computer sciences and mathematics forum

MDPI

Proceeding Paper

DSGCNN-DA: A Deep Stacked Graph Convolutional Neural Network with Dynamic Aggregation for Malware Behavioral Learning †

Ghida Almusned [1], Lama Almutairi [1], Emna Benmohamed [1,2,*] and Rana Albelaihi [1]

[1] Department of Computer Science, College of Engineering and Information Technology, Onaizah Colleges, Qassim 51911, Saudi Arabia; 451210063@oc.edu.sa (G.A.); 431210094@oc.edu.sa (L.A.); ralbelaihi@oc.edu.sa (R.A.)
[2] Research Groups in Intelligent Machines, National School of Engineers (ENIS), University of Sfax, BP 1173, Sfax 3038, Tunisia
* Correspondence: emna.benmohamed@oc.edu.sa
† Presented at the 1st International Conference on Emerging Tech & Innovation (ICETI), Buraydah, Saudi Arabia, 10 February 2026.

Abstract

Malware remains a major threat to computer systems, posing serious risks to security and privacy by stealing sensitive data, disrupting services, and compromising system integrity. Traditional detection methods are often ineffective against rapidly evolving malware. In response, data-driven deep learning has emerged as a powerful alternative. Recent models have demonstrated promising performance in detecting malicious behavior by learning from these behavioral traces. Behavior-based detection represents a significant advancement in the fight against malware. This paper introduces a deep stacked Graph Convolutional Network (GCN) for effective malware behavioral analysis. The aggregation of multiple GCN layers and blocks results in dynamically performed Jumping Knowledge (JK) method, especially Long Short-Term Memory (LSTM). LSTM-based JK dynamically selects and weights the most informative GCN layers for each node to improve the model's ability. Experimental results demonstrate the superior performance of our deep stacked Graph Convolutional Network with Dynamic Aggregation (DSGCN-DA) model, achieving an accuracy of 98.93% on the API-Call-Sequences dataset, outperforming the state-of-the-art approaches.

Keywords: behavioral malware detection; deep learning; graph convolutional network; jumping knowledge

1. Introduction

The widespread adoption of internet technologies has led to a surge in cybersecurity threats, marked by increasingly frequent and sophisticated cyberattacks [1]. Among these, malicious software, commonly known as malware, serves as the primary tool attackers use to compromise victims' systems. Anomaly detection in attributed networks plays a pivotal role across diverse domains such as cybersecurity, social network analysis, and bioinformatics, where identifying suspicious or atypical behavior is essential for system integrity and insight discovery [2]. Several studies have explored malware detection using deep learning on behavioral graphs and API call sequences. For instance, authors in [3] employed LSTM for malware classification on the Mal-API-2019 dataset, comprising 7107 malware samples. By treating malware detection as a text classification problem, their LSTM-based model

Check for updates

Academic Editor: Sulaiman Al Amro

Published: 15 April 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

achieved accuracies ranging from 83.5% to 98.5%, with an overall accuracy of up to 98.50%. The approach exhibited low model complexity and robustness to class imbalance, though it required relatively long training times and did not report testing time or memory usage.

In recent years, Graph Convolutional Networks (GCNs) have attracted significant attention due to their outstanding performance in processing graph-structured data [4]. They have demonstrated substantial academic and practical value across a wide range of application domains. In [5], a behavioral malware detection method based on Deep Graph Convolutional Neural Networks (DGCNNs) was proposed, following an eight-step process that analyzes Portable Executable (PE) files in a Cuckoo Sandbox, extracts API call sequences, and constructs behavioral graphs for binary classification. Using the API-Call-Sequences dataset, the study found that LSTM outperformed two DGCNN models in malware detection, while the AUC metric demonstrated the models' effectiveness. PCA visualization revealed distinct classification behaviors across different architectures. The study also highlighted the need for additional regularization techniques to mitigate overfitting observed after ten epochs. Building on these works, we propose a stacked Deep Graph Convolutional Neural Networks with dynamic aggregation using Jumping Knowledge. This LSTM-based JK dynamically selects and weights the most informative GCN layers for each node for constructing richer and more discriminative node representations, which improves the model's ability to capture hierarchical and temporal patterns in malware behavior graphs.

In summary, our work offers the following contributions:

- We construct behavioral graphs from malware API call sequences to capture structural relationships.
- A dynamic aggregation using Jumping Knowledge based on LSTM (JK-LSTM) mechanism aggregates important node features, enhancing accuracy and interpretability in malware detection.
- A stacked Deep GCN enhanced with Jumping Knowledge (JK) is applied to accurately model and classify malware behavior.

The remainder of this paper is organized as follows. Section 2 reviews related work, Section 3 details the introduced DSGCN-DA model for anomaly detection, Section 4 describes the dataset, experimental protocol, and results, Section 5 presents the model's performance evaluation, and Section 6 concludes with future work.

