Malware Detection Using Dynamic Graph Neural Networks

Pushkaraj Kulkarni and Stephen OShaughnessy
School of Informatics and Cyber Security, Technological University Dublin, Ireland

pushkaraj1234@gmail.com
stephen.OShaughnessy@tudublin.ie

Abstract: The increasing complexity and sophistication of malware pose significant challenges to traditional detection techniques. Conventional methods like signature-based detection are ineffective against advanced threats such as polymorphic and zero-day malware. This research investigates the application of Dynamic Graph Neural Networks (DGNNs) for malware detection using a dataset of API call sequences. DGNNs, an advanced form of Graph Neural Networks, are capable of modeling dynamic graphs, capturing both the temporal and structural evolution of API interactions. Using these strengths, the study develops and evaluates a DGNN-based framework designed to effectively distinguish between benign and malicious behavior in real time, demonstrating its suitability for detecting complex, evolving malware patterns. The results show that DGNN outperform traditional machine learning models in detecting complex malware patterns, achieving high accuracy of up to 97%, F1 scores of up to 98% in unbalanced datasets, and competitive results in balanced datasets. The models also achieved ROC-AUC scores exceeding 97% in specific configurations, highlighting their effectiveness in identifying advanced malware pat- terns and resilience against novel threats. Although challenges in scalability and computational complexity remain, this work proposes potential solutions to enhance practical implementation. These findings highlight the potential of DGNNs to transform malware detection and significantly improve endpoint security, making them a promising tool for addressing the evolving challenges of modern cybersecurity.

Keywords: Malware detection, API call sequences, Dynamic graph neural networks, Machine learning, Endpoint security

1. Introduction

The rapid evolution and increasing complexity of malware have rendered traditional detection methods insufficient in addressing modern cybersecurity threats. Conventional approaches, such as signature-based and heuristic detection, fail to identify polymorphic and zero-day malware due to their static nature and reliance on pre-existing patterns. Malware authors have adopted advanced techniques, including polymorphism, metamorphism, and sophisticated obfuscation, to bypass these detection systems. This necessitates the development of more dynamic and robust solutions.

This research addresses the critical challenge of detecting malware in dynamic and evolving environments by leveraging Dynamic Graph Neural Networks (DGNNs). DGNNs extend the capabilities of traditional Graph Neural Networks (GNNs) by incorporating the dimension of graph data, enabling the modelling of complex interactions and patterns over time. This unique ability makes DGNNs well-suited for analysing API call sequences, which capture the behavioural characteristics of malware during execution.

The scientific question guiding this research is: *How can Dynamic Graph Neural Networks improve the detection of advanced malware by modelling various system interactions and changes?* Specifically, the study investigates whether DGNNs can outperform existing methods in identifying zero-day and polymorphic malware, while also addressing challenges related to scalability and real-time application.

By proposing a novel DGNN-based framework for malware detection and evaluating it on a real-world dataset of API call sequences, this research contributes to the growing field of machine learning in cybersecurity. The findings are expected to enhance endpoint security and provide insights into the practical deployment of DGNN-based systems, making a significant step toward addressing the limitations of current malware detection technologies.

2. Related Works

The literature review identifies key advances and gaps in the fields of malware detection and dynamic graph learning.

The literature on malware detection has evolved from traditional signature-based methods to more advanced machine learning (ML) approaches. Schofield et al (2021) proposed Signature-based methods are ineffective against evolving malware variants, such as polymorphic and zero-day malware. They proposed a CNN-based model that achieves 98.17% accuracy but highlight limitations in generalizing across diverse malware families. Kim and Chan Woo (2018) mentions behavior-based methods, leveraging API call analysis, show promise but face challenges in scalability and robustness.

Studies by Catak et al. (2020) and Kim and Chan Woo (2018) using Support Vector Machines (SVMs) and Long Short-Term Memory (LSTM) models demonstrate high accuracy in detecting malware based on dynamic API calls. However, these models lack flexibility in modeling temporal and structural relationships.

Research by Kolbitsch et al. (2009) on endpoint security shows that ML and artificial intelligence (AI) can improve threat detection and resilience against adversarial attacks by leveraging supervised and unsupervised models. Behaviour-based approaches focus on analysing deviations in software behaviour to detect unknown malware patterns proactively. While these methods enhance detection, they face challenges such as high false positives, scalability issues, and continuous updates of behaviour profiles.

