Journal of Information & Optimization Sciences
ISSN 0252-2667 (Print), ISSN 2169-0103 (Online)
Vol. 47 (2026), No. 5-A, pp. 1603–1611
DOI : 10.47974/JIOS-2250

GCANQGM : Design of an efficient graph convolutional AHP neural network with Deep Q generative adversarial network for classification of android malwares

Harshal Devidas Misalkar †
School of Computing Science & Engineering
VIT Bhopal University
Kothrikalan
Sehore 466114
Madhya Pradesh
India

Pon Harshavardhanan *
School of Computing Science Engineering and Artificial Intelligence
VIT Bhopal University
Kothrikalan
Sehore 466114
Madhya Pradesh
India

Abstract

Android malware is also changing fast and is becoming problematic to conventional static and dynamic detection techniques that tend to ignore real-time API communications, log patterns, and access patterns. The current paper suggests a new Android malware detection and classification system based on the combination of Graph Convolutional Neural Networks (GCN) and multi-source data fusion comprising of logcat outputs, API calls, permissions, device logs, and debug messages. The GCN outputs are used to optimize Analytical Hierarchy Process (AHP) weights and Deep Q-GAN (DQ-GAN) is used to improve malware classification. The proposed model is also more precise, accurate, recall, AUC, and specificity than detection delay, showing a better adaptability and strength to emerging Android malware threats.

Subject Classification: *68M25.*

Keywords: *Android threat intelligence, Behavioral malware classification, Graph-based deep learning, Reinforcement learning for cybersecurity, Dynamic API call analysis.*

† *E-mail:* harshal.misalkar2019@vitbhopal.ac.in
* *E-mail:* pon.harshavardhanan@vitbhopal.ac.in (Corresponding Author)

© Taru Publications

I. Introduction

Android enables billions of gadgets all over the globe, yet its open source and wide personalization have triggered malware propagation, endangering user privacy, monetary and system resources [1]–[3]. Statics can analyze code without running it, but they can be difficult to defeat obfuscation and encryption, whereas dynamics can be used to observe runtime behavior, but are frequently not capable of detecting conditionally triggered attacks and often use a lot of resources [4], [5]. Besides, previous researchers often consider API calls, permissions, or logs independently, which restricts contextual knowledge [6]. These drawbacks emphasize the necessity of a combined, dynamic framework that will be able to model the interrelationships between heterogeneous data sources [7].

Recent studies include machine learning, deep learning and graph-based Android malware classification [8]. Although CNN and GNN models enhanced the feature representations, most of them are limited by the issues of small data fusion and scalability [9]. Rule generation using AHP and GAN-based modeling improved the accuracy of the decision and adversarial resistance but without full adaptability [10]. New paradigms like blockchain auditing, federated learning and others reinforce collaborative and privacy-conserving detection further [11].

II. Proposed GCNN-AHP-DQGAN Framework

Current malware detecting models with Android do not have rule-based engines in place, and tend to be high-complexity [12], however the efficiency of this detecting declines with the number of malware classes. In order to overcome such drawbacks, we are going to suggest an effective GCNN-AHP-DQGAN model as shown in Figure 1.

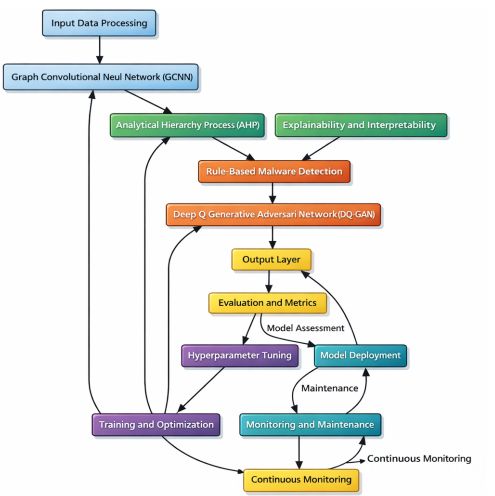

Figure 1
Design of the proposed model for malware analysis

The suggested architecture, which will have been used to detect Android malware, is presented in Figure 2. The input data are preprocessed and learned in the form of features through a graph based learning through GCNN. AHP produces rule-based decisions, which are classified with DQGAN. The framework has training, hyperparameter optimization, evaluation, deployment, explainability, and monitoring aspects that make sure it has an adaptive and scalable malware detection [13].

