 was to be used to fight overfitting. The final layer uses a sigmoid activation function to provide an output that is the probability that the sample is malicious.

This is a simple yet efficient design. There are no redundant layers and branches on it, but it is extremely expressive. This architecture is based on the decision of the agent to select efficient solutions that contribute to good performance, and with a short time of training.

Table 3 is the conclusion of the architecture, and Table 4 is the hyperparameter configurations. These configurations indicate the last architecture adopted by the A2C agent after convergence.

Table 3: The architectural configuration of the selected final model.

Layer No.	Layer Type	Configuration	Description
1	Input Layer	Feature vector	Receives processed data
2	1D Convolution	64 filters, kernel size 3, Leaky ReLU	Extract local patterns
3	Average Pooling	Pool size 2	Reduce dimensionality
4	1D Convolution	128 filters, kernel size 5, Leaky ReLU	Deeper feature extraction
5	Average Pooling	Pool size 2	Further reduction
6	LSTM Layer	128 units	To model implicit dependencies between feature groups
7	Fully Connected	128 neurons	Feature refinement
8	Dropout	Rate 0.5	Prevent overfitting
9	Output Layer	Sigmoid	Binary classification

Table 4: Hyperparameter Values Used

Hyperparameter	Value
Optimizer	AdamW
Initial Learning Rate	0.0001
Batch Size	64
Training Epochs	50
Dropout Rate	0.5
A2C Learning Rate	0.0003
Discount Factor	0.99

This architecture demonstrates how a clever search strategy may result in a balanced architecture without assuming extremely complex architectures. It reaches a high representation power and can be used practically.

4. Results and Analysis

In this part, a thorough analysis of the proposed network of the A2C-based neural network, used to detect Windows malware, is provided. It begins with the experimental setting, elaborating on the implementation and training variables. Then, the offered model is tested against the usual measures of performance, displaying the effectiveness and durability of the strategy. Finally, it contrasts the proposed method with those of the state of the art used in related works to demonstrate the superior performance of the proposed method in terms of accuracy, efficiency, and generalization.

4.1 Experimental Setup

The given framework was experimentally tested to see whether it could detect Windows malware or not through automatic exploration of an architecture discovered by the A2C-equipped intelligent agent. The development of the code was done with the help of TensorFlow and Keras in Python. The experiments were

executed using a powerful computer with a graphics processor (NVIDIA RTX 3080), an Intel Core i7 processor, and 32 GB of RAM.

It was possible to train the models effectively and ensure the architecture search and the final model converged quickly. Training of both the A2C agent and the candidate architectures was performed simultaneously at a limited number of training epochs to ensure that an efficient exploration with manageable computation needs was being performed.

4.2 Performance Evaluation

The EMBER dataset was used in running the test using the above method of data preparation. Accuracy, precision, recall, F1-score, and the area under the receiver operating characteristic (ROC) curve (AUC) using equations (1-5) were used to provide a complete understanding of the performance of the models [32].

$$\text{Accuracy} = \frac{\text{TP+TN}}{TP+TN+FP+FN} \quad (1)$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \quad (2)$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}} \quad (3)$$

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision+Recall}} \quad (4)$$

$$AUC = \sum_{i=1}^{n-1} (FPR_{i+1} - FPR_i) \times \frac{TPR_{i+1} + TPR_i}{2} \quad (5)$$

TN is used as the true negative result, TP is used as the true positive result, FN is used as the false negative result, and FP is used as the false positive result. Moreover, FPR means the level of false positive rate, and TPR means the level of true positive rate.

These measures provide a general measure of the model in its capacity to effectively detect malicious and benign samples and to avert false positives and false negatives. The performance metrics of the EMBER data are demonstrated in Table 5.

Table 5: Performance Metrics on Dataset.