Several studies explore malware detection using API call sequences. Busch et al. (2021) proposed Graph-based models, including Graph Neural Networks (GNNs) and DGNNs, which have shown promise in analysing dynamic behaviours. These models outperform traditional classifiers by representing executable behaviours through network graphs. GNNs have proven effective in leveraging graph-structured data, such as network traffic, to enhance malware detection. Studies by Moallemy-Oureh et al. (2022) applying flow-based graph models for malware detection show that utilizing network traffic data as flow graphs significantly improves detection performance. The use of edge-based GNN models enhances the classification of malicious behaviors by capturing communication patterns across the network. They introduced Fully Dynamic Graph Neural Networks (FDGNNs) that adapted to evolving graph data, a critical requirement for capturing malware behaviors. However, gaps remain in incorporating diverse data sources and addressing real-world scalability.

Recent advancements in DGNNs by address limitations in handling dynamic graph data. Fully dynamic GNN models by Wang et al. (2021) and time-indexed approaches have been developed to manage changes in graph structures over time. Dynamic graphs integrate temporal, structural, and sequential data, making them suitable for malware detection. Recent surveys emphasize the potential of DGNNs for modeling dynamic systems, though their application in cybersecurity remains limited. Research by limited Wang et al. (2021) , Yang, Chatelain and Adam (2024) and Pareja et al. (2020) highlights the need for efficient training strategies and scalability to improve real-world deployment.

Studies by Moallemy-Oureh et al. (2022) applying flow-based graph models for malware detection show that utilizing network traffic data as flow graphs significantly improves detection performance. The use of edge-based GNN models based on Yang, Chatelain, and Adam (2024) enhances the classification of malicious behaviors by capturing communication patterns across the network. However, challenges remain in optimizing DGNN models for large-scale malware detection. Despite advancements in GNNs by Yang, Chatelain and Adam (2024) and Moallemy-Oureh et al.(2022), there is limited research applying DGNNs to malware detection . Existing studies by Schofield et al. (2021) and Kim (2018) focus primarily on static or behavioral analysis without fully integrating dynamic temporal graphs to detect evolving malware.

This research leverages insights from prior studies to address critical gaps in applying DGNNs to malware detection. By focusing on dynamic API call sequences and utilizing state-of-the-art graph-based learning techniques, the study seeks to develop a scalable and robust detection framework capable of identifying advanced malware behaviors. This contribution not only fills an existing research void but also offers practical implications for enhancing real-world cybersecurity systems.

2.1 Justification for Research

Traditional malware detection methods, such as signature-based approaches, are increasingly in- adequate in identifying polymorphic and zero-day threats. Dynamic malware analysis, leveraging behavioral data such as API call sequences, has demonstrated potential to address these challenges. Recent studies have highlighted the effectiveness of machine learning (ML) and deep learning techniques, particularly GNNs, for modeling and detecting complex malware behaviors. However, while GNNs have been applied to various fields, their extension to DGNNs for malware detection remains largely unexplored. Recent studies highlight the growing importance of machine learning, particularly Dynamic Graph Neural Networks (DGNNs), in enhancing malware detection accuracy by capturing both temporal and structural patterns in malware behaviour. Given the success of GNNs in dynamic domains like social networks and traffic systems, applying DGNNs to malware detection offers an innovative way to capture the temporal and structural evolution of malicious behaviors. This research addresses a significant gap by employing DGNNs to analyze API call sequences, which are crucial for detecting sophisticated malware patterns. By filling this gap, the study contributes to advancing robust, scalable, and real-world-applicable malware detection techniques.

3. Methodology

3.1 Overview

This section presents the methods applied in this thesis, encompassing the research approach, design, and experimental setup and describe the dataset, preprocessing steps, model development of dynamic graphs along with the implementation and also details about hyperparameter optimization and performance evaluation.

3.2 Dataset

The dataset, provided by Oliveira (2019), contains 42,797 malware samples and 1,079 goodware samples. Each sample includes a unique identifier hash, a sequence of 100 API calls t_0 to t_99, and a binary label indicating whether the sample is malware (1) or goodware (0). These data were collected from Portable Executable (PE) files executed in a Cuckoo Sandbox environment, generating JSON logs that detail API calls, network traffic, and dropped files.

After collection, the raw API calls were normalized into ordinal categorical values, ensuring consistency in how each API call is represented. Table 1 summarizes the key columns in the dataset.