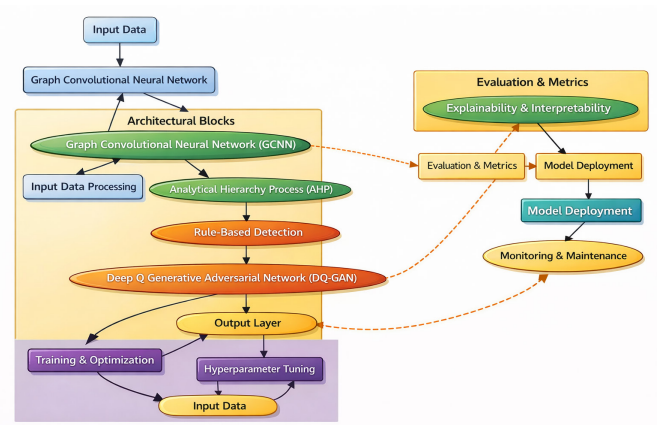

Figure 2
Flow of the proposed model for Android Malware Analysis

Android behavioral dataset is defined as a graph G(V,E) where the nodes are extracted behavioral attribute and the edges capture inter feature dependencies. Graph convolution processing is built on the structural basis of the connectivity strength between pairs of features as represented in Matrix A [14].

Graph convolution layer maintains node embeddings which is updated using normalized information of neighbourhoods. In this case, A has self-loops and D is the degree matrix. This model guarantees that there is stable propagation of features and that there is reduction of scale imbalance.

$$H(l+1) = \sigma\left(D^{-21}A^{D^{-21}}H(l)W(l)\right) \quad (1)$$

The degree of node i is a measure of all the influence of connected features. It is calculated as the total of weights of adjacency. This is necessary to facilitate normalization when operations of graph convolution are being performed [15].

$$di = \sum_{j=1}^{n} aij \quad (2)$$

The feature importance generated by GCNN is used to produce priority weights as the pairwise comparison matrix 20. The first eigenvector w of the largest eigenvalue is used to provide consistency in hierarchical decision modelling.

$$Cw = \lambda maxw \tag{3}$$

Raw eigenvector weights are scaled in order to get proportional scores of significance. This ensures that the cumulative contribution is one, and this makes it possible to perform a balanced multi-criteria assessment between malware behavioral dimensions.

$$w' = \sum_i \frac{w}{wi} \tag{4}$$

The score for every malware class k is calculated as a weighted projection of feature vector y. This linear unification incorporates hierarchical priorities and behavioral evidence obtained out of the application.

$$S_k = w_k^T y \tag{5}$$

The malware category that is predicted is the highest-scoring class. This rule of decision is used to guarantee the optimal selection based on weighted evaluation criteria produced by the AHP-enhanced GCNN outputs.

$$k^* = \arg maxSk \tag{6}$$

In case of rule-based validation, every rule feature vector fr is tested with optimized weights. This allows interpretation classification through mapping domain-specific behavioral rules with acquired importance factors.

$$R_r = w_k^T * f_r \tag{7}$$

Rule vectors represent certain indicators of behavior, like suspicious system calls, unusual validation patterns on the certificates, or permission violation. These structured vectors have explainable cybersecurity reasoning.

$$fr1 = [s, u] \tag{8}$$

$$fr2 = [c, v, p] \tag{9}$$

III. Deep Q-GAN Framework

The DQ-GAN model comprises of a malware classifier that is adversarialized with a generator and a discriminator. Discriminator appraises authenticity through hierarchical feature extraction which enhances adaptive detection performance as well as classification reliability.

Generator Mapping

$$G(z, x) = g(z, x;\ \theta_g) \tag{10}$$

The generator generates synthetic malware representations of feature based on a random noise vector z mixed with contextual feature vector x.

Composition of Generator Layer Wise.

$$g(z, x;\ \theta_g) = g_{L\left(g_{\{L-1\}}(\cdots g_{1(z,x)}\cdots)\right)} \tag{11}$$

The generator is designed in form of series of L neural layers. At each level, the latent representation output at the prior level is refined, and the hierarchical process of extracting nonlinear patterns of behaviour among fused Android malware features is performed.

