Rethinking Temporal Graph Transformers for Outlier Detection

Kay Liu*
University of Illinois Chicago
Chicago, Illinois, USA
zliu234@uic.edu

Jiahao Ding
Amazon Web Services
New York, New York, USA
djiahao@amazon.com

MohamadAli Torkamani
Amazon Web Services
New York, New York, USA
alitor@amazon.com

Abstract

Graph outlier detection identifies substructures in graphs that significantly deviate from normal patterns. Traditional graph outlier detection methods are mostly limited to static graphs, which overlook the dynamic nature of real-world graphs and ignore temporal signals providing critical information for detecting outliers. Recently, Transformers revolutionized machine learning on time-series data. However, existing Transformers on temporal graphs face limitations due to their reliance on temporal subgraph extraction, restricted receptive fields, and suboptimal generalization capability beyond link prediction. To address these challenges, we propose TGFormer, a novel Temporal Graph Transformer for outlier detection. TGFormer leverages global attention to model both structural and temporal dependencies within temporal graphs. To improve the scalability, TGFormer partitions large temporal graphs into spatiotemporal patches. These patches are processed by a hierarchical Transformer architecture, which includes intra-patch, inter-patch, and temporal Transformers. We conduct experiments on two public datasets compared with a set of baselines, including graph neural networks, graph outlier detectors, and Transformers based methods. Experimental results demonstrate the superiority of TGFormer. Our analysis and experiments on efficiency metrics prove efficiency of TGFormer.

CCS Concepts

• **Mathematics of computing** → *Time series analysis*; *Graph algorithms*; • **Security and privacy** → *Software and application security*.

Keywords

Graph Outlier Detection, Temporal Graphs, Graph Transformers

ACM Reference Format:
Kay Liu, Jiahao Ding, and MohamadAli Torkamani. 2018. Rethinking Temporal Graph Transformers for Outlier Detection. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX)*. ACM, New York, NY, USA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

*Work done during an internship at Amazon Web Services.

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Outlier detection, as a prominent task in artificial intelligence, aims to identify outlying data points that deviate significantly from the normal patterns. Outlier detection has become a critical task in identifying cyberattacks [24], fraudulent activities [6], fake news [3], etc. Graph-structured data, which is a geometric data structure in the form of nodes connected by edges, has been widely used to model complex relationships across numerous domains, including citations [25], molecules [20], and so on. Graph outlier detection, as a prominent task in graph machine learning, aims to identify outlying substructures in graphs that deviate significantly from the normal patterns. Because of complex non-Euclidean nature of graph data, graph outlier detection is inherently challenging [11]. Traditional methods for graph outlier detection have mainly focused on static graphs, where the structure and attributes do not change over time [10]. However, in real-world applications, graphs typically evolve over time, and temporal signals provide abundant information for detecting outliers. In these cases, traditional methods for static graphs fall short when applied to temporal graphs, as the important temporal aspect for outlier detection is fully ignored.

Recently, Transformers have revolutionized machine learning on language [16] and vision [2] with their powerful ability to model complex dependencies in data through self-attention mechanisms. Unlike traditional recurrent neural network architectures that rely heavily on sequential processing, Transformers utilize attention mechanisms to weigh the importance of different parts of the input data, allowing them to capture long-range dependencies effectively and efficiently. This self-attention mechanism in Transformers offers a promising direction by integrating temporal dynamics into the graph representation learning process.

Some efforts have been made to applying Transformers on temporal graphs. DyGFormer extracts one-hop interactions and feeds their neighbor, link, time, and co-occurrence encoding into Transformer to capture temporal edges between nodes [26]. Similarly, SimpleDyG also models neighbors in temporal graphs as a sequence and introduces a temporal alignment technique to capture temporal evolution patterns [22]. Despite their potential, existing methods Transformers on temporal graph have several limitations. Firstly, they require temporal subgraph extraction for each edge. This subgraph extraction process can be slow and hard to process in parallel, limiting the scalability of Transformers. In addition, due to efficiency constraint, current methods only extract one-hop neighboring subgraph. It greatly limits the receptive field for Transformers, overlooking higher order structural information. Also, these Transformers are trained on link prediction task, they have suboptimal generalization capability to outlier detection at node level.