2. Related Work

2.1. Deep Learning Models-Based Malware Analysis

In recent years, the cybersecurity landscape has rapidly evolved due to the increasing sophistication and volume of cyber threats. Malware remains a persistent and adaptive menace, encompassing diverse types such as viruses, worms, trojans, spyware, ransomware, and logic bombs, each employing unique behavioral patterns and attack strategies [6]. Precise classification of malware families is essential for reinforcing preventive measures and improving the efficiency of threat analysis for security professionals. Malware analysis, a critical component of cybersecurity, entails the systematic examination of malicious software to determine its functionality, origin, and potential impact. Traditional detection techniques, particularly signature-based methods, are often ineffective against polymorphic and obfuscated variants that evade static pattern matching. Consequently, data-driven approaches, especially deep learning (DL), have become prominent for enabling more robust and adaptive malware detection and classification.

Deep learning models, a subset of machine learning, use multi-layered neural architectures to automatically extract high-level features and learn complex patterns from large-scale, high-dimensional data. Their ability to operate on both labeled and unlabeled

https://doi.org/10.3390/cmsf2026013009

data enables effective performance across various learning paradigms: (1) Semi-supervised learning [7], which combines limited labeled data with abundant unlabeled data [8]. (2) Unsupervised learning, useful for clustering and anomaly detection [9]. These capabilities allow deep models to achieve high detection accuracy with low false-positive rates, making them essential components in protecting modern cyber infrastructures. Additionally, (3) reinforcement learning has been explored in malware detection, where software agents learn optimal decision-making strategies through interaction with an environment [10].

In recent years, there has been a growing shift toward behavioral malware detection, which observes program execution in controlled environments (e.g., sandboxes). This approach is robust against evasion techniques such as code obfuscation, dynamic linking, and encryption [11]. To model the temporal and structural dynamics of malware behavior, particularly sequences of system or API calls, researchers have employed various deep learning architectures. Recent studies have explored various deep learning architectures for malware detection by utilizing low-level system behaviors. In [12], the authors employed a Convolutional Neural Network (CNN) to analyze opcode sequences by integrating both static and dynamic features. The authors in [13] investigated the effectiveness of Recurrent Neural Networks (RNNs) in capturing sequential patterns for malware classification. The proposal in [14] is a Bidirectional LSTM (BI-LSTM) model enhanced with embedding and convolutional layers to capture semantic relationships among API calls. The aim of the study in [5] is to introduce a dynamic malware analysis approach based on directed cyclic graphs derived from API call sequences, effectively exploiting both temporal and spatial characteristics of the execution traces. The work in [15] introduces a novel anomaly detection method that combines graph convolutional networks with deep residual architectures, augmented by a custom residual-based attention mechanism. As illustrated, the integration of these elements enhances feature representation and model expressiveness, enabling more accurate and efficient anomaly detection in complex networks. Complementing these efforts, authors in [16] and [1] presented ensemble learning strategies for malware classification using system call sequences. Furthermore, with the growing adoption of eXplainable Artificial Intelligence (XAI), recent research has begun integrating XAI methodologies into AI-driven malware detection frameworks to enhance the interpretability and transparency of model decisions, as discussed in [17].

2.2. Graph Neural Network

Graph Neural Networks (GNNs) are a class of deep learning models designed to operate directly on graph-structured data, effectively capturing dependencies between nodes through message passing and neighborhood aggregation [18]. GNNs have shown strong performance in applications such as social network analysis, recommendation systems, and malware detection, where relational information is crucial [19]. Graph Convolutional Networks (GCNs) are a popular subclass of GNNs that generalize convolution operations to graphs, enabling efficient feature extraction from node neighborhoods [20]. GCNs are neural architectures designed to process graph-structured data by aggregating information from a node's local neighborhood [21]. Each GCN layer updates node representations by combining the features of neighboring nodes, allowing the model to capture structural and relational dependencies [22]. Unlike traditional neural networks, GCNs operate directly on graphs, making them suitable for tasks where data are non-Euclidean, such as social networks, molecular structures, and API call sequences [18,20]. GCNs have been effectively applied in malware behavioral analysis, where they capture structural and temporal patterns of API calls to enhance detection accuracy [23].

3. Deep Stacked Graph Convolutional Neural Network with Dynamic Aggregation

The aim of this paper is to introduce a dynamic aggregation of multi-level graph features across stacked Graph Convolutional Neural Networks. As illustrated in Figure 1, our architecture involves three main phases for modeling and classifying malware behavior, which are data preparation, graph construction, and Deep Stacked GCNN-based prediction with dynamic aggregation.