Table 1: Summary of Dataset Columns

	Description
hash	Unique identifier for each sample, used to trace or verify the specific file.
t 0 to t 99	Sequential API calls made during execution, where each entry is a numerical code corresponding to a specific API call.
malware	Binary label indicating whether a sample is malware (1) or good- ware (0).

3.3 Preprocessing and Feature Extraction

Preprocessing focused on normalizing the API calls and ensuring variability in the data. API call sequences were converted into high-dimensional matrices to capture the frequency and transitions of each call. To reduce dimensionality while retaining critical information, these matrices were grouped by API operation types. Key features extracted were:

- API call sequence length
- Most frequent API calls
- Unique API call counts
- Transition patterns among calls

By leveraging these sequence features, the DGNN models can learn abnormal and malicious behaviors without manual feature selection. Each sample's API sequence is represented as a dynamic graph, where nodes represent API calls and edges represent transitions between consecutive API calls. This approach captures temporal information and reflects the evolving nature of malware behavior. Early-stage behaviors, derived from parent processes, are prioritized to emphasize malicious patterns.

3.4 Model Development and Evaluation

Two DGNN-based models were implemented to classify samples as malware or goodware. Hyperparameters (e.g., learning rate, number of hidden units) were optimized to maximize predictive performance. The models were evaluated using accuracy, precision, recall, F1-score, and the area under the ROC curve (AUC) to assess model effectiveness.

To ensure robust evaluation of the models, the dataset was split into training and testing subsets using a 70-30 split, a common best practice in machine learning. The train_test_split function from sklearn was employed with a fixed random seed for reproducibility. This split ensured the model was trained on a diverse set of data, to optimize generalization and a separate testing set for evaluating performance on unseen data.

3.4.1 Adapted DGNN Architecture

The primary model in this study is a Dynamic Graph Neural Network (DGNN), which captures temporal and structural relationships in the malware behavior data. The DGNN was adapted from two open-source

repositories Zhang et al. (2022); Fu and He (2021) originally designed for Sequential Recommendation Systems. Modifications included:

- Transforming each API call sequence into a graph structure.
- Using graph convolutional layers (GCNs) to extract spatio-temporal relationships.
- Integrating dropout layers to mitigate overfitting during training.
- Feeding the final GCN outputs into fully connected layers and a sigmoid binary classifier for malware probability estimates.

Two versions of the DGNN model were developed:

- Model 1: Trained on the original, imbalanced dataset.
- Model 2: Trained on a balanced dataset using an undersampling technique to address class imbalance.

Both models were trained using 5-fold cross-validation k=5 to improve generalization and reduce overfitting.

3.4.2 Hyperparameter optimization

Hyperparameters (learning rate, batch size, dropout rate, number of epochs, etc.) were fine-tuned using GridSearchCV to ensure optimal performance. For Model 2, additional parameters such as weight_dim_2 and weight_dim_4 were included to expand the search space. The exhaustive grid search helped identify the best configuration, reducing overfitting and improving accuracy.

3.4.3 Model architecture details

API call sequences were converted into graph representations using adjacency and degree matrices. Multiple GCN layers were stacked to capture multi-scale substructures in the data. The final learned representations were fed into a fully connected layer followed by a sigmoid binary classifier to distinguish between malware and goodware.

Table 2 compares the core characteristics of the two DGNN models:

Table 2: Comparison of Model 1 and Model 2

Parameter	Model 1	Model 2	
DGNN layers	1	2	
Hyperparameters	Basic	Advanced	using
		GridSearchCV	
Graph learning	Single-level	Hierarchical	
Performance	High accuracy	Improved accuracy	

3.4.4 Implementation pipeline

The models were integrated into a pipeline optimized with NeuralNetBinaryClassifier, ensuring compatibility with scikit-learn workflows. Key steps included:

1. Data loading and preprocessing of API call sequences.
2. Graph construction (using adjacency and degree matrices).
3. Input encoding for neural network compatibility.
4. Stacking GCN layers and applying dropout.
5. Fine-tuning hyperparameters via GridSearchCV.
6. Model evaluation on the separate test set.