To bridge these gaps, we explore to adopt global spatiotemporal attention on the whole temporal graph for node level outlier

detection, and propose TGFormer, a novel paradigm to apply Transformers on temporal graphs. It allows TGFormer models not only the structural dependencies within the graph but also the temporal dependencies. Inspired by visual patching method used in vision model [1], we divide the large temporal graph into spatiotemporal patches to alleviate the challenge of scalability. For the spatial aspect, we use a graph clustering algorithm to partition the large graph into relatively small clusters. For the temporal aspect, we aggregate the interaction within the clusters into a snapshot with a specific time length. These obtained patches are fed into a hierarchical Transformer, including an intra-patch Transformer, an inter-patch Transformer, and a temporal Transformer. Our contributions can be summarized as follows:

- We propose TGFormer and make the first attempt to leverage global spatiotemporal attention in outlier detection on temporal graphs.
- We introduce a spatiotemporal patching inspired by visual models, significantly improving the scalability of TGFormer and enabling outlier detection on large-scale temporal graphs.
- Extensive experiments demonstrate the effectiveness and efficiency of proposed TGFormer.

2 Methodology

Figure 1 provides an overview of the proposed TGFormer. We aim to conduct global attention among the whole temporal graph on both spatial and temporal aspects. However, scalability of all pairs attention remains a great challenge. We start from problem definition in Section 2.1, and introduce our spatiotemporal patching method to divide large-scale temporal graphs into relatively managable patches in Section 2.2. Then, we define the efficient architecture of proposed TGFormer in Section 2.3, following by detailed design of intra-patch Transformers that further reduce computational complexity in Section 2.4. We summarize the training procedure of TGFormer in Section 2.5.

2.1 Problem Definition

Consider a genaral discrete temporal graph $\mathcal{G}$ consisting of a sequence of graph snapshots $\{\mathcal{G}^t\}_{t=1}^T$. In a snapshot $\mathcal{G}^t = (\mathcal{V}^t, \mathcal{E}^t, \mathbf{X}^t)$ at timestamp t, $\mathcal{V}^t$ represents the node set, $\mathcal{E}^t$ represents the edge set, and $\mathbf{X}^t$ is the feature matrix. $\mathcal{V} = \bigcup_{t=1}^T \mathcal{V}^t$ is the total node set of size N, while $\mathcal{E} = \bigcup_{t=1}^T \mathcal{E}^t$ is the total edge set. Given a partially labeled temporal graph $\mathcal{G}$, the problem of supervised graph outlier detection is a binary classification task that learns a detector $f : \mathcal{V} \to \{0, 1\}^N$ that classifies every node in $\mathcal{G}$ to either an inlier (0) or an outlier (1).

2.2 Spatiotemporal Patching

Our approach is inspired by the video generation model Sora [1], drawing parallels between video data and temporal graph data. A temporal graph is a sequence of graph snapshots with nodes, akin to how a video is a sequence of image frames with pixels. For instance, a 1-minute 1080p video at 24Hz consists of approximately 3 billion pixels. Directly treating these pixels as tokens and feeding them into Transformers, which have quadratic complexity, would result in prohibitive computational cost. In computer vision, this complexity is managed by dividing the video data into visual patches, segmented over time and space. In this work, we adopt a similar strategy to divide large temporal graphs $\mathcal{G}$ into small spatiotemporal patches $\{p_c^s\}$, employing time slotting and graph clustering. Figure 1a shows an example of spatiotemporal patching, where a column represents a graph cluster and a row corresponds to a timeslot.

