

Graph Neural Networks for Malware and Financial Fraud Detection

T Ramaprabha

Associate Professor, Department of Computer Science, Nehru Arts and Science College, Coimbatore, India

Article information

Received: 27th May 2026

Received in revised form: 30th June 2026

Accepted: 31st July 2026

Available online: 12th September 2026

Volume: 1

Issue: 3

DOI: <https://doi.org/10.5281/zenodo.22701803>

Abstract

Malware and payment fraud share a structural feature that flat feature vectors throw away. Both leave behind a graph. A binary induces a call graph; a fraud ring induces a web of accounts, devices and shared attributes. Graph neural networks learn on that structure directly, and this review examines what the move has bought in each domain. We cover thirty two works. The modelling machinery turns out to be common to both problems, while the graph construction step, which is where most of the engineering effort goes, is not. Three obstacles recur and none is solved. Fraud graphs are extremely imbalanced, so the majority class dominates the message passing. Adversaries adapt, and camouflage that connects a fraudulent node to legitimate neighbours defeats naive aggregation. And evaluation protocols that ignore time leak future information into training, which inflates reported accuracy in ways that field deployment does not reproduce. We set out what a defensible evaluation looks like and where the research gaps sit.

Keywords:- Graph Neural Network, Malware Detection, Fraud Detection, Call Graph, Message Passing, Class Imbalance, Concept Drift.

I. INTRODUCTION

Two detection problems, two industries, one shape. A malware analyst looks at a binary and asks which functions call which. A fraud analyst looks at a payment stream and asks which accounts share a device, a card, or an address. Both questions are about relations, and relations form graphs.

Traditional pipelines flattened those graphs into feature vectors. Ye and colleagues catalogued the malware side of that practice [8]; Bhattacharyya and colleagues did the same for card fraud [2]. Flattening works, up to a point. It also discards exactly the information an analyst uses when reasoning by hand, which is that this account is two hops from a known mule, or that this function calls the same encryption routine as a known dropper.

Graph neural networks remove the need to flatten. A node's representation is built from its neighbours, then from theirs, and so on for a fixed number of rounds. Kipf and Welling gave the widely used convolutional formulation [5], Hamilton and colleagues made it inductive so unseen nodes could be handled [6], Velickovic and colleagues added attention over neighbours [10], and Gilmer and colleagues showed that these are all instances of one message passing scheme [7]. This review asks how well that machinery serves two adversarial detection problems, and what breaks when it is applied carelessly.

II. FROM GRAPH TO PREDICTION

Message passing has three steps and they are worth stating plainly. Each node gathers vectors from its neighbours. It aggregates them with a permutation invariant function. It combines the result with its own current state. Repeat k times and every node has seen its k -hop neighbourhood. Figure 2 shows one round.

The aggregation function is where the models differ. Mean aggregation is smooth and loses count information. Sum aggregation keeps it, which is why Xu and colleagues showed sum based models to be more expressive in the sense of graph isomorphism testing [13]. Attention weights neighbours unequally [10], which matters when most neighbours are noise. Two surveys cover the design space in detail [18], [21].

Detection tasks come in two shapes. Malware detection is usually graph level: one binary, one label. Fraud detection is usually node level: one account, one label, inside a graph that never stops growing. Figure 1 shows both pipelines side by side. The modelling step is shared. The construction step is not, and it is where most practical difficulty lives.