Single Generator Layer Technology.

$$g_{\ell(y)} = a_{\ell(W_\ell y + b_\ell)} \tag{12}$$

The affine transformation with a nonlinear activation is done at each layer of the generators. W l and b l are the weight matrix and the bias of the ellipt l layer and a l is the ReLU activation that increases the nonlinear modeling capacity.

Expanded Generator Forward Pass

$$G(z,x) = a_L\left(W_L a_{\{L-1\}}\left(\ldots\ a_{1(W_1[z,x]^T + b_1)}\ \ldots\right) + b_L\right) \tag{13}$$

This term denotes the entire forward propagation of all the generator layers. The input [z, x] T is gradually converted through the linear mappings and activations until the last synthetic malware feature vector is generated.

Discriminator Mapping

$$D(y,x) = d(y,x;\ \theta_d) \tag{14}$$

The discriminator is fed with a real or a generated sample y and a contextual vector x and produces the probability that the sample is genuine.

Layer-Wise Discriminator Composition

$$d(y,x;\ \theta_d) = d_M\left(d_{\{M-1\}}\left(\ldots\ d_{1(y,x)}\ \ldots\right)\right) \tag{15}$$

M stacked layers are used to make up the discriminator. The successive levels of representation will produce an increasingly higher level of discriminatory representations that will allow the network to differentiate between actual malware patterns and adversarial generated samples.

A single layer of Discriminator works as follows.

$$d_{m(z)} = a_{m(W_m z + b_m)} \tag{16}$$

The discriminator layers have linear transformation with weight matrix W m and bias b m, with the subsequent ReLU activation function a_m.

IV. Result Analysis and Discussion

The paper proposes an Android malware detection system hybrid wherein GCNN, AHP, and DQ-GAN are used to detect data and correctly classify them. This kind of multi-source information as logcat logs, API calls, permission patterns, debug traces and device logs are fused and modeled using GCNN to detect relational dependencies. The learned representations maximize the weights of AHP to effectively detect the rules and the classification of the app signatures to a specific malware family is maximized by the learned representations of DQ-GAN. To ensure the effectiveness, the proposed GCANQGM model was experimentally compared to ESFCG-OMM, EDLNN, and GNN with controlled condition, which ensured the same and credible performance analysis.

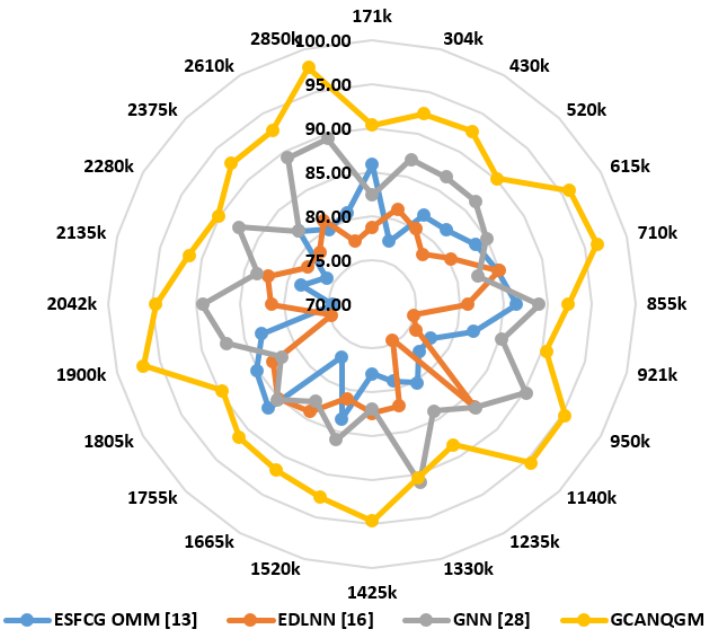

Figure 3
Observed Precision to determine Android Malwares using AHP Rules

Figure 3 displays a radar comparison between ESFCG-OMM, EDLNN, GNN, and GCANQGM with the increasing number of malware samples. GCANQGM is always more successful in all the evaluation ranges, being more stable and scalable.