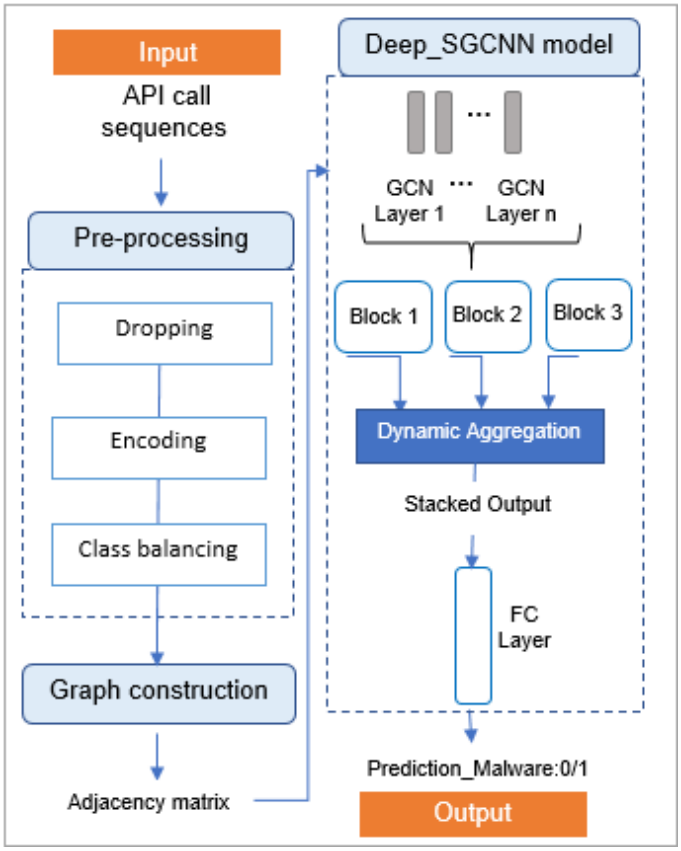


Figure 1. DSGCN architecture using LSTM-based Jumping Knowledge.

The data pre-processing stage is critical for data preparation for effective learning (training and inference). In our architecture, this process consists of three steps: feature selection, data encoding, and class balancing, each of which is described below.

- Feature Selection:
  In this study, we propose to select features by dropping irrelevant columns. For dimensionality reduction and noise elimination, we discarded columns that were redundant, containing null values, or not informative for behavioral analysis. The feature selection step allows us to improve the computational efficiency and enables the model to focus on the relevant patterns in the input data.
- Data Encoding:
  In this step, we used an encoding technique to transform categorical variables into a numerical format that is suitable for deep learning based on graphs. Specifically, the one-hot technique was applied to represent behavior as a binary vector. This input conversion is important for feeding adequate data into the deep stacked GCNN model.
- Class Balancing:
  For robust model training, we used a class balancing technique to address the inherent class imbalance in malware datasets. We applied the Synthetic Minority Over-

sampling Technique (SMOTE) to ensure a balanced class distribution. This technique improves generalization in binary classification settings.

In the behavioral graph construction stage, malware samples are represented as sequences of API calls. This exhibit has both structural and temporal characteristics. Each sample is converted into a directed behavioral graph. The final structure is represented as G = (V, E), shown in Figure 2, where V is the set of nodes (behavioral entities) and E is the set of edges, encoded in an adjacency matrix A. This constructed graph serves as the input structure for the deep stacked GCN layers, enabling behavioral anomaly detection. Malicious sequences often form distinct subgraphs, which the proposed model learns to identify.

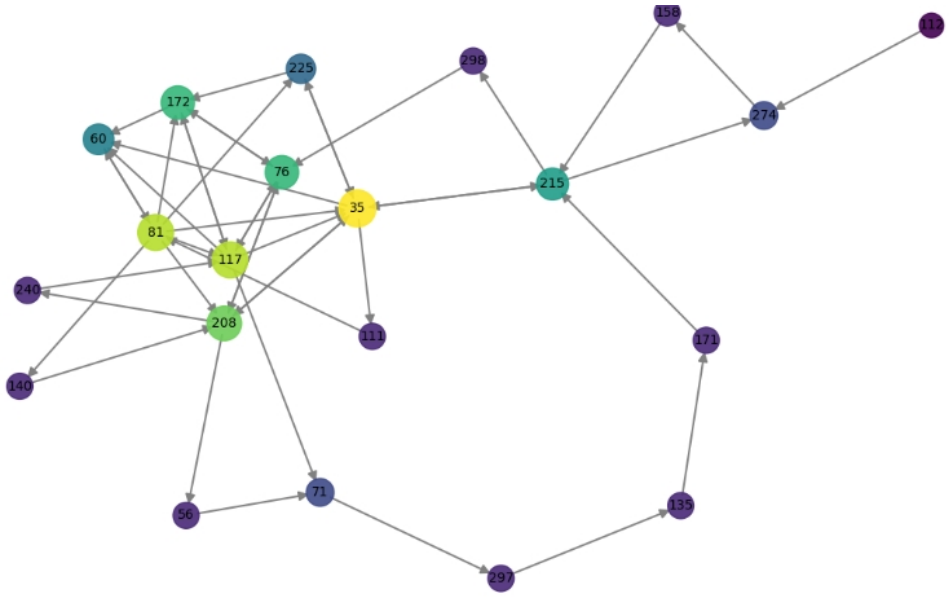


Figure 2. Behavioral graph (21 nodes and 45 edges).