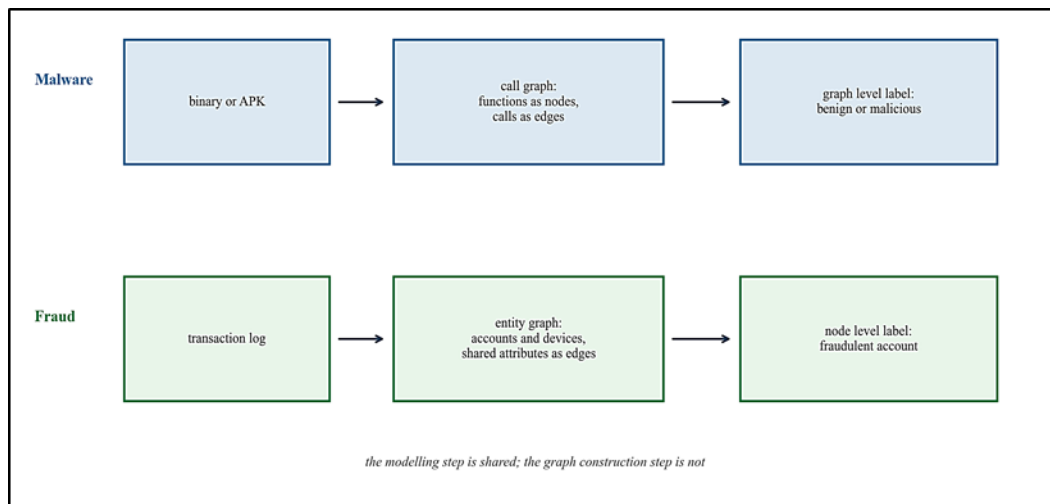


Figure. 1: Two detection pipelines. The graph construction step differs by domain; the learning step does not.

III. GRAPHS FROM BINARIES

Call graph analysis for malware predates deep learning. Hu and colleagues indexed malware by function call graph and retrieved near duplicates at scale [1]. Gascon and colleagues embedded android call graphs and classified them with a linear model [3]. DREBIN took a different route, using lightweight static features and staying explainable, and it remains a reference dataset [4]. MaMaDroid abstracted call sequences into Markov chains over packages, which gave it more resilience to small changes in the code [9].

Yan and colleagues brought graph convolution to the control flow graph directly, learning on the structure rather than on hand designed summaries of it [15]. Hei and colleagues went further and built a heterogeneous graph over apps, permissions, APIs and other entity types, then applied attention across relation types [29]. Heterogeneity matters here: an edge meaning calls and an edge meaning shares a certificate carry different evidence, and the model should not average them [14].

Two dataset problems shadow this work. Labels age, because malware families evolve, and the MalNet collection was assembled partly to give the field a large public graph resource with family labels [26]. Evaluation protocols also age badly, which is the subject of Section V. Ransomware in particular has moved fast enough that detection research struggles to keep pace with the operational picture [31].

IV. GRAPHS FROM TRANSACTIONS

Fraud graphs are built from entities and the attributes they share. Liu and colleagues modelled accounts, devices and IP addresses as a heterogeneous graph and caught malicious registrations that looked innocent one at a time [11]. Wang and colleagues added attention in a semi supervised setting, which suits fraud because labels arrive late and in small numbers [16].

Then the adversary adapted. Fraudsters connect their accounts to legitimate ones on purpose, so a fraudulent node sits in a mostly benign neighbourhood and averaging over neighbours washes out the signal. Dou and colleagues named this camouflage and answered it with a neighbour selector trained by reinforcement learning [19]. Liu and colleagues attacked the same inconsistency from the feature and relation sides [20]. Later work

combined neighbour selection with resampling to handle the extreme imbalance that fraud labels always show [22], and FRAUDRE addressed inconsistency and imbalance together [23].

Time is the other axis. Fraud arrives in bursts, so spatial structure alone misses the pattern, and spatial temporal attention has been used to capture both [24]. Tang and colleagues took a spectral view and argued that anomalies concentrate in the high frequency part of the graph signal, which suggests that the usual low pass filtering of graph convolution is working against the task [25]. That argument deserves more attention than it has received. Table 1 compares the fraud specific designs.

Table 1 Fraud specific designs and the difficulty each one targets, following the descriptions in the cited papers.