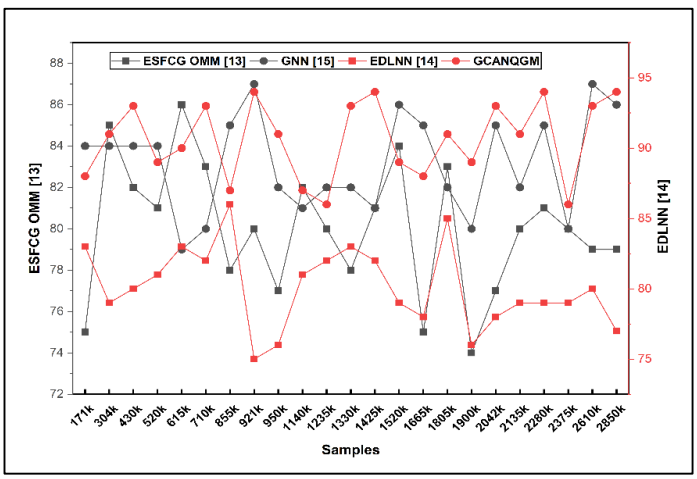

Figure 4
Observed Accuracy to determine Android Malwares using AHP Rules

Fluctuations in competing models are especially apparent at large datasets, and demonstrate the strength and flexibility of GCANQGM to various classification settings. The accuracy performance of ESFCG-OMM, EDLNN, GNN, and GCANQGM at different sizes of malware samples are shown in Figure 4. The model of GCANQGM records the most accurate results with consistent improvement whereas other models exhibit variability and poor performance.

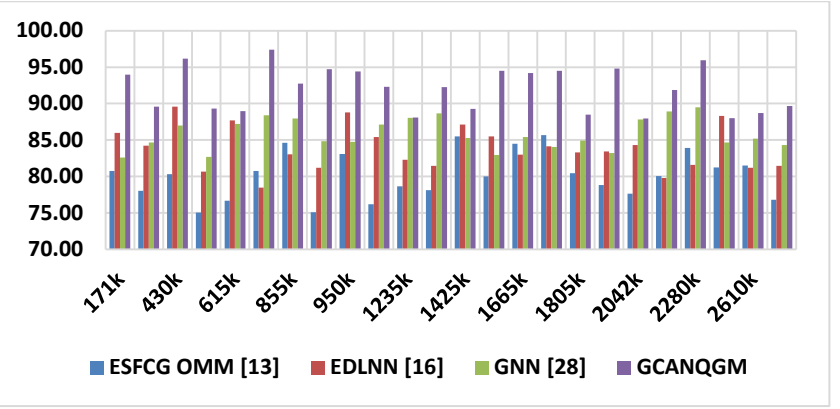

Figure 5
Observed Recall to determine Android Malwares using AHP Rules

In Figure 5, the comparison of the precision of ESFCG-OMM, EDLNN, GNN, and GCANQGM against the sample size of malware is presented. The highest precision values are always achieved by GCANQGM and it has high potential in identifying malicious applications correctly with minimum cases of false positives. GNN and EDLNN perform variably even though they improve moderately as larger datasets are being used. ESFCG-OMM is still relatively low.

V. Conclusion & Future Scopes

This paper rated Android malware prediction models in a comprehensive manner and showed that the proposed GCANQGM framework is better compared to the ESFCG-OMM, EDLNN, and GNN methods. GCANQGM had better precision, accuracy, recall, AUC, and specificity but at low classification delay, which guaranteed good real-time detection with fewer false positives and false negatives. The fact that it outperforms various malware types is indicative of good flexibility and durability of the product in real-life applications. The GCANQGM offers an Android malware detection solution that is efficient, scalable and high and performances, and it provides better security assurance and user experience in dynamic malware threat Android-based environments.

References

[1] D. K. A. Dhanya, P. Vinod, Y. Y. Suleiman, A. Bashar, A. David, T. Abhiram, A. Antony, K. S. Ashil, and T. G. Kumar, "Obfuscated malware detection in IoT Android applications using Markov images and CNN," *IEEE Systems Journal*, vol. 17, no. 2, pp. 2756–2766 (Jun. 2023) doi: 10.1109/JSYST.2023.3238678.

[2] G. Renjith, P. Vinod, and S. Aji, "Evading machine-learning-based Android malware detector for IoT devices," *IEEE Systems Journal*, vol. 17, no. 2, pp. 2745–2755 (Jun. 2023), doi: 10.1109/JSYST.2022.3215014.