Our proposed anomaly detection approach aims to improve model robustness and learning efficiency using deep stacked GCNs. For this reason, we present an innovative double-level improvement of GCN using layers ensembling and blocks stacking. Firstly, we assemble multiple GCN layers to create deep GCNs; each one represents a block. Then, by stacking three constructed blocks, we obtained our deep stacked GCN. Furthermore, we suggest a dynamic aggregation method that synergizes this improved model with Jumping Knowledge (JK), especially with the LSTM mechanism to aggregate multi-level graph features across stacked GCNN blocks. This allows the DSGCN model to learn how to combine information from different depths (GCN layers/blocks). This LSTM-based Jumping Knowledge (LSTM-JK) method is crucial to capture both local and long-range dependencies in malware behavioral patterns. Figure 3 shows that after each GCN block, we collect its output, which represents a feature map of node embeddings. These feature maps are sequentially fed into an LSTM. This model learns to weigh which layers contain the most useful information for each node (or sequence). Finally, the FC layer integrates all learned patterns into a unified decision (classification).

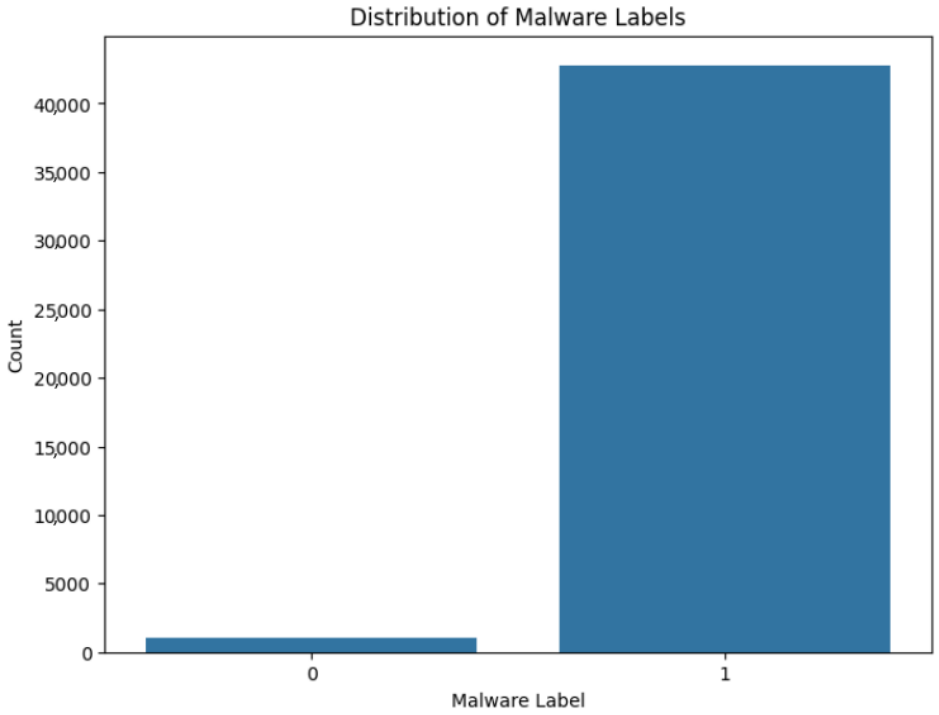


Figure 3. Behavioral graph of API call sequence.

4. Experimentation

4.1. API-Call-Sequences

In our experimentation, we have used the API-Call-Sequences dataset [5,24] for evaluating our proposal. The API-Call-Sequences dataset consisted of 42,797 malicious and 1079 benign API call sequences, and every sequence contains 100 unique. As shown in Figure 3, the used dataset exhibited a significant class imbalance where the malware class (class 1) outnumbered the benign class (class 0) by over 40%. This imbalance has a negative impact on the model performance, particularly on test data. For this reason, we addressed this imbalance by using the SMOTE technique.

4.2. Hyper-Parameters Setting

The performance of our Deep Stacked Graph Convolutional Network with dynamic aggregation, using LSTM-based Jumping Knowledge, depends on the defined hyperparameters that are summarized in Table 1.

Table 1. Hyperparameters of the Deep Stacked GCN with LSTM-based Jumping Knowledge.

Hyperparameter	**Value**
GCN Hidden Dimensions	[64, 48, 32]
Number of GCN Layers	3
Activation Function	ReLU
Dropout Rate	0.5
LSTM Hidden Size (JK)	64
Batch Size	32
Learning Rate	0.0001 (1×10^{-4})
Optimizer	Adam
Training Epochs	30, 50, 100

As described in Table 1, various hyper-parameters, including the architecture depth, hidden dimensions, regularization, and optimization settings, have been defined. This configuration critically shape the experimental results presented in the following section.