3.5 Summary

This methodology integrates advanced machine learning techniques, graph-based modeling, and dynamic analysis to address critical challenges in malware detection. By leveraging DGNNs, the approach captures the evolving nature of threats, providing a scalable solution for real-world cybersecurity applications. The combination of dynamic graph construction, careful handling of imbalanced data, and thorough hyperparameter optimization positions this research at the forefront of malware detection technology.

4. Results

The results of this research provide a comprehensive evaluation of the proposed models, showcasing their effectiveness in detecting malware using Dynamic Graph Neural Networks (DGNNs). The models were evaluated under two distinct training strategies: one using the original imbalanced dataset and another utilizing a balanced dataset created through random undersampling. Both strategies demonstrate the strengths and challenges of implementing DGNNs in scenarios with varying class distributions.

For the original imbalanced dataset, the first model achieved an accuracy of 98.85%, precision of 97.97%, recall of 98.85%, F1-Score of 99.10%, and ROC-AUC score of 95.87%. These results highlight the model's ability to accurately classify malware samples, with minimal false positives and false negatives. The high F1-Score indicates a strong balance between precision and recall, while the ROC-AUC score reflects the model's robust discriminatory power across varying thresholds. However, in scenarios involving imbalanced datasets, the accuracy metric can be misleading. The emphasis on precision, recall, and F1-Score provides a more nuanced understanding of the model's effectiveness in handling the minority class, ensuring that both false positives and false negatives are minimized.

In the case of the balanced dataset, the first model's performance adjusted slightly, reflecting the impact of data balancing techniques. The model achieved an accuracy of 90.08%, a precision of 89.01%, a recall of 92.46%, an F1-Score of 90.15%, and a ROC-AUC score of 97.40%. While the accuracy decreased compared to the imbalanced dataset, the other metrics indicate a well-rounded performance. Precision and recall balance the trade-offs between false positives and false negatives, making the model effective in scenarios where both types of errors carry significant risks. The ROC-AUC score demonstrates the model's strong ability to differentiate between malware and benign samples across thresholds, emphasizing its reliability for varying classification boundaries.

The second training strategy involved a more advanced implementation of the DGNN model, which refined the architecture and utilized additional hyperparameter tuning. For the imbalanced dataset, this model achieved a 97.53% accuracy, 98.01% precision, 99% recall, 98.14% F1-Score, and a 96.20% ROC-AUC score. These results the model's exceptional ability to capture true positive cases with minimal false positives. The high precision and recall balance reinforce the reliability of the model in identifying malware effectively, even when faced with imbalanced data distributions.

When trained on the balanced dataset, the second model delivered a 91.30% accuracy, 93.21% precision, 92% recall, 91.89% F1-Score, and an impressive 98.02% ROC-AUC score. The high precision and recall highlight the model's effectiveness in mitigating false positives and capturing true positives. The F1-Score further emphasizes the balance achieved between these two critical metrics, ensuring robust performance in detecting malware. The ROC-AUC score, nearing perfection, indicates the model's exceptional capability in distinguishing between classes under varying thresholds.

Overall, the results demonstrate that both models effectively leverage the strengths of DGNNs in capturing the dynamic and temporal characteristics of malware behavior. The nuanced use of performance metrics such as precision, recall, F1-Score, and ROC-AUC ensures that the evaluation reflects the models' real-world applicability, particularly in handling imbalanced datasets. The findings validate the potential of DGNNs as a powerful tool in malware detection, offering a scalable and robust approach to addressing evolving cybersecurity challenges.

5. Discussion

5.1 Result Interpretation

This research investigates whether patterns in API call datasets can be effectively leveraged using Dynamic Graph Neural Networks (DGNNs) to enhance malware detection. The first model, built with a single DGNN layer and trained on an imbalanced dataset, showcased strong performance metrics such as high precision, recall, F1-Score, and ROC-AUC. These results highlight the model's ability to accurately classify malware while minimizing false positives, demonstrating its robustness in handling imbalanced datasets. The high F1-Score confirmed a balance between precision and recall, ensuring that the model avoided overemphasis on either metric. Furthermore, the ROC-AUC score highlighted the model's capability to distinguish between malware and benign instances across varying thresholds. Despite the dataset imbalance, the model maintained unbiased performance, indicating its reliability and suitability for real-world deployment scenarios where imbalanced datasets are common.