For the temporal aspect, we aggregate the edges in graph snapshots over specific time lengths Δt, merging the node set $\mathcal{V}^s = \bigcup_{t=t_i}^{t_i+\Delta t} \mathcal{V}^t$ and the edge set $\mathcal{E}^s = \bigcup_{t=t_i}^{t_i+\Delta t} \mathcal{E}^t$. Here, Δt is a hyperparameter that is minimized within the constraints of available memory. A smaller Δt allows for more fine-grained temporal processing and reduces information loss.

For the spatial aspect, we use METIS [7] to partition the aggregated large graph into relativelly small clusters $\{\mathcal{V}_c\}$, where $\mathcal{V} = \bigcup_{c=1}^C \mathcal{V}_c$. METIS is an off-the-shelf graph clustering algorithm that is highly scalable and efficient, and does not incur significant overhead as a preprocessing step.

2.3 Transformer Architecture

The architecture of TGFormer is shown in Figure 1b. Our proposed temporal graph Transformer, TGFormer, takes these spatiotemporal patches as input, while the output is derived from a prediction head tailored to specific tasks, i.e., node level outlier detection in our case. It can be as simple as logistic regression. TGFormer separates spatial Transformers and temporal Transformers (TFormer) to reduce attention complexity. The spatial Transformer is divided into intra-patch Transformers (GFormer) and inter-patch Transformers (PFormer) to further reduce complexity.

Within each patch, an intra-patch Transformer conducts all pair attention within the patch to obtain the embeddings for each node.

$$\mathbf{Z}_c^s = \text{GFormer}(\mathbf{X}_c^s), \tag{1}$$

where $\mathbf{X}_c^s \in \mathbb{R}^{|\mathcal{V}_c| \times d}$ is the patch node feature, $\mathbf{Z}_c^s \in \mathbb{R}^{|\mathcal{V}_c| \times d'}$ is the intra-patch embedding, and d and d' are feature dimension and the hidden dimension, respectively.

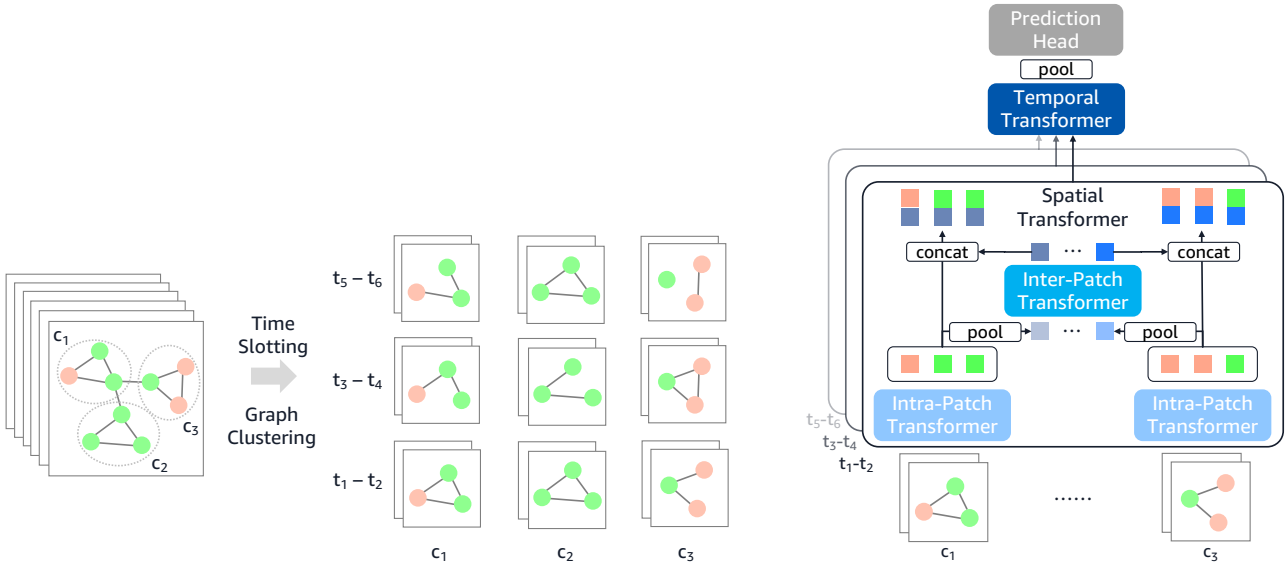
To extend the receptive field beyond individual patches, we employ an inter-patch Transformer. The intra-patch embedding undergo pooling to produce embedding of patch:

$$\mathbf{p}_c^s = \text{pooling}(\mathbf{Z}_c^s), \tag{2}$$

where $\mathbf{p}_c^s \in \mathbb{R}^{d'}$ is the embedding of patch c and mean pooling is used in our implementation. The patch embeddings $\mathbf{P} = [\mathbf{p}_1, ..., \mathbf{p}_C]^\intercal$ are then processed by the inter-patch Transformer to estimate $\bar{\mathbf{P}} = [\bar{\mathbf{p}}_1, ..., \bar{\mathbf{p}}_C]^\intercal$:

$$\bar{\mathbf{P}} = \text{PFormer}(\mathbf{P}). \tag{3}$$

When node i in cluster c, the intra-patch embeddings $\mathbf{z}_i^s$ is concatenated with the corresponding obtained patch embedding $\bar{\mathbf{p}}_c$ to form the final spatial Transformer output $\bar{\mathbf{z}}_i^s = \text{concat}(\mathbf{z}_i^s, \bar{\mathbf{p}}_c$. To manage memory consumption efficiently, we enable mini-batch training, updating the embedding of one patch $\mathbf{p}_i$ at a time while keeping others frozen.



(a) Spatiotemporal patching in TGFormer. **(b) Architecture of TGFormer.**

Figure 1: An overview of the proposed TGFormer.

2.4 Intra-Patch Transformer

Our intra-patch Transformer is specially designed to encode the graph structure. A vanilla Transformer would ignore the graph structure, so we integrate a Graph Neural Network (GNN) as a residual connection within the intra-patch Transformer. The output is a summation of the GNN and Transformer outputs. Additionally, we employ a kernel method [21] to approximate the all-pair attention, reducing complexity from quadratic to linear, allowing for larger patch sizes. The output of GFormer is formulate as:

$$\text{GFormer}(\mathbf{X}_c^s) = \alpha \cdot \text{GNN}(\mathbf{X}_c^s) + (1-\alpha) \cdot \text{AppxAttn}(\mathbf{X}_c^s), \quad (4)$$

where α is a hyperparameter weight. We adopt Graph Convolutional Networks (GCNs) [8] as the GNN implementation.

2.5 Training

The temporal Transformer operates as a vanilla Transformer, computing attention over spatial embeddings across timeslots.

$$\bar{\mathbf{Z}}_i = \text{TFormer}(\mathbf{Z}_i), \quad (5)$$

where $\mathbf{Z}_i = [\mathbf{z}_i^1, ..., \mathbf{z}_i^S]^\intercal$ is the spatial embedding matrix of node v_i. The result $\bar{\mathbf{Z}}_i$ is pooled into a final embedding across the temporal dimension, either by mean or concatenation.

$$\mathbf{e}_i = \text{pooling}(\bar{\mathbf{Z}}_i). \quad (6)$$

The prediction head is adaptable to different tasks, e.g., binary classification in our case:

$$\hat{y}_i = \text{LogisticRegression}(\mathbf{e}_i). \quad (7)$$

The output $\hat{y}_i$ is used to estimate loss values with the binary cross-entropy loss function for end-to-end training:

$$\mathcal{L} = \frac{1}{N}\sum_i^N y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i). \quad (8)$$

This methodology ensures that TGFormer effectively captures both spatial and temporal dependencies in temporal graphs, improving scalability and generalization for graph outlier detection.