4.3. Experimental Results

In this section, we evaluated the efficiency of our proposal described in Section 3. We introduced deep stacked GCN. Instead of treating all GCN layers equally, we used LSTM-JK to learn which layers are more useful for each node by looking at all layers as a sequence. It dynamically decides how much to take from each layer, depending on the information they carry; not all layers contribute equally (see Figure 4). The heatmaps in Figures 5 and 6 compare node activation patterns across three GCN layers using deep GCN and DSGCN-DA (with JK-LSTM aggregation) models. The second model shows stronger and more focused activations on key nodes, indicating improved feature aggregation and class separation.



Figure 4. Node activation with vs. without JK-LSTM.

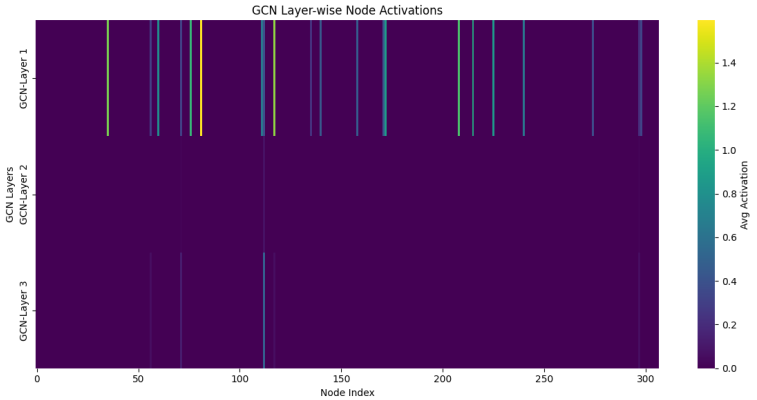


Figure 5. Node activation-layer without JK-LSTM.

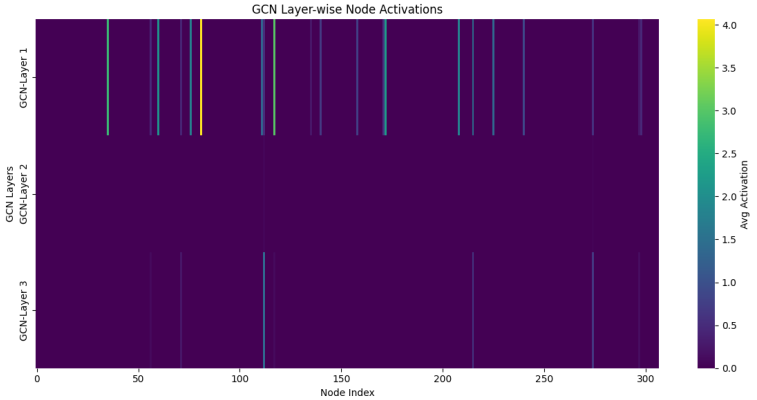


Figure 6. Node activation-layer with JK-LSTM.

The side-by-side simulation in Figure 4 shows how the Jumping Knowledge (JK) mechanism with LSTM (JK-LSTM) can make the top 10 nodes stand out more strongly, while without JK-LSTM the peaks are weaker and less distinct. This mechanism learns which layers' outputs are most informative for each node. In addition, it dynamically integrates the information from all GCN layers instead of relying solely on the final layer's features. As mentioned by [25], focusing on key nodes improves the model's ability to separate classes, leading to better generalization and higher accuracy on unseen data. The subsequent figures demonstrate this effect through standard performance metrics such as accuracy, loss, recall, precision, F1-score, and ROC-AUC. We will present the experimental results, starting from the model training phase and progressing to final classification. We evaluate both the original deep GCN and our proposed Deep Stacked GCN with Dynamic Aggregation (DSGCN-DA) on balanced API-Call-Sequences dataset.

Both figures (Figures 7 and 8) demonstrate that the models exhibit similar learning patterns with rapid initial improvements in both loss reduction and accuracy augmentation, followed by slower progress as the models approach their optimal performance. However, the GCN model appears to achieve slightly lower final accuracy compared to the DSGCN-DA model.

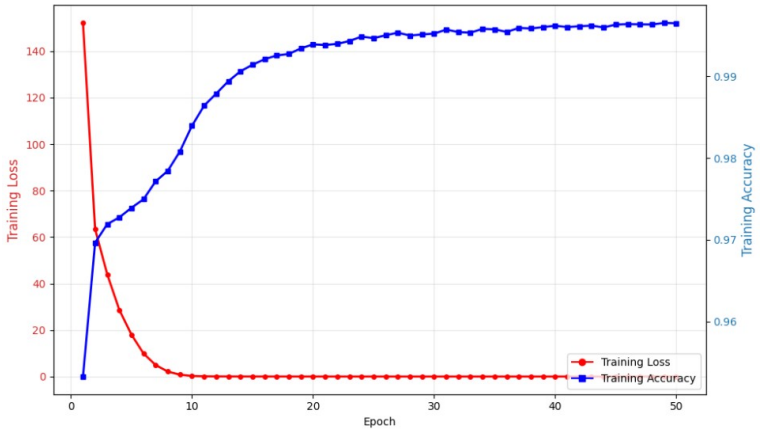


Figure 7. Deep GCN Model training performance with balanced Data.