[3] G. Suarez-Tangil and G. Stringhini, "Eight years of rider measurement in the Android malware ecosystem," *IEEE Transactions on Dependable and Secure Computing,* vol. 19, no. 1, pp. 107–118 (Jan.–Feb. 2022), doi: 10.1109/TDSC.2020.2982635.

[4] I. Almomani, A. Alkhayer, and W. El-Shafai, "An automated vision-based deep learning model for efficient detection of Android malware attacks," *IEEE Access*, vol. 10, pp. 2700–2720 (2022), doi: 10.1109/ACCESS.2022.3140341.

[5] L. D. Costa and V. Moia, "A lightweight and multi-stage approach for Android malware detection using non-invasive machine learning techniques," *IEEE Access,* vol. 11, pp. 73127–73144 (2023), doi: 10.1109/ACCESS.2023.3296606.

[6] J. Qiu, Q.-L. Han, W. Luo, L. Pan, S. Nepal, J. Zhang, and Y. Xiang, "Cyber code intelligence for Android malware detection," *IEEE Transactions on Cybernetics*, vol. 53, no. 1, pp. 617–627 (Jan. 2023), doi: 10.1109/TCYB.2022.3164625.

[7] Y. Ban, S. Lee, D. Song, H. Cho, and J. H. Yi, "FAM: Featuring Android malware for deep learning-based familial analysis," *IEEE Access*, vol. 10, pp. 20008–20018 (2022), doi: 10.1109/ACCESS.2022.3151357.

[8] L. Huang, J. Xue, Y. Wang, D. Qu, J. Chen, N. Zhang, and L. Zhang, "EAODroid: Android malware detection based on enhanced API order," *Chinese Journal of Electronics*, vol. 32, no. 5, pp. 1169–1178 (Sep. 2023), doi: 10.23919/cje.2021.00.451.

[9] G. Aldehim, M. A. Arasi, M. Khalid, S. S. Aljameel, R. Marzouk, H. Mohsen, I. Yaseen, and S. S. Ibrahim, "Gauss-mapping black widow optimization with deep extreme learning machine for Android malware classification model," *IEEE Access*, vol. 11, pp. 87062–87070 (2023), doi: 10.1109/ACCESS.2023.3285289.

[10] L. Gong, Z. Li, H. Wang, H. Lin, X. Ma, and Y. Liu, "Overlay-based Android malware detection at market scales: Systematically adapting to the new technological landscape," *IEEE Transactions on Mobile Computing*, vol. 21, no. 12, pp. 4488–4501 (Dec. 2022), doi: 10.1109/TMC.2021.3079433.

[11] R. Yumlembam, B. Issac, S. M. Jacob, and L. Yang, "IoT-based Android malware detection using graph neural network with adversarial defense," *IEEE Internet of Things Journal,* vol. 10, no. 10, pp. 8432–8444 (May 2023), doi: 10.1109/JIOT.2022.3188583.

[12] H. Alamro, W. Mtouaa, S. Aljameel, A. S. Salama, M. A. Hamza, and A. Y. Othman, "Automated Android malware detection using optimal ensemble learning approach for cybersecurity," *IEEE Access,* vol. 11, pp. 72509–72517 (2023), doi: 10.1109/ACCESS.2023.3294263.

[13] C. Gao, M. Cai, S. Yin, G. Huang, H. Li, W. Yuan, and X. Luo, "Obfuscation-resilient Android malware analysis based on complementary features," *IEEE Transactions on Information Forensics and Security,* vol. 18, pp. 5056–5068 (2023), doi: 10.1109/TIFS.2023.3302509.

[14] S. N. Ajani, J. Moulick, J. Mohur, R. H. J. Rani, A. D. Sonawane, and P. Shende, "Enhancing secure access in RF communications through advanced elliptic curve encryption," *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 29, no. 2-A, pp. 571–579 (2026), doi: 10.47974/JDMSC-2498.

[15] D. Motwani, V. Chitre, V. Bhosale, M. M. Nashipudmath, S. Shinde, and A. Nerurkar, "Implementing secure elliptic curve cryptography for blockchain-based financial transactions," *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 29, no. 2-A, pp. 617–627 (2026), doi: 10.47974/JDMSC-2504.

Received April, 2025