3 Experiments

To evaluate TGFormer and compare with contemporary methods, we conduct experiments in a unified environment, which is partially adapted from GADBench [14] and DyGLib [26].

3.1 Experimental Setups

3.1.1 Datasets. Table 2 provides the statistics of the data sets used in our experiments. In the table, #Nodes stands for the number of nodes, and #Edges stands for the number of edges. #Feat. denotes the raw feature dimension. #Outlier denotes the outlier ratio. #Time is the number of timestamps, i.e., the number of graph snapshots. The detailed descriptions for each dataset are as follows:

DGraph [6]: DGraph is a large-scale graph dataset provided by Finvolution Group, encompassing approximately 3 million nodes, 4 million dynamic edges, and 1 million node labels. The nodes represent user accounts within a financial organization that offers personal loan services, with edges indicating that one account has designated the other as an emergency contact. Nodes labeled as fraud correspond to users exhibiting delinquent financial behavior. For accounts with borrowing records, outliers are identified as accounts with a history of overdue payments, while inliers are those without such a history. The dataset also includes 17 node features derived from user profile information.

Table 1: AUC, AP, and Recall@k (%) in graph outlier detection. Best score in bold.

Category	Method	Elliptic AUC	Elliptic AP	Elliptic Rec@k	DGraph AUC	DGraph AP	DGraph Rec@k
GNN	SGC [19]	75.4±1.2	12.8±0.7	11.0±1.7	66.1±0.3	2.4±0.1	4.2±0.2
	GCN [8]	81.4±1.8	21.9±2.9	25.0±6.0	75.9±0.2	4.0±0.1	7.1±0.2
	GraphSAGE [5]	85.3±0.7	32.9±6.0	37.3±4.6	75.6±0.2	3.8±0.1	7.0±0.4
	GAT [17]	84.9±1.9	25.2±5.6	27.9±11.3	75.9±0.2	3.9±0.1	7.4±0.2
	GIN [23]	82.7±2.0	23.5±4.9	27.3±8.3	74.0±0.2	3.3±0.1	5.9±0.2
Detector	GAS [9]	85.6±1.6	27.9±6.6	34.6±9.6	76.0±0.2	3.8±0.1	6.8±0.2
	PCGNN [12]	85.8±1.8	35.6±10.2	40.4±12.0	72.0±0.3	2.8±0.0	5.0±0.2
	GATSep [27]	86.0±1.4	26.4±4.5	31.3±8.8	76.0±0.2	3.9±0.1	7.5±0.3
	BWGNN [15]	85.2±1.1	26.0±3.5	31.7±6.2	76.3±0.1	4.0±0.1	7.5±0.3
	GHRN [4]	85.4±1.9	27.7±6.6	33.3±10.3	76.1±0.1	4.0±0.1	7.5±0.2
Transformer	Graph Transformer [13]	85.1±1.5	25.1±4.5	26.3±11.1	75.8±0.1	3.9±0.1	7.5±0.2
	DyGFormer [26]	85.4±1.8	30.2±5.3	33.7±9.4	77.5±0.2	3.9±0.1	7.5±0.2
Ours	w/o time	87.4±0.9	57.3±3.9	57.3±3.2	76.4±0.3	3.6±0.1	5.8±0.4
	w/o inter-patch	88.7±1.0	60.8±5.0	60.8±1.7	78.0±0.4	**4.1**±0.1	**6.5**±0.4
	w/o intra-patch	88.3±1.4	59.6±7.0	**60.9**±5.9	77.7±0.0	3.9±0.1	6.0±0.4
	w/o GNN	87.8±0.8	49.3±5.2	52.2±3.8	72.5±0.2	2.8±0.1	4.4±0.3
	TGFormer	**89.2**±0.5	**64.4**±5.9	60.7±2.6	**78.3**±0.3	**4.1**±0.1	**6.5**±0.4

Table 2: Statistics of data sets.