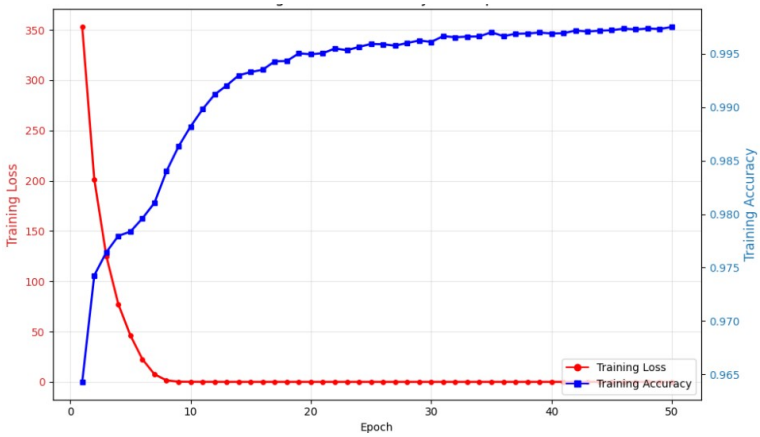


Figure 8. Deep Stacked GCN-based JK-LSTM model training performance.

The training accuracy (blue line) in the first figure reaches approximately 99.65% by the end of training. However, the second model reaches approximately 99.75%. Both models demonstrate strong performance in identifying the majority class (class 1), as evidenced by the high true positive counts. As shown in Figures 9 and 10, the DSGCN-DA model exhibits better performance in minimizing errors (false positives and false negatives) for the minority class (class 0). The datasets are highly imbalanced, which significantly influences

the models' performance metrics. After examining the training dynamics, we now assess the models' performance on the test set using confusion matrices to evaluate generalization and misclassification patterns.

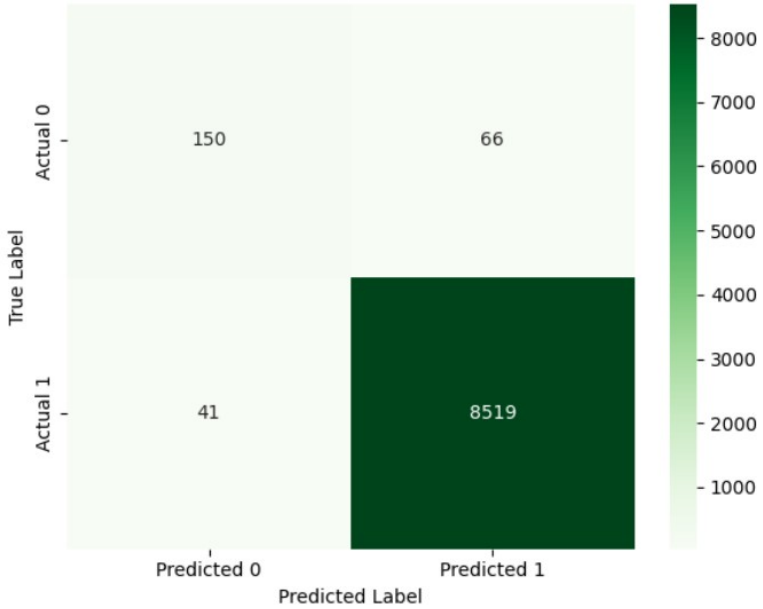


Figure 9. Deep GCN model performance on test dataset.

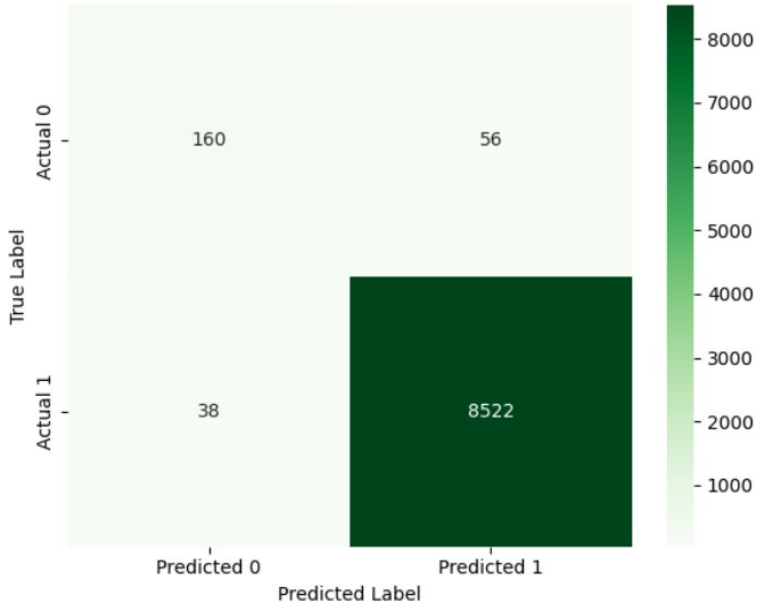


Figure 10. DSGCN-DA model performance on test dataset.