Method	Problem addressed	Mechanism	Source
Heterogeneous account graph	Entities of different types	Type aware aggregation	[11], [14]
Semi supervised attention	Few labels	Attention over neighbours	[16]
CARE-GNN	Camouflaged neighbours	Learned neighbour selection	[19]
PC-GNN	Class imbalance	Neighbour sampling and resampling	[22]
FRAUDRE	Inconsistency and imbalance	Both, in one training objective	[23]
Spatial temporal attention	Bursty behaviour	Attention across time and structure	[24]

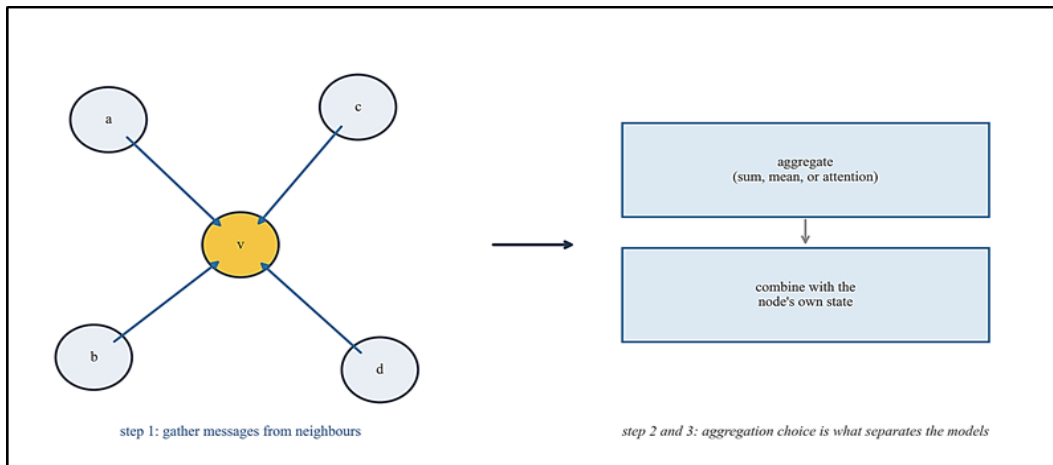


Figure 2: One round of message passing at node v. The choice of aggregation function is what separates the models discussed in Section II [5], [6], [7], [10], [13].

V. EVALUATION, AND HOW IT GOES WRONG

This is the part of the literature that needs the most correction. Pendlebury and colleagues showed that malware classifiers evaluated with random splits report accuracies that collapse when the split respects time, because a random split lets the model learn from samples that appeared after the ones it is tested on [17]. The same criticism applies to fraud, where a random split places later transactions in training.

Three rules follow. Split by time, always. Report precision at the alert budget an analyst can actually review, since a detector that flags five percent of accounts is useless whatever its recall. And report the class balance, because accuracy on a dataset with one positive in a thousand says nothing at all. Dal Pozzolo and colleagues make the last point carefully for card fraud and add the further complication that labels arrive with a delay [12]. Table 2 lists the failure modes we found in this reading.

Table 2. Evaluation failure modes observed across the reviewed works, with the correction each one calls for.

Failure mode	Effect on reported results	Correction	Source
Random train and test split	Future information leaks backwards	Split by time	[17]
Accuracy on imbalanced data	Majority class dominates the score	Precision at a fixed alert budget	[12]
Ignoring label delay	Optimistic supervision	Simulate the real labelling lag	[12]
Single dataset reporting	Family or institution overfitting	Cross dataset evaluation	[26], [30]

Privacy is the constraint that architecture papers tend to skip. Transaction graphs contain personal financial data, and building one across institutions is a legal problem before it is a technical one. Approaches based on homomorphic encryption and differential privacy offer a path [32], but they change what the model can see, and no reviewed paper reports detection quality under such a constraint.