Dataset	#Nodes	#Edges	#Feat.	#Outliers	#Time
Elliptic	203,769	234,355	165	9.8%	49
DGraph	3,700,550	4,300,999	17	1.3%	821

Elliptic [18]: This graph dataset contains over 200,000 Bitcoin transaction nodes, 234,000 directed payment flow edges, and 165 dimensional node features. The dataset maps Bitcoin transactions to real-world entities, categorizing them into both licit categories, including exchanges, wallet providers, miners, and legal services, and illicit categories, such as scams, malware, terrorist organizations, ransomware, and Ponzi schemes.

3.1.2 Metrics. We follow the existing literature in graph outlier detection [11, 14] to evaluate the performance of graph outlier detectors with three commonly used metrics:

AUC: area under receiver operating characteristic curve, which is constructed by plotting the true positive rate against the false positive rate across varied determined threshold levels.

AP: average precision, which also called area under precision-recall curve. It represented as the weighted average of precision values obtained at each threshold.

Rec@k: recall at k, where the value of k is set to the number of actual outliers present in the dataset.

3.2 Experiments Results

Tabel 1 shows the comparison results between different types of baselines and TGFormer. The baselines we compared include general graph neural networks (GNN), graph outlier detectors (Detector), Transformer-based methods (Transformer), and the variants of TGFormer. From the figure, we can observe that, among all these methods, our method reaches the best performance of 78.3 in AUC on DGraph. In addition, the methods consider temporal information is better than static methods. This observation proves that temporal information matters in the task of graph outlier detection. Moreover, our variants removing on of the components perform worse than the complete model, showing that all of the module inside our method contribute to our effectiveness.

3.3 Efficiency Analysis

On the top of effectiveness, efficiency is another important aspect for the application of TGFormer. We estimate and empirically evaluate the efficiency of DyGFormer and TGFormer. To ensure fair comparison, we evaluate the training on local Apple M3 Pro CPUs. The results are presented in Table 3. #Param is the number of learnable parameters, Time is the training time for one epoch, and the Memory represents the main memory consumption in CPU training. TGFormer shows a significantly higher efficiency in terms of parameters, training time, and memory consumption.

Table 3: Efficiency results of temporal graph Transformers.

Dataset	Model	#Param	Time (s)	Memory
Elliptic	DyGFormer	1,087,035	132	2,568M
	TGFormer	31,329	40	769M
DGraph	DyGFormer	1,087,035	3,037	49G
	TGFormer	6,865	759	16G

4 Conclusion

By rethinking the application of temporal graph Transformers for outlier detection, this study makes the first attempt to leverage global spatiotemporal attention in outlier detection on temporal graphs and propose TGFormer. Our method significantly improves scalability through spatiotemporal patching, thus enabling outlier detection on large-scale temporal graphs. We aims to push the boundaries of current methodologies and establish a foundation for future advancements in the broader field of temporal graph modeling. We anticipate that our findings will inspire the development of more sophisticated models capable of adeptly handling the temporal and structural complexities inherent in graph data.

References

[1] T. Brooks, B. Peebles, C. Holmes, W. DePue, Y. Guo, L. Jing, D. Schnurr, J. Taylor, T. Luhman, E. Luhman, C. Ng, R. Wang, and A. Ramesh. Video generation models as world simulators. 2024.

[2] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020.

[3] Y. Dou, K. Shu, C. Xia, P. S. Yu, and L. Sun. User preference-aware fake news detection. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, pages 2051–2055, 2021.

[4] Y. Gao, X. Wang, X. He, Z. Liu, H. Feng, and Y. Zhang. Addressing heterophily in graph anomaly detection: A perspective of graph spectrum. In *Proceedings of the ACM Web Conference 2023*, pages 1528–1538, 2023.