Unlike the stacked GCN, LSTM-JK dynamically learns which layer contains the most informative features for each node across the sequence. As a result, this mechanism plays a crucial role in improving model accuracy (see Figures 9 and 10). The performance comparison between the Deep GCN and our proposed Deep Stacked GCN-LSTM-JK model shows a consistent improvement across all evaluation metrics, as indicated in Table 2. Our model achieves slightly higher accuracy (98.93% vs. 98.78%), precision (99.35% vs. 99.23%), recall (99.56% vs. 99.52%), and F1-score (99.45% vs. 99.38%), indicating better overall classification performance. Notably, the ROC AUC increases from 84.48% to 86.82%, suggesting enhanced discriminative ability between classes, particularly in handling imbalanced datasets.

Table 2. Performance comparison between Deep GCN and Deep Stacked GCN-LSTM-JK model.

Metric	Deep GCN	Our Deep Stacked GCN-LSTM-JK
Accuracy	98.78%	98.93%
Precision	99.23%	99.35%
Recall	99.52%	99.56%
F1-Score	99.38%	99.45%
ROC AUC	84.48%	86.82%

5. Performance Evaluation

To assess the effectiveness of the proposed DSGCN-DA model, we conducted a comprehensive evaluation using benchmark malware datasets, comparing its performance against state-of-the-art models. The highest accuracy is achieved by the LSTM model at 99.43% [17], indicating it as the best-performing model. Attention-based models (99.32%), Multi-Head Attention (98.99%), and BiLSTM/GRU (98.91%) [17] closely follow, demonstrating comparable performance. Models from 2023, including Model-1 (91.05%) and Model-2 (92.44%) [5], LSTM (84.88%), Dummy (47.38%) [5], and DGCNN (84%) [23], exhibit significantly lower accuracies, highlighting the clear improvement achieved by more recent architectures. The proposed DSGCN-DA model achieves 98.93%, slightly below the top-performing models but still highly competitive relative to earlier approaches. As shown in Table 3, the proposed DSGCN-DA model demonstrates substantially higher efficiency than the standard DGCNN [23], achieving 98.93% accuracy compared to DGCNN's 84%, reflecting its superior capability in capturing both structural and temporal malware behaviors. Furthermore, the models in [16,17], including LSTM, Attention, and Multi-Head Attention, achieve the highest accuracy in their study, which proposes an eXplainable Artificial Intelligence (XAI) framework, highlighting the effectiveness of combining sequence modeling with interpretability.

Table 3. Performance Performance evaluation.

Work	ML Model	Year	Accuracy (%)
[5]	Model-1	2023	91.05
	Model-2	2023	92.44
	LSTM	2023	84.88
	Dummy	2023	47.38
[23]	DGCNN	2023	84.00
[17]	LSTM	2024	99.43
	BiLSTM	2024	98.91
	GRU	2024	98.91
	Attention	2024	99.32
	Multi-Head Attention	2024	98.99
	Our DSGCN-DA	2025	98.93

6. Conclusions

In this study, we propose a stacked Deep Graph Convolutional Network (GCN) integrated with a Jumping Knowledge (JK)-LSTM module to model malware behavior from API call sequences. The proposed DSGCN-DA model effectively captures both the structural and temporal patterns of malware, enabling robust representation learning. Experimental results demonstrate that our model achieves high accuracy (98.93%), outperforming traditional graph-based models such as DGCNN while remaining competitive with state-of-the-art sequence-based models. This highlights the efficiency of combining deep graph

convolution with temporal aggregation for malware classification. Moreover, the model's design offers potential interpretability advantages using the JK-LSTM mechanism to focus on important nodes across layers. Despite these strengths, some XAI-based malware detection models achieve higher performance than our proposed model. In future work, we plan to integrate eXplainable Artificial Intelligence (XAI) tools into the AI-based malware detection process to make the results more interpretable and understandable. In addition, we will explore optimization strategies and model compression techniques to improve efficiency while maintaining accuracy, as well as extending the approach to more diverse malware datasets.

Author Contributions: Supervision, E.B. and R.A.; methodology, E.B. and R.A.; writing, G.A. and L.A.; review and editing, E.B. and R.A.; visualization, G.A. and L.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data can be provided on request.