Metric	Value
Accuracy	97.1
Precision	97.2
Recall	97.0
F1-Score	97.1
AUC	97.9

These results demonstrate that the proposed model works quite well and is efficient in all metrics. The high accuracy of 97.1% indicate that the model could be effective in classifying malware and benign software. The high precision score means a low false-positive or in real life conditions false alarm may impact a business. Likewise, the recall value is extremely high so that the model can identify a high percentage of malware, and this will decrease the false negative. The F1-score also proves precision and recall, and high value of AUC is used to prove good discrimination with different classification boundaries.

The high performance is explicable in terms of architecture search process. The A2C agent has found a simple yet powerful architecture that effectively captures feature interactions without adding unnecessary complexity. The use of convolutional layers and a recurrent element allow the network to capture local and global interactions between features. Furthermore, dropout regularization and hyperparameter tuning play a role in enhancing generalization and preventing overfitting. Figure 2 shows the performance measures on the EMBER dataset.

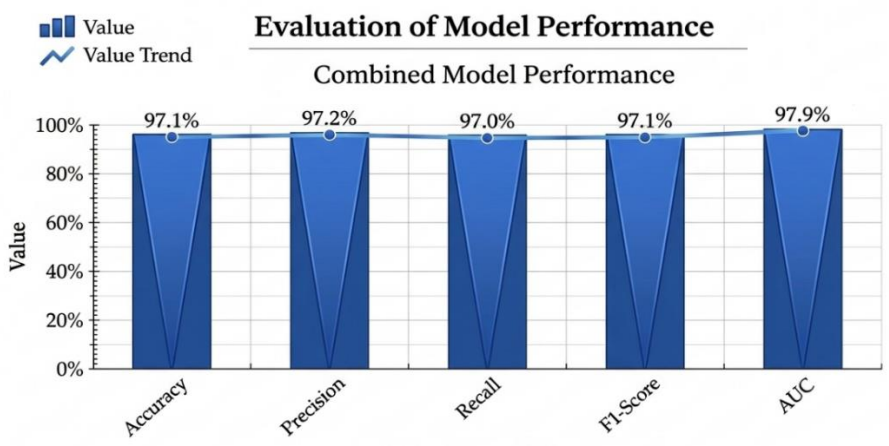

Figure 2. Performance metrics of the proposed A2C-NAS model across the EMBER dataset.

To better understand the learning of the proposed model, Figure 3 illustrates the training and validation accuracy as a function of the number of epochs. The stability in convergence and low generalization gap confirms the effectiveness of the proposed architecture designed by the A2C agent.

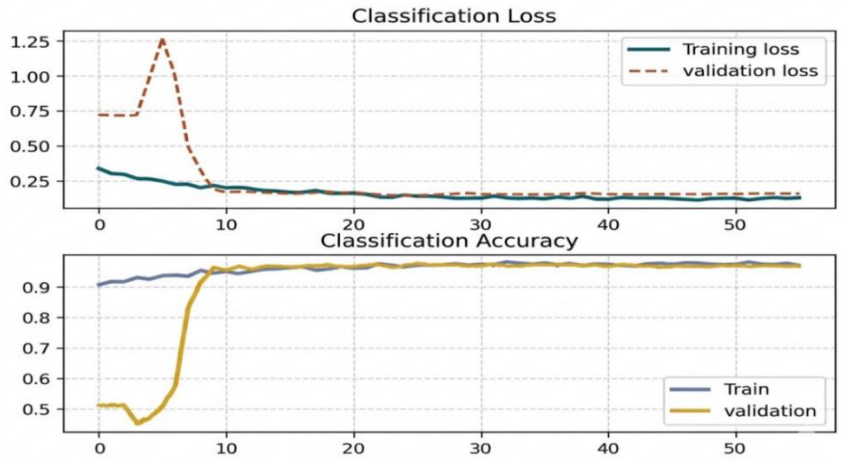

Figure 3: The training and validation accuracy trajectories over successive epochs.

4.3 Comparison with Related Works

The proposed approach is compared with some representative works from the aforementioned related works in order to further evaluate its effectiveness. The comparison with related works on the EMBER dataset is presented in Table 6.