When the same model was trained on a balanced dataset created through undersampling, its metrics remained strong, albeit with some minor adjustments. The balanced dataset allowed the model to exhibit high accuracy and a well-maintained balance between precision and recall. The F1-Score of 90.15% illustrated its consistent ability to minimize false positives and negatives, while the ROC-AUC of 97.40% reinforced its ability to distinguish between the two classes effectively. Although precision slightly declined compared to recall, indicating a marginal increase in false positives, this result could be optimized further through fine-tuning. The balanced dataset provided a clearer interpretation of the model's capabilities, ensuring that its performance was not skewed by the class distribution.

The second model, enhanced with an additional DGNN layer and tuned hyperparameters, delivered even more impressive results on the imbalanced dataset. Metrics such as nearly perfect recall and a high precision score underscored its effectiveness in identifying true positives while minimizing false positives. The F1-Score and ROC-AUC further validated its superior performance, indicating the model's reliability in distinguishing between malware and benign files even under challenging conditions. However, when trained on a balanced dataset, the second model exhibited slightly lower accuracy but retained robust precision, recall, and ROC-AUC metrics. This outcome aligns with expectations, as balancing datasets often reduces overall accuracy but ensures more equitable attention to both classes. The model’s ability to maintain high ROC-AUC and F1-Score across different data distributions affirmed its adaptability and reliability for cybersecurity applications.

5.2 Comparison with Existing Literature

This research contributes significantly to the field of malware detection by addressing limitations observed in traditional machine learning approaches such as Support Vector Machines, Decision Trees, and Random Forests. These conventional models often struggle with the class imbalance prevalent in cybersecurity datasets, where benign instances far outnumber malicious ones, resulting in high false-positive rates. By employing DGNNs, the study demonstrated superior performance in managing such imbalances, achieving an accuracy of 98.87% and a precision of 99% on the imbalanced dataset. This highlights the effectiveness of deep learning models, particularly those leveraging graph-based structures, in overcoming the challenges faced by traditional methods.

Moreover, the study builds on prior work in sequential and graph-based neural networks, particularly the advancements in Dynamic Graph Neural Networks as proposed by researchers such as Moallemy-Oureh et al. (2022) and Pareja et al. By incorporating these dynamic layers, the current research enhanced the model's ability to capture and interpret the interactions between API calls, leading to high-performance metrics like a ROC-AUC of 0.96 on the imbalanced dataset. This underscores the potential of DGNNs in understanding the complex and evolving nature of malware behavior.

When evaluated on a balanced dataset, the model's accuracy was slightly lower at 91.30%, with a precision of 93.21%. These results align with observations from previous studies, such as Zhang et al. (2022), which noted that balancing datasets often results in a trade-off between accuracy and the model’s focus on both classes. Despite this, the model demonstrated improved generalizability and robustness, making it more effective for real-world applications where balanced datasets may better represent operational conditions. Overall, this research reinforces the value of DGNNs in advancing malware detection techniques and lays the groundwork for further exploration of graph-based learning in cybersecurity.

6. Conclusion

6.1 Summary

This study investigated the application of Dynamic Graph Neural Networks (DGNNs) for detecting malware in dynamic and evolving cybersecurity environments. The research aimed to address two primary objectives: assessing the capability of DGNNs to identify malware behavior through API call datasets and comparing their performance with traditional machine learning methods. A model architecture leveraging DGNN layers was developed and tested under different configurations to explore these objectives. The results confirmed that DGNNs could effectively identify patterns in malware behavior by analyzing relationships between API calls, demonstrating significantly higher accuracy and robustness than conventional approaches. This performance was consistent across both imbalanced and balanced datasets, highlighting DGNNs' adaptability and utility in real-world cybersecurity scenarios. Additionally, enhancements to the architecture and training methodologies improved model interpretability and operational viability. Overall, this study provided strong evidence supporting the potential of DGNNs as a scalable and reliable tool for combating dynamic and evolving malware threats.

6.2 Contributions

This research presented a novel approach to malware detection by employing DGNNs, which are particularly suited for addressing the challenges posed by zero-day and polymorphic malware. Unlike traditional methods, which often struggle with imbalanced datasets and lack the capacity to capture dynamic interactions within systems, DGNNs demonstrated an enhanced ability to adapt to these challenges. The study confirmed that DGNNs surpass traditional methods in accuracy, precision, and robustness, especially in detecting constantly evolving malware. By bridging the gap between theoretical advancements and practical applications, the study showcased the real-world feasibility of DGNN-based solutions for malware detection. The proposed framework not only modeled system interactions effectively but also proved scalable and adaptable to modern cybersecurity challenges. Furthermore, the research identified practical solutions for implementing DGNNs, making them relevant for deployment in diverse operational scenarios. This work underscores the critical role DGNNs can play in advancing cybersecurity, offering a promising avenue for further exploration and refinement in addressing increasingly sophisticated cyber threats.