Acknowledgments: The authors express their sincere gratitude and appreciation to Onaizah Colleges, Saudi Arabia, for supporting this research.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Demirkıran, F.; Çayır, A.; Ünal, U.; Dağ, H. An ensemble of pre-trained transformer models for imbalanced multiclass malware classification. *Comput. Secur.* **2022**, *121*, 102846. [CrossRef]
2. Hojjati, H.; Armanfard, N. Dasvdd: Deep autoencoding support vector data descriptor for anomaly detection. *IEEE Trans. Knowl. Data Eng.* **2023**, *36*, 3739–3750. [CrossRef]
3. Catak, F.O.; Yazı, A.F.; Elezaj, O.; Ahmed, J. Deep learning based Sequential model for malware analysis using Windows exe API Calls. *PeerJ Comput. Sci.* **2020**, *6*, e285. [CrossRef] [PubMed]
4. Jiang, W.; Liu, Y.; Zhang, H. SSM-GCN: An FPGA-based efficient GCN accelerator for symmetric sparse matrices. *J. Eng. Res.* **2025**, *14*, 712–721. [CrossRef]
5. De Oliveira, A.S.; Sassi, R.J. Behavioral malware detection using deep graph convolutional neural networks. *Authorea Prepr.* 2023. [CrossRef]
6. Gibert, D.; Mateu, C.; Planes, J. The rise of machine learning for detection and classification of malware: Research developments, trends and challenges. *J. Netw. Comput. Appl.* **2020**, *153*, 102526. [CrossRef]
7. Maniriho, P.; Mahmood, A.N.; Chowdhury, M.J.M. API-MalDetect: Automated malware detection framework for windows based on API calls and deep learning techniques. *J. Netw. Comput. Appl.* **2023**, *218*, 103704.
8. Gao, X.; Hu, C.; Shan, C.; Liu, B.; Niu, Z.; Xie, H. Malware classification for the cloud via semi-supervised transfer learning. *J. Inf. Secur. Appl.* **2020**, *55*, 102661. [CrossRef]
9. Mohaisen, A.; Alrawi, O.; Mohaisen, M. AMAL: High-fidelity, behavior-based automated malware analysis and classification. *Comput. Secur.* **2015**, *52*, 251–266.
10. Bidoki, S.M.; Jalili, S.; Tajoddin, A. PbMMD: A novel policy based multi-process malware detection. *Eng. Appl. Artif. Intell.* **2017**, *60*, 57–70.
11. Gaber, M.G.; Ahmed, M.; Janicke, H. Malware detection with artificial intelligence: A systematic literature review. *ACM Comput. Surv.* **2024**, *56*, 148 . [CrossRef]
12. Jeon, S.; Moon, J. Malware-detection method with a convolutional recurrent neural network using opcode sequences. *Inf. Sci.* **2020**, *535*, 1–15. [CrossRef]
13. Marques, P.; Rhode, M.; Gashi, I. Waste not: Using diverse neural networks from hyperparameter search for improved malware detection. *Comput. Secur.* **2021**, *108*, 102339. [CrossRef]
14. Liu, Y.; Tantithamthavorn, C.; Li, L.; Liu, Y. Deep learning for android malware defenses: A systematic literature review. *ACM Comput. Surv.* **2022**, *55*, 153. [CrossRef]

15. Khan, W.; Mohd, A.; Suaib, M.; Ishrat, M.; Shaikh, A.A.; Faisal, S.M. Residual-enhanced graph convolutional networks with hypersphere mapping for anomaly detection in attributed networks. *Data Sci. Manag.* **2024**, 8, 137–146.
16. Bhat, P.; Behal, S.; Dutta, K. A system call-based android malware detection approach with homogeneous & heterogeneous ensemble machine learning. *Comput. Secur.* **2023**, *130*, 103277. [CrossRef]
17. Galli, A.; La Gatta, V.; Moscato, V.; Postiglione, M.; Sperlì, G. Explainability in AI-based behavioral malware detection systems. *Comput. Secur.* **2024**, *141*, 103842. [CrossRef]
18. Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; Yu, P.S. A Comprehensive Survey on Graph Neural Networks *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *32*, 4–24. . . [CrossRef] [PubMed]
19. Waikhom, L.; Patgiri, R. A survey of graph neural networks in various learning paradigms: methods, applications, and challenges. *Artif. Intell. Rev.* **2023**, 56, *6295–6364.*
20. Kipf, T.N.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. In Proceedings of the International Conference on Learning Representations (ICLR), Toulon, France, 24–26 April 2017.
21. Zhang, Z.; Cui, P.; Zhu, W. Deep Learning on Graphs: A Survey. *IEEE Trans. Knowl. Data Eng.* **2022**, *34*,249–270 .
22. Luo, H.; Wang, Y.; Zhang, L.; Chen, W.; Liu, X. Behavioral Graph Representation for Malware Detection Based on API Call Sequences. *Comput. Secur.* **2024**, *133*, 103330.
23. Owoh, N.; Adejoh, J.; Hosseinzadeh, S.; Ashawa, M.; Osamor, J.; Qureshi, A. Malware Detection Based on API Call Sequence Analysis. *Future Internet* **2023**, *16*, 369. [CrossRef]
24. Angelo, O. Malware Analysis Datasets: API Call Sequences. *IEEE Dataport*; IEEE: Piscataway, NJ, USA, 2019.
25. Xu, K.; Li, C.; Tian, Y.; Sonobe, T.; Kawarabayashi, K.i.; Jegelka, S. Representation learning on graphs with jumping knowledge networks. In Proceedings of the International Conference on Machine Learning,Stockholm, Sweden, 10–15 July 2018; PMLR; pp. 5453–5462.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.