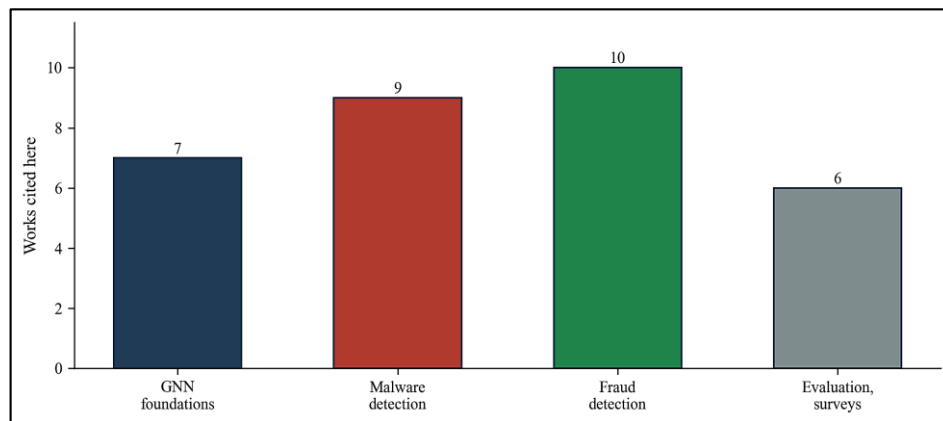


Figure. 3: The thirty two works cited here, coded by the role they play in this review. The chart counts our own reference list and reports no experimental result.

VI. OPEN PROBLEMS

Scale comes first. Payment graphs run to billions of edges and rebuild continuously, while most published evaluations use static graphs small enough to fit in memory. Second, adversarial robustness of the graph itself. An attacker who can add edges can move a prediction, and camouflage is only the simplest version of that attack.

Third, explanation. A fraud alert leads to an account freeze, and a freeze needs a reason that a human can check. Fourth, transfer across institutions and across malware families, which the surveys identify as open and which no reviewed method claims to have solved [27], [30]. Anomaly detection outside these two domains faces the same list [28].

VII. CONCLUSION

Graph neural networks earn their place in both problems, for the same reason in each: the relation between entities carries evidence that per entity features do not. The gains reported in the fraud literature are largest where the method confronts camouflage and imbalance head on rather than treating the graph as ordinary input [19], [22], [23].

Our main recommendation is about evaluation rather than architecture. A malware or fraud result from a random split should be treated as unreported until it is repeated with a temporal split [17]. Much of the apparent progress in this area would look different under that rule, and the literature would be more useful to the people who have to deploy these systems.

REFERENCES

- [1] X. Hu, T. Chiueh, and K. G. Shin, "Large-scale malware indexing using function-call graphs," in *Proc. ACM Conf. Comput. Commun. Security (CCS)*, 2009, pp. 611–620.
- [2] S. Bhattacharyya, S. Jha, K. Tharakunnel, and J. C. Westland, "Data mining for credit card fraud: A comparative study," *Decision Support Systems*, vol. 50, no. 3, pp. 602–613, 2011.
- [3] H. Gascon, F. Yamaguchi, D. Arp, and K. Rieck, "Structural detection of android malware using embedded call graphs," in *Proc. ACM Workshop Artif. Intell. Security (AISec)*, 2013, pp. 45–54.
- [4] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, and K. Rieck, "DREBIN: Effective and explainable detection of android malware in your pocket," in *Proc. Netw. Distrib. Syst. Security Symp. (NDSS)*, 2014.
- [5] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017.
- [6] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 1024–1034.
- [7] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2017, pp. 1263–1272.
- [8] Y. Ye, T. Li, D. Adjeroh, and S. S. Iyengar, "A survey on malware detection using data mining techniques," *ACM Computing Surveys*, vol. 50, no. 3, pp. 41:1–41:40, 2017.
- [9] E. Mariconti, L. Onwuzurike, P. Andriotis, E. De Cristofaro, G. Ross, and G. Stringhini, "MaMaDroid: Detecting android malware by building Markov chains of behavioral models," in *Proc. NDSS*, 2017.