6.3 Limitations and Future Work

While this research advanced the field of malware detection, it also highlighted several limitations. The computational demands of training and applying DGNNs, particularly for real-time detection, remain a significant challenge, especially in resource-constrained environments. The reliance on labeled data for training also limits the model's generalizability, as acquiring labeled datasets for new or polymorphic malware can be difficult. Additionally, like other deep learning methods, DGNNs suffer from interpretability issues, which could impede their adoption in critical security systems where decision-making transparency is paramount.

The evaluation conducted in this study was based on specific API call sequences and offline conditions, leaving the performance of the DGNN model in real-time, online environments largely unexplored. Furthermore, the research focused on binary classification (malware vs. benign), while real-world scenarios often require multi-class classification to handle various malware families. Another limitation involves potential vulnerabilities of DGNN architectures to adversarial attacks, where malicious inputs could compromise the model's reliability. Future work should address these challenges by exploring ways to improve the interpretability, scalability, and robustness of DGNNs. Research should also extend to evaluating the models in diverse and complex network environments, investigating their integration into existing cybersecurity frameworks, and developing strategies to mitigate adversarial threats. These advancements will help ensure that DGNN-based solutions can effectively meet the evolving demands of modern malware landscapes.

References

Busch, J., Kocheturov, A., Tresp, V. and Seidl, T. (2021). NF-GNN: network flow graph neural networks for malware detection and classification. pp.121–132.

Catak, Ferhat Ozgur, Yazı, Ahmet Faruk, Elezaj, O. and Ahmed, J. (2020). Deep learning based Sequential model for malware analysis using Windows exe API Calls. *PeerJ computer science*, 6, p.e285.

Feng, Z., Wang, R., Wang, T., Song, M., Wu, S. and He, S. (2024). *A comprehensive survey of dynamic graph neural networks: Models, frameworks, benchmarks, experiments and challenges*. *arXiv preprint arXiv:2405.00476*.

Fu, D. and He, J. (2021). SDG: A simplified and dynamic graph neural network. [online] ACM, pp.2273–2277. doi: https://doi.org/10.1145/3404835.3463059.

Kim, C.W. (2018). *Ntmaldetect: A machine learning approach to malware detection using native api system calls*. *arXiv preprint arXiv:1802.05412*.

Kolbitsch, C., Milani, C.P., Kruegel, C., Kirda, E., Zhou, X. and Wang, X. (2009). Effective and efficient malware detection at the end host. pp.351–366.

Moallemy-Oureh, A., Beddar-Wiesing, S., Nather, R. and Thomas, J.M. (2022). *Fdgnn: Fully dynamic graph neural network*. *arXiv preprint arXiv:2206.03469*.

Oliveira, A. (2019). Malware analysis datasets: API call sequences. [online] doi: https://doi.org/10.21227/tqqm-aq14.

Pareja, A., Domeniconi, G., Chen, J., Ma, T., Suzumura, T., Kanezashi, H., Kaler, T., Schardl, T. and Leiserson, C. (2020). Evolvegcn: Evolving graph convolutional networks for dynamic graphs. pp.5363–5370.

Schofield, M., Alicioglu, G., Binaco, R., Turner, P., Thatcher, C., Lam, A. and Sun, B. (2021). Convolutional neural network for malware classification based on API call sequence. pp.23–24.

Wang, P.Y., Sapra, S., George, V.K. and Silva, G.A. (2021). Generalizable machine learning in neuroscience using graph neural networks. *Frontiers in artificial intelligence*, 4, p.618372.

Yang, L., Chatelain, C. and Adam, S. (2024). Dynamic graph representation learning with neural networks: A survey. *IEEE Access*, 12, pp.43460–43484.

Zhang, M., Wu, S., Yu, X., Liu, Q. and Wang, L. (2022). Dynamic graph neural networks for sequential recommendation. *IEEE Transactions on Knowledge and Data Engineering*, pp.1–1. doi: https://doi.org/10.1109/TKDE.2022.3151618.