Table 6: Comparison with Related Works

Method	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
[19]	92.9	92.8	93.2	92.9
[23]	94.2	93.9	94.5	94.1
[25]	96.0	96.2	95.9	96.0
[30]	96.7	96.1	96.8	96.4
Proposed Method	97.1	97.2	97.0	97.1

The results show that the proposed method outperforms several other methods reported in the literature. The proposed framework's key strengths are its ability to generate a dynamic architecture tailored to the dataset. Moreover, the model has the ability to achieve high detection accuracy with low computational complexity. In summary, the results show that the proposed framework achieves a good compromise between accuracy, efficiency, and model size.

Figure 4 presents a comparison between the proposed A2C-NAS and other approaches. The results show that our approach outperforms the baselines on every evaluation metric.

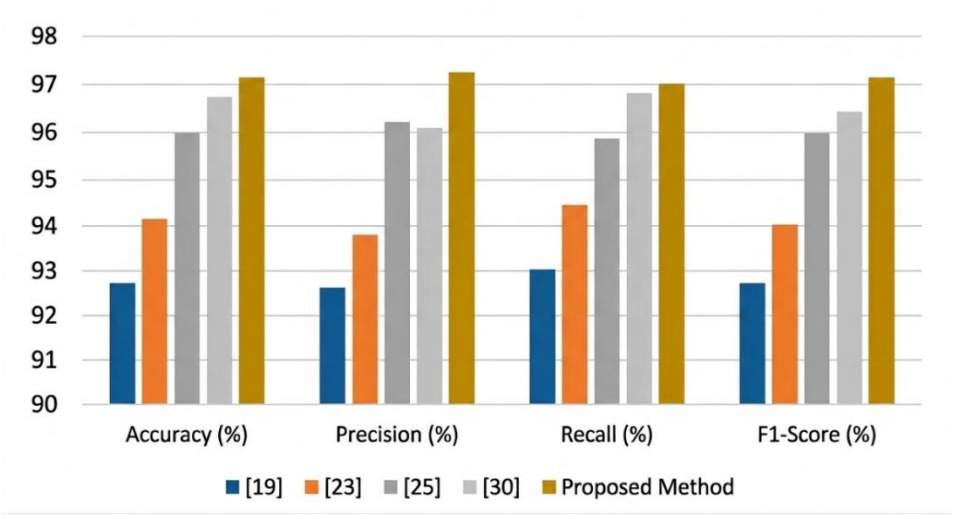

Figure 4. Graphical comparison between the proposed A2C-NAS framework and related methods.

The results indicate that the use of reinforcement learning for neural network architecture design is an effective approach for Windows malware detection. Our method strikes a good balance between detection performance, computational efficiency, and model size, making it ideal for practical cybersecurity applications.

5. CONCLUSION

This paper proposed an adaptive approach for Windows malware detection by leveraging neural architecture search with an Intelligent Agent. The proposed method overcomes several shortcomings of current approaches, including the need for manual architecture design, a lack of adaptivity to new malware variants, and the complexity-computational efficiency trade-off. Treating architecture design as a decision-making process, the proposed approach allows the agent to explore and construct an optimal network architecture that matches the data distribution. The evaluation results showed the proposed model exhibits high detection accuracy on the EMBER dataset, achieving competitive performance across several metrics, such as accuracy, precision, recall, F1-score, and AUC. Based on these findings, it can be concluded that the proposed architecture is effective in capturing the interactions and dependencies among features both at the local level and the global level. Furthermore, the simplicity of the model increases the ease of training and makes it much more efficient than more complicated ensemble or attention-based models. Another important work in this area is the reduction of relying on expert knowledge in designing the models. The reinforcement learning allows system to discover the most productive configuration without the presence of a hideous manual experiment. This not only enhances the scalability but also the generalization capability of the model to new and emerging malware. One of the most promising directions in coming up with a highly effective and efficient system of malware detection is the neural architecture search using an intelligent agent. Further improvements can be made by incorporating some of the features of other types, cross-dataset validation, and inducement of robustness against adversarial attacks.