- [10] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *Proc. ICLR*, 2018.
- [11] Z. Liu, C. Chen, X. Yang, J. Zhou, X. Li, and L. Song, "Heterogeneous graph neural networks for malicious account detection," in *Proc. ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, 2018, pp. 2077–2085.
- [12] A. Dal Pozzolo, G. Boracchi, O. Caelen, C. Alippi, and G. Bontempi, "Credit card fraud detection: A realistic modeling and a novel learning strategy," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 8, pp. 3784–3797, 2018.
- [13] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *Proc. ICLR*, 2019.
- [14] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu, "Heterogeneous graph attention network," in *Proc. World Wide Web Conf. (WWW)*, 2019, pp. 2022–2032.
- [15] J. Yan, G. Yan, and D. Jin, "Classifying malware represented as control flow graphs using deep graph convolutional neural network," in *Proc. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, 2019, pp. 52–63.
- [16] D. Wang *et al.*, "A semi-supervised graph attentive network for financial fraud detection," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, 2019, pp. 598–607.
- [17] F. Pendlebury, F. Pierazzi, R. Jordaney, J. Kinder, and L. Cavallaro, "TESSERACT: Eliminating experimental bias in malware classification across space and time," in *Proc. USENIX Security Symp.*, 2019, pp. 729–746.
- [18] J. Zhou *et al.*, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020.
- [19] Y. Dou, Z. Liu, L. Sun, Y. Deng, H. Peng, and P. S. Yu, "Enhancing graph neural network-based fraud detectors against camouflaged fraudsters," in *Proc. ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, 2020, pp. 315–324.
- [20] Z. Liu, Y. Dou, P. S. Yu, Y. Deng, and H. Peng, "Alleviating the inconsistency problem of applying graph neural network to fraud detection," in *Proc. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2020, pp. 1569–1572.
- [21] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, 2021.
- [22] Y. Liu *et al.*, "Pick and choose: A GNN-based imbalanced learning approach for fraud detection," in *Proc. Web Conf. (WWW)*, 2021, pp. 3168–3177.
- [23] G. Zhang *et al.*, "FRAUDRE: Fraud detection dual-resistant to graph inconsistency and imbalance," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, 2021, pp. 867–876.
- [24] D. Cheng, X. Wang, Y. Zhang, and L. Zhang, "Graph neural network for fraud detection via spatial-temporal attention," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 8, pp. 3800–3813, 2022.
- [25] J. Tang, J. Li, Z. Gao, and J. Li, "Rethinking graph neural networks for anomaly detection," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2022, pp. 21076–21089.
- [26] S. Freitas, R. Duggal, and D. H. Chau, "MalNet: A large-scale image database of malicious software," in *Proc. ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, 2022, pp. 3948–3952.
- [27] G. Pang, C. Shen, L. Cao, and A. van den Hengel, "Deep learning for anomaly detection: A review," *ACM Computing Surveys*, vol. 54, no. 2, pp. 38:1–38:38, 2022.
- [28] Y. Wu, H.-N. Dai, and H. Tang, "Graph neural networks for anomaly detection in industrial Internet of Things," *IEEE Internet Things J.*, vol. 9, no. 12, pp. 9214–9231, 2022.
- [29] Y. Hei *et al.*, "Hawk: Rapid android malware detection through heterogeneous graph attention networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 4, pp. 4703–4717, 2024.
- [30] S. Motie and B. Raahemi, "Financial fraud detection using graph neural networks: A systematic review," *Expert Systems with Applications*, vol. 240, 2024, Art. no. 122156.
- [31] K. Abraham, "Ransomware evolution: How AI is changing both attack and defence strategies," *Int. J. Inf. Technol. Res. Stud. (IJITRS)*, vol. 1, no. 3, p. 164, Oct. 2025, doi: 10.63090/ijitrs/3139.3209.0015.
- [32] M. J. Komban, "Privacy-preserving techniques in data mining: A comprehensive analysis of homomorphic encryption and differential privacy approaches," *Int. J. Inf. Technol. Res. Stud. (IJITRS)*, vol. 1, no. 2, p. 127, Jul. 2025, doi: 10.63090/ijitrs/3139.3209.0011.