[5] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[6] X. Huang, Y. Yang, Y. Wang, C. Wang, Z. Zhang, J. Xu, L. Chen, and M. Vazirgiannis. Dgraph: A large-scale financial dataset for graph anomaly detection. *Advances in Neural Information Processing Systems*, 35:22765–22777, 2022.

[7] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing*, 20(1):359–392, 1998.

[8] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2016.

[9] A. Li, Z. Qin, R. Liu, Y. Yang, and D. Li. Spam review detection with graph convolutional networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 2703–2711, 2019.

[10] K. Liu, Y. Dou, X. Ding, X. Hu, R. Zhang, H. Peng, L. Sun, and S. Y. Philip. Pygod: A python library for graph outlier detection. *Journal of Machine Learning Research*, 25(141):1–9, 2024.

[11] K. Liu, Y. Dou, Y. Zhao, X. Ding, X. Hu, R. Zhang, K. Ding, C. Chen, H. Peng, K. Shu, et al. Bond: Benchmarking unsupervised outlier node detection on static attributed graphs. *Advances in Neural Information Processing Systems*, 35:27021–27035, 2022.

[12] Y. Liu, X. Ao, Z. Qin, J. Chi, J. Feng, H. Yang, and Q. He. Pick and choose: a gnn-based imbalanced learning approach for fraud detection. In *Proceedings of the web conference 2021*, pages 3168–3177, 2021.

[13] Y. Shi, H. Zhengjie, S. Feng, H. Zhong, W. Wang, and Y. Sun. Masked label prediction: Unified message passing model for semi-supervised classification. In *International joint conference on artificial intelligence*, pages 1548–1554, 08 2021.

[14] J. Tang, F. Hua, Z. Gao, P. Zhao, and J. Li. Gadbench: Revisiting and benchmarking supervised graph anomaly detection. *Advances in Neural Information Processing Systems*, 36, 2024.

[15] J. Tang, J. Li, Z. Gao, and J. Li. Rethinking graph neural networks for anomaly detection. In *International Conference on Machine Learning*, pages 21076–21089. PMLR, 2022.

[16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[17] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[18] M. Weber, G. Domeniconi, J. Chen, D. K. I. Weidele, C. Bellei, T. Robinson, and C. E. Leiserson. Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics. *arXiv preprint arXiv:1908.02591*, 2019.

[19] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR, 2019.

[20] L. Wu, Z. Hou, J. Yuan, Y. Rong, and W. Huang. Equivariant spatio-temporal attentive graph networks to simulate physical dynamics. *Advances in Neural Information Processing Systems*, 36, 2024.

[21] Q. Wu, W. Zhao, Z. Li, D. P. Wipf, and J. Yan. Nodeformer: A scalable graph structure learning transformer for node classification. *Advances in Neural Information Processing Systems*, 35:27387–27401, 2022.

[22] Y. Wu, Y. Fang, and L. Liao. On the feasibility of simple transformer for dynamic graph modeling. In *Proceedings of the ACM on Web Conference 2024*, pages 870–880, 2024.

[23] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2018.

[24] F. Yang, J. Xu, C. Xiong, Z. Li, and K. Zhang. {PROGRAPHER}: An anomaly detection system based on provenance graph embedding. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 4355–4372, 2023.

[25] Z. Yang, W. Cohen, and R. Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pages 40–48. PMLR, 2016.

[26] L. Yu, L. Sun, B. Du, and W. Lv. Towards better dynamic graph learning: New architecture and unified library. *Advances in Neural Information Processing Systems*, 36:67686–67700, 2023.

[27] A. Zimek, R. J. Campello, and J. Sander. Ensembles for unsupervised outlier detection: challenges and research questions a position paper. *Acm Sigkdd Explorations Newsletter*, 15(1):11–22, 2014.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009