ACKNOWLEDGEMENTS

The authors thank the Computer Science Department, College of Science, Mustansiriyah University (uomustansiriyah.edu.iq), for supporting this study.

REFERENCES

[1] Y. M. Mohialden, M. T. Younis, S. A. Salman, and E. A. W. Hachim, "Challenges and advancements in quantum cryptography: Standardization, security risks, and practical implementations," *Proc. Int. Conf. Applied Innovations in IT (ICAIIT)*, vol. 13, no. 4, pp. 307–314, 2025.
[2] J. Ferdous, R. Islam, A. Mahboubi, and M. Z. Islam, "A survey on ML techniques for multi-platform malware detection: Securing PC, mobile devices, IoT, and cloud environments," *Sensors*, vol. 25, no. 4, p. 1153, 2025.
[3] M. Tanha and S. Kafaie, "A review of explainable machine learning for Android malware detection and analysis," *IEEE Access*, 2025.
[4] S. Berrios, D. Leiva, B. Olivares, H. Allende-Cid, and P. Hermosilla, "Systematic review: Malware detection and classification in cybersecurity," *Applied Sciences*, vol. 15, no. 14, p. 7747, 2025.
[5] A. Basim, I. M. Ali, R. A. Lateef, and Z. L. Ali, "Enhanced ransomware traffic detection using a hybrid ResNeSt101 and Isolation Forest framework," in *Proc. SPIE, Third Int. Conf. Emerging Trends in AI and Computational Technologies (ICONEST 2025)*, vol. 14141, pp. 87–96, 2026.
[6] M. Alshoulie and A. Mehmood, "Deep learning approaches for malware detection: A comprehensive review of techniques, challenges, and future directions," *IEEE Access*, 2025.
[7] I. K. Abbas, M. S. Mahdi, A. Basim, and K. I. Abbas, "An agent-based reinforcement learning framework for dynamic cryptographic security," *Dijlah J. Eng. Sci.*, vol. 3, no. 1, pp. 377–391, Mar. 2026.
[8] S. Alzahrani, Y. Xiao, S. Asiri, J. Zheng, and T. Li, "A survey of ransomware detection methods," *IEEE Access*, 2025.
[9] S. F. Ali, M. R. Abdulrazzaq, and M. T. Gaata, "Learning techniques-based malware detection: A comprehensive review," *Mesopotamian J. CyberSecurity*, vol. 5, no. 1, pp. 273–300, 2025.
[10] W. Almobaideen, O. Abu Alghanam, M. Abdullah, S. B. Hussain, and U. Alam, "Comprehensive review on machine learning and deep learning techniques for malware detection in Android and IoT devices," *Int. J. Inf. Security*, vol. 24, no. 3, p. 110, 2025.
[11] Y. Dehfouli and A. Habibi Lashkari, "Memory analysis for malware detection: A comprehensive survey using the OSCAR methodology," *ACM Comput. Surv.*, vol. 58, no. 4, pp. 1–58, 2025.
[12] X. Huang, L. Ma, W. Yang, and Y. Zhong, "A method for Windows malware detection based on deep learning," *J. Signal Process. Syst.*, vol. 93, no. 2–3, pp. 265–273, 2020.
[13] E. Amer and I. Zelinka, "A dynamic Windows malware detection and prediction method based on contextual understanding of API call sequence," *Computers & Security*, vol. 92, p. 101760, 2020.
[14] F. Çatak, A. Yazı, O. Elezaj, and J. Ahmed, "Deep learning based sequential model for malware analysis using Windows exe API calls," *PeerJ Comput. Sci.*, vol. 6, p. e285, 2020.
[15] C. Wang, Z. Zhao, F. Wang, and Q. Li, "A novel malware detection and family classification scheme for IoT based on DEAM and DenseNet," *Security Commun. Networks*, vol. 2021, pp. 1–16, 2021.
[16] N. Azeez, O. Odufuwa, S. Misra, J. Oluranti, and R. Damaševičius, "Windows PE malware detection using ensemble learning," *Informatics*, vol. 8, no. 1, p. 10, 2021.
[17] M. Dener, G. Ok, and A. Orman, "Malware detection using memory analysis data in big-data environment," *Applied Sciences*, vol. 12, no. 17, p. 8604, 2022.
[18] V. Ravi and M. Alazab, "Attention-based convolutional neural network deep learning approach for robust malware classification," *Computational Intelligence*, vol. 39, no. 1, pp. 145–168, 2022.
[19] F. ALGorain and J. Clark, "Bayesian hyper-parameter optimisation for malware detection," *Electronics*, vol. 11, no. 10, p. 1640, 2022.
[20] S. Kudrekar and U. Rani, "Classification of malware using multinomial linked latent modular double Q learning," *Indonesian J. Electr. Eng. Comput. Sci.*, vol. 28, no. 1, p. 577, 2022.
[21] M. Kim, H. Cho, and J. Yi, "Large-scale analysis on anti-analysis techniques in real-world malware," *IEEE Access*, vol. 10, pp. 75802–75815, 2022.
[22] A. Koçak, E. Söğüt, M. Alkan, and O. Erdem, "Detection of different Windows PE malware using machine learning methods," *Politeknik Dergisi*, vol. 26, no. 3, pp. 1185–1197, 2023.
[23] I. E. Maulani, T. Herdianto, M. O. Laksana, F. Syawaludin, and K. Komarudin, "Exploring the effectiveness of artificial intelligence in detecting malware and improving cybersecurity in computer networks," *Eduvest: J. Universal Studies*, vol. 3, no. 4, 2023.
[24] C. Du *et al.*, "Toward detecting malware based on process-aware behaviors," *Security Commun. Networks*, vol. 2023, pp. 1–16, 2023.
[25] D. Syeda and M. Asghar, "Dynamic malware classification and API categorisation of Windows portable executable files using machine learning," *Applied Sciences*, vol. 14, no. 3, p. 1015, 2024.
[26] I. Zada *et al.*, "Fine-tuning cyber security defenses: Evaluating supervised machine learning classifiers for Windows malware detection," *Computers, Materials & Continua*, vol. 80, no. 2, pp. 2917–2939, 2024.
[27] H. Gururaja, N. Khandige, N. Nayak, R. Srivatsan, and N. S., "Ensemble learning for robust malware detection in the Windows 7 environment," *Int. Res. J. Adv. Engg. Hub*, vol. 2, no. 2, pp. 261–270, 2024.
[28] M. Imran, A. Appice, and D. Malerba, "Evaluating realistic adversarial attacks against machine learning models for Windows PE malware detection," *Future Internet*, vol. 16, no. 5, p. 168, 2024.
[29] S. Khan and M. Nauman, "Interpretable detection of malicious behavior in Windows portable executables using multi-head 2D transformers," *Big Data Mining and Analytics*, vol. 7, no. 2, pp. 485–499, 2024.
[30] M. S. Mahdi, "A deep learning–based hybrid neural network model for malware detection," *Journal of Techniques*, vol. 8, no. 1, pp. 62–70, 2026.
[31] "EMBER: Dataset for training static PE malware machine learning models," *Kaggle*. [Online]. Available: https://www.kaggle.com/datasets/dhoogla/ember-2018-v2-features. [Accessed: Mar. 14, 2026].
[32] M. S. Mahdi, Z. L. Ali, A. R. Rashid, N. K. Ibrahim, and A. W. Abdulghafour, "A hybrid deep learning model for facial emotion recognition: Combining multi-scale features, dynamic attention, and residual connections," *Proc. Int. Conf. Applied Innovations in IT (ICAIIT)*, vol. 13, no. 2, pp. 69–77